6 Scanner Support

The 700 Series Color Mobile Computer is available with imaging or laser scanning technologies, including the following:

APS Linear Imager: *(standard for 730 Computers)* — includes the EV10 Scan Engine	Reads 1D symbologies and PDF417 bar codes. Linear imaging using Vista Scanning technology reads low-contrast bar codes, laminated bar codes, and bar codes displayed on CRT or TRT displays. This imaging uses harmless LEDs for illumination and does not require any warning labels. Vista Scanning is more reliable than lasers as it is a completely solid state with no moving parts or oscillating mirrors.
2D Imager: *(not supported on 730 Computers)* — includes the IT4000 Scan Engine	This decodes several stacked 1D and 2D symbologies, including PDF417, Data Matrix, and MaxiCode without "painting." It can also read 1D codes from any orientation, for example the scan beam does not need to align perpendicular to the symbol to read it. Photography is a secondary application; the lens in the device favors bar code reading. Photos are 640x480, 256 gray-scale.
1D Laser Scanner: *(not supported on 730 Computers)* — includes the SE900, SE900HS, SE900-S6, and SE900HS-S6 scan engines	Traditional laser scanner that decodes 1D bar codes.
PDF417 Laser Scanner: *(not supported on 730 Computers)*	Higher speed laser scanner that can read PDF417 labels by "painting" the label.

A 700 Color Imager Demo application demonstrates the more common features of the 700 Color Computer imager. See the *Imager Demo V1.0 User's Guide* on the *Intermec Developer's Library CD* for information.

Scanner Control and Data Transfer

Note: To use the methods described below, enable Data Collection functionality on the 700 Computer using the bootloader configuration menu.

The Data Server and associated software provide several ways to manipulate scanner control and data transfer between the scanner subsystem and user applications:

- **Automatic Data Collection COM Interfaces**:
  These COM interfaces allow user applications to receive bar code data, and configure and control the bar code reader engine.
- **ITCAxBarCodeReaderControl functions**:
  These ActiveX controls allow user applications to collect bar code data from the scanner, to configure the scanner, and to configure audio and visual notification when data arrives.
- **ITCAxReaderCommand functions**:
  Use these ActiveX controls to modify and retrieve configuration information using the reader interface commands.
- **Scanning EasySet bar code labels**:
  You can use the EasySet® bar code creation software from Intermec Technologies Corporation to print configuration labels. Scan the labels to change the scanner configuration and data transfer settings.

For more information, see the *SDK User's Manual* provided with your *Windows CE/PocketPC SDK.*

EasySet Software

Use the Intermec EasySet software to print configuration labels you can scan to change your configuration settings. For more information, see the EasySet online help. EasySet is available from the Intermec Data Capture web site.

Data Collection Configuration

For Units with PSM Builds Older than 3.00

Scanner settings for the 700 Series Computer can be configured via the **Data Collection** control panel applet. From the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Data Collection**. See *Appendix A*, "*Configurable Settings*" for more information about the following parameters. *Note that these are in alphabetical order.*

- Codabar *(page 304)*
- Code 11 *(page 318)*
- Code 128 *(page 307)*
  - Code 128 Options *(page 308)*
  - Code 128 FNC1 Character *(page 309)*
- Code 39 *(page 302)*
- Code 93 *(page 306)*
  - Code 93 Length *(page 306)*
- Data Matrix *(page 320)*
- Interleaved 2 of 5 *(page 315)*
- Matrix 2 of 5 *(page 316)*
- MaxiCode *(page 321)*
- MSI *(page 311)*
- PDF417 *(page 312)*
  - Macro PDF *(page 312)*
  - Micro PDF417 *(page 314)*
- Plessey *(page 310)*
- QR Code *(page 319)*
- Standard 2 of 5 *(page 303)*
- Telepen *(page 317)*
- UPC/EAN *(page 305)*

For Units With PSM Build 3.00 or Newer

You can configure scanner settings for the 700 Series Computer via the Intermec Settings control panel applet. From the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > the **Intermec Settings** icon. See the *Intermec Computer Command Reference Manual* (P/N: 073529) for information about the settings you can configure with this applet. This online manual is available from the Intermec web site at www.intermec.com.

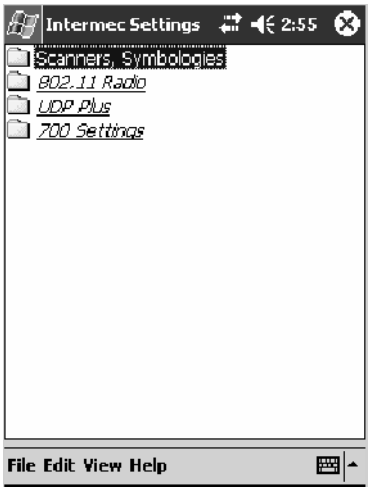
Internal Scanners

The Intermec Internal Scanner feature allows Automatic Data Collection (ADC) by accepting data from the COM1 port and wedging it into the keyboard interface. You can enable or disable this feature from the Today screen on the 700 Series Computer.

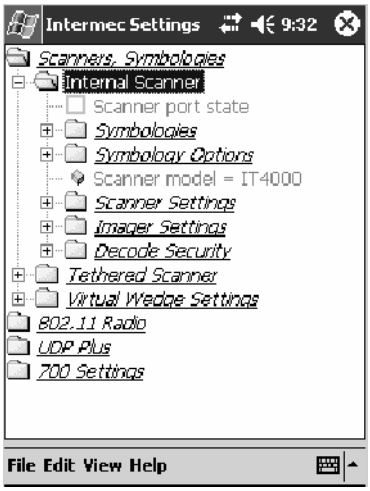
For Units With PSM Build 3.00 or Newer

Do the following before you configure your internal scanner from the Intermec Settings control panel applet. Information about the settings you can configure with this applet is described in the *Intermec Computer Command Reference Manual*. The online manual is available from the Intermec web site at www.intermec.com.

1 From the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > the **Intermec Settings** icon.



2 Double-tap the **Scanners, Symbologies** folder, then expand (+) the **Internal Scanner** option. *This sample screen is for the IT4000 scan engine.*



Scanner and Imager Settings

Depending on what is selected as the scanner model, image settings, decode security, scanner settings, and virtual wedge are configured from the Intermec Settings applet. See the the *Intermec Computer Command Reference Manual*, available from the Intermec web site at www.intermec.com, for more information about each enabled option.

Internal Scanner Supported Symbologies

See the following table for a guideline and Appendix B, "*Bar Code Symbologies*" for more information on each supported symbology:

Symbologies	EV10	IT4000	SE900	SE900HS	SE900-S6	SE900HS-S6
Code39	X	X	X	X	X	X
UPC/EAN	X	X	X	X	X	X
Code 128	X	X	X	X	X	X
Interleaved 2 of 5	X	X	X	X	X	X
Code 93	X	X	X	X	X	X
Codabar	X	X	X	X	X	X
Code 2 of 5	X	X	X	X	X	X
MSI	X	X	X	X	X	X
Plessey	X	X	X	X	X	X
Code 11	X	X	X	X	X	X
Matrix 2 of 5	X	X	X	X	X	X
Telepen	X	X	X	X	X	X
PDF417	X	X	X	X	X	X
Micro PDF417	X	X	X	X	X	X
MaxiCode		X				
Data Matrix		X				
QR Code		X				
RSS 14	X	X	X	X	Available in f/w Sxxp304	Available in f/w Sxxp304
RSS Limited	X	X	X	X	Available in f/w Sxxp304	Available in f/w Sxxp304
RSS Expanded	X	X	X	X	Available in f/w Sxxp304	Available in f/w Sxxp304
Codablock A	X	X	X	X	X	X
Codablock F	X	X	X	X	X	X
UCC Composite		X				

Tethered Scanners

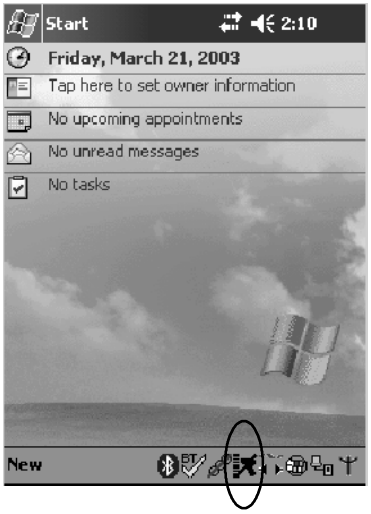
The Intermec Tethered Scanner feature allows Automatic Data Collection (ADC) by accepting data from the COM1 port and wedging it into the keyboard interface. You can enable or disable this feature from the Today screen on the 700 Series Computer.

The following information is divided between units with PSM Builds older than 3.00 *(next paragraph)* or units with PSM Builds 3.00 or newer *(starting on page 213)*.

For Units With PSM Builds Older than 3.00

Enabling and Disabling

On the 700 Series Computer, tap **Start** > **Today**. Tap the bar code scanner icon in the System Tray *(circled below)*. Initially, the bar code scanner icon indicates that this feature is disabled *(shown to the left)*.

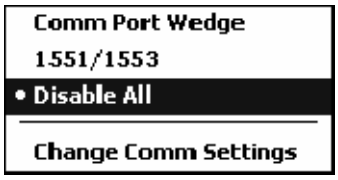


- Select **Comm Port Wedge** to send any data, coming into the 700 Series Computer through the COM1 port from an external input device, as keyboard data to an application on the desktop.

  For example, if you have Pocket Word running on your 700 Series Computer desktop, information scanned with a scanner connected to the COM1 port appears in the Word document. If another data collection application is running and is active on the 700 Series Computer, the scanned information appears in that application.

Note: When **Comm Port Wedge** is selected, regardless of the data sent by the external input device, you cannot control the device or the data format using any of the Intermec scanner control or data transfer APIs from the SDK or the internal Data Collection software. The external input device is governed by what software it has onboard to tell it how to scan, take pictures, or send the data elsewhere.

- Select **1551/1553** to enable the Sabre 1551E or 1553 Tethered Scanner to scan, then send data as keyboard data. The 1551/1553 Tethered Scanner has software onboard that translates scanned data into characters, so the running/active application does not need to know how to do that. All the scanner control and data transfer APIs will work with the 1551/1553 Tethered Scanner, so you can control the device.
- Select **Disable All** to disable this feature and use the COM1 port for another application, such as ActiveSync. An error message will result if this option were not selected, but this action was attempted. Similarly, if ActiveSync is using the COM1 port, and you select **Comm Port Wedge** or **1551/1553**, an error message will result. See "*Error Message*" for more information.



Error Message

If the COM1 port is used by another application, such as ActiveSync, neither the Comm Port Wedge nor the 1551/1553 Tethered Scanner can be enabled. As a result, the following message may appear. *Note that this message is for the Comm Port Wedge.* You must disable that application to free up the COM1 port before you can enable either the wedge or the scanner.

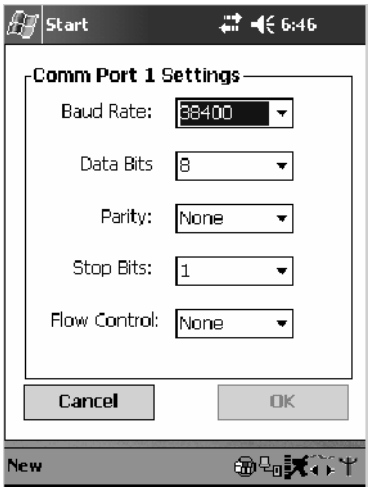
Changing Comm Settings

Tap **Change Comm Settings** to configure the settings for the COM1 port. Current settings are restored after a warm-boot is performed, but are lost after a cold-boot is performed. When these settings are not changed, the **OK** button is disabled (grayed out). When changes are made, tap **OK** after it is enabled to accept these changes.

- **Baud Rate**: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- **Data Bits**: 7 or 8
- **Parity**: None, Odd, Even, Mark, Space
- **Stop Bits**: 1 or 2
- **Flow Control**: None or Hardware

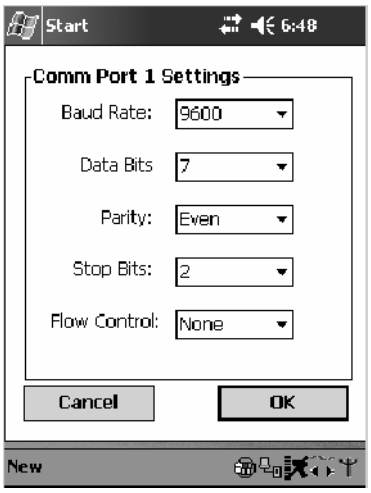
Tethered Scanner

Default settings for the Tethered Scanner are shown in this illustration:



Sabre 1551E or 1553 Tethered Scanner

The default communication configuration for the Sabre 1551E or 1553 Tethered Scanner is shown in the following illustration. Scan the EasySet Reset Factory Defaults label to set the Sabre 1551E or 1553 tethered scanner communications settings to this configuration. The COM1 port configuration settings must also match those of the scanner to scan labels.



Welch Allyn 1470 Imager Settings

You can set the Welch Allyn 1470 Imager to this configuration by scanning the Factory Default Settings label.

Scanner Cabling

A null modem cable is required for the Welch Allyn 1470 Imager to communicate with the 700 Series Computer when using the 700 Series Serial Cable (P/N: 226-999-001).

Sabre 1551E / 1553 Cables connect directly to the Model 700 COM Port.

Limitations and Capabilities

The Tethered Scanner has the following limitations:

- No auto detection of a scanner's physical connection to COM1 port. User needs to ensure the communication settings of COM1 port matched the settings of the device.
- The Pocket PC Pocket Office applications misbehave when control characters such as carriage return are wedged. This is a known Pocket PC problem, which is being worked with Microsoft and for which a work around is being developed.
- Communications port is COM1 and cannot be changed.
- A complete bar code label is detected when the time between bytes (the inter-byte gap) exceeds 100 ms. This allows that data could be concatenated if two labels were received while the Comm Port Wedge or the 1551/1553 Tethered Scanner was not performing a read. That is, it could be wedging data just read or the read thread could be preempted. Also, the labels could appear concatenated if the scanner itself were to buffer the labels before transmitting them.

When enabled, the "Comm Port Wedge" menu option has this limitation:

- There is no bar code API to get bar code data from the bar code scanner. The Comm Port Wedge transmits the data through the keyboard interface only.

When enabled, the "1551/1553" menu option has these capabilities:

- Grid Data Editing is available.
- The source of the symbology configurations is only available via the Easy Set command labels. You can only configure the Virtual Wedge configurations via the Data Collection control panel applet Virtual Wedge page. See Appendix A, "*Configurable Settings*," for information.
- May transmit the data through the keyboard interface (via the Virtual Wedge).

- The bar code APIs, defined in the IADC interface, are available to get bar code data from the bar code scanner. The following example shows how to programmatically collects bar code data:

```
#include “IADC.h”                    // Linked with ITCUUID.LIB
#include “ITCAdcMgmt.h”              // Linked with ITCAdcDevMgmt.lib

  IADC* pIADC;
  HRESULT hrStatus = S_OK;

// Create a ADC COM interface to collect bar code data from the 1551E/1553
// when the 1551/1553 menu option is enabled.
  hrStatus =
  ITCDeviceOpen(TEXT(“ExtScanner”), // Name of the ADC device.
    IID_IADC,                       // COM interface to return
    ITC_DHDEVFLAG_READAHEAD,        // Device’s Flags
    (LPVOID *) &pIADC);             // the returned interface

if( SUCCEEDED(hrStatus) )
  {
    BYTE byteBuffer[MAX_LABEL_SIZE];
    DWORD dwLength = 0;
  HRESULT hr = pIDC->Read(
    byteBuffer,                     // Buffer to put the ADC data.
    MAX_LABEL_SIZE,                 // Size of pDataBuffer in bytes.
    &dwLength,                      // Number bytes returned.
    NULL,                           // Time stamp of the received data. NULL.
    INFINITE                        // Number of milliseconds to wait.
  );

}
    when done using this COM interface, delete it:
ITCDeviceClose( (IUnknown **) pIADC);
```

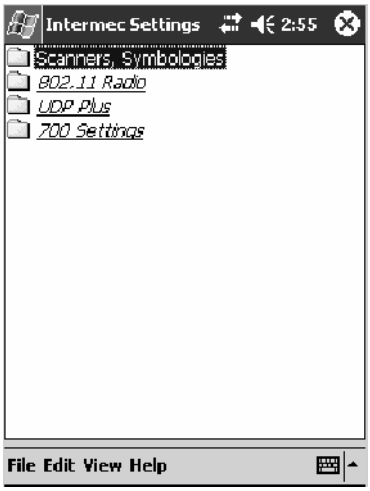
For Units With PSM Build 3.00 or Newer

Configuring the Tethered Scanner

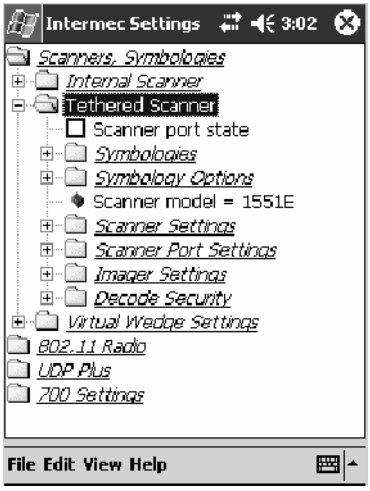
Do the following before you configure your tethered scanner from the Intermec Settings control panel applet. Information about the settings you can configure with this applet is described in the *Intermec Computer Command Reference Manua*l. The online manual is available from the Intermec web site at www.intermec.com.

1 Connect your tethered scanner to the tethered scanner port.

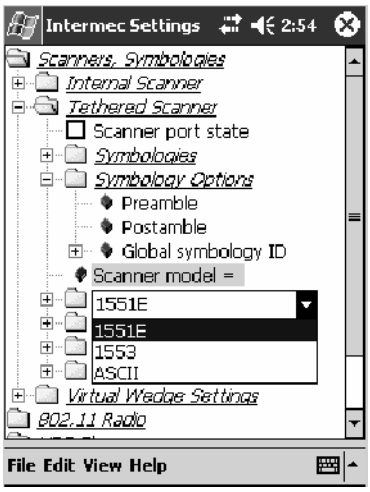
2 From the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > the **Intermec Settings** icon.



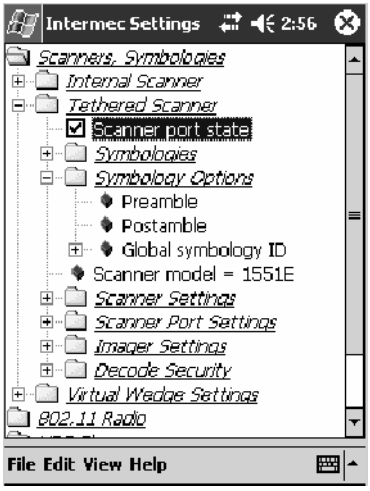
3 Double-tap the **Scanners, Symbologies** folder, then expand (+) the **Tethered Scanner** option.



4 Tap the **Scanner model** option for a drop-down list, then select the applicable scanner, such as “1551E” or “1553” in this sample screen.



5 Make sure a scanner is connected to your 700 Computer properly. Then, tap to check **Scanner Port State**, then tap **File** > **Save Settings** from the bottom of the screen. These changes will take several moments to reset.



1551E or 1553 Selected for Scanner Model

When "1551E" or "1553" is selected from the **Scanner model** option *(see step 4 above)*, and the port state is already enabled *(see step 5)*, the process will take several moments to reset. When 1551E or 1553 is successfully connected during this step, the unit will emit some beeps. Here, the terminal is initializing the scanner at 9600 for the baud rate, 7 data bits, even parity, and 2 stop bits and synchronizing the terminal's configuration with the attached scanner.

With "1551E" or "1553" selected, Symbologies, Symbology Options, Hardware Trigger, and Scanner Port settings are configured from the Intermec Settings applet. See the the *Intermec Computer Command Reference Manual*, available from the Intermec web site at www.intermec.com, for more information about each enabled option.

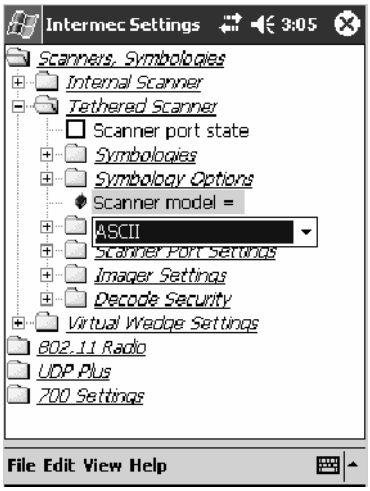
ASCII Selected for Scanner Model

To send data coming into the 700 Series Computer through the COM1 port from an external input device, as keyboard data to an application on the desktop, do the following:

1 Select "ASCII" from the **Scanner model** option.

2 Tap to check **Scanner Port State**.

3 Tap **File** > **Save Settings** from the bottom of the screen.



With "ASCII" selected, Symbology Options, Hardware Trigger, and Scanner Port settings are configured from the Intermec Settings applet. See the the *Intermec Computer Command Reference Manual*, available from the Intermec web site at www.intermec.com, for more information about each enabled option.

Note: When selecting either the 1551E or the 1553 Scanner or enabling the scanner port for these scanners, the 700 Computer tries to communicate with the attached scanner. If the scanner is not powered, if the cable is not connected properly, the wrong cable is used, or if the scanner firmware is older than 2.0, and the "Failed to save one or more settings" message appears, then this step failed.

This process can take time as the terminal is going through a group of RS-232 settings to communicate with the scanner. After successful communicated with the scanner (about eight beeps are generated), it initializes the scanner with the 700 Computer's current settings. This process might generate a series of beeps pending on the firmware version installed in the scanner. These beeps are suppressed in firmware versions 2.08 or greater.

Troubleshooting the 1551E/1553 Tethered Scanner

Do the following to troubleshoot your 1551/1553 Tethered Scanner:

1. Ensure the correct cable is used for the scanner on the tethered scanner port. *Note the 700 Computer cannot supply power to the scanner.*
2. Perform a quick test to determine whether the connection is good. Temporary select the scanner model as "ASCII," then enable the scanner port state. Go to a command prompt or a notepad and scan a data label. If a label is wedged into the command prompt or notepad, then the connection is good.
3. If step 2 passes, reset the scanner configurations to their defaults (scan the Reset Factory Defaults label on the next page) to prevent miscommunication, then reenable the scanner port state.
4. If step 2 fails, then the firmware installed in the tethered scanner may be older than version 2.0. Upgrade your scanner firmware.

Reset Factory Defaults

Scan the EasySet software bar code label "Reset Factory Default" to restore all of your scanner's configurations to their factory defaults. When this command label is scanned, reinitialize the tethered scanner (such as disable the scanner port state, then enable it) on the 700 Computer. Otherwise, the online configuration and scanning on the 700 Computer are not functional. In general, scan this label only to initially reset the scanner.

Do not scan EasySet command labels to change the following settings:

- Symbologies code mark
- Code 128, EAN29 Identifier
- Preamble and Postamble
- Enable/Disable symbologies
- Symbology ID transmit option

In some cases, scanning EasySet Command labels cause the current setting on the user interface to be out of sync with the scanner settings. However, in some cases, scanning these labels does corrupt scanned data.

The "Open COMx error: 0x00000037" message appears if the COM port cannot open due to another application using the port. Disable that application to free up the COM1 port before you can enable the scanner. "x" is the COM port number, such as 1, 2, or 3.

Tethered Scanner Supported Symbologies

The user interface may allow configuration of PDF417, Micro PDF417, RSS, and Codablock bar code symbologies. However, these symbologies are dependant on what scanner models and firmware versions are in use. See the following table for a guideline and Appendix B, "*Bar Code Symbologies*" for more information on each supported symbology:

You can use a generic ASCII scanner with the 700 Computer. Pending on the scanner, linear symbologies such as Code39, should decode correctly. However, 2D symbologies such as PDF417 may not decode correctly.

Symbologies	1551E	1553
Code39	X	X
UPC/EAN	X	X
Code 128	X	X
Interleaved 2 of 5	X	X
Code 93	X	X
Codabar	X	X
Code 2 of 5	X	X
MSI	X	X
Plessey	X	X
Code 11	X	X
Matrix 2 of 5	X	X
Telepen	X	X
PDF417	Available in 1551 0808 PDF	
Micro PDF417	Available in 1551 0808 PDF, Sxxp217_ or later	
MaxiCode		
Data Matrix		
QR Code		
RSS 14	F/w version 2.15 or later	F/w version 2.15 or later
RSS Limited	F/w version 2.15 or later	F/w version 2.15 or later
RSS Expanded	F/w version 2.15 or later	F/w version 2.15 or later
Codablock A	Available in 1551 0808 PDF	
Codablock F	Available in 1551 0808 PDF	
UCC Composite		

7 Programming

The following programming information pertains to the 700 Series Color Mobile Computer:

- Creating CAB Files *(page 220)*
- Customization and Lockdown *(page 237)*
- FTP Server *(page 239)*
- Kernel I/O Control Functions *(page 247)*
- Network Selection APIs *(page 263)*
- Notifications *(page 286)*
- Reboot Functions *(page 288)*
- Remapping the Keypad *(page 289)*

Creating CAB Files

The Windows CE operating system uses a .CAB file to install an application on a Windows CE-based device. A .CAB file is composed of multiple files that are compressed into one file. Compressing multiple files into one file provides the following benefits:

- All application files are present.
- A partial installation is prevented.
- The application can be installed from several sources, such as a desktop computer or a Web site.

Use the CAB Wizard application (CABWIZ.EXE) to generate a .CAB file for your application.

Creating Device-Specific CAB Files

Do the following to create a device-specific .CAB file for an application, *in the order provided*:

1. Create an .INF file with Windows CE-specific modifications *(page 220)*.
2. *Optional* Create a SETUP.DLL file to provide custom control of the installation process *(page 232)*.
3. Use the CAB Wizard to create the .CAB file, using the .INF file, the optional SETUP.DLL file, and the device-specific application files as parameters *(page 235)*.

Creating an .INF File

An .INF file specifies information about an application for the CAB Wizard. Below are the sections of an .INF file:

[Version]

This specifies the creator of the file, version, and other relevant information.

Required? Yes

- **Signature**: "*signature_name*"
  "$Windows NT$"
- **Provider**: "*INF_creator*"
  The company name of the application, such as "Microsoft."
- **CESignature**
  "$Windows CE$"

Example

```
[Version]
Signature = "$Windows NT$"
Provider = "Intermec"
CESignature = "$Windows CE$"
```

[CEStrings]

This specifies string substitutions for the application name and the default installation directory.

Required? Yes

- **AppName**: *app_name*
  Name of the application. Other instances of %AppName% in the .INF file are replaced with this string value, such as RP32.
- **InstallDir**: *default_install_dir*
  Default installation directory on the device. Other instances of %InstallDir% in the .INF file are replaced with this string value. Example: \SDMMC_Disk\%AppName%

Example

```
[CEStrings]
AppName="Game Pack"
InstallDir=%CE1%\%AppName%
```

[Strings]

This section is optional and defines one or more string keys. A string key represents a string of printable characters.

Required? No

- **string_key**: *value*
  String consisting of letters, digits, or other printable characters. Enclose *value* in double quotation marks """" if the corresponding string key is used in an item that requires double quotation marks. No string_keys is okay.

Example

```
[Strings]
reg_path = Software\Intermec\My Test App
```

[CEDevice]

Describes the platform for the targeted application. All keys in this section are optional. If a key is nonexistent or has no data, Windows CE does not perform any checking with the exception being *UnsupportedPlatforms*. If the *UnsupportedPlatforms* key exists but no data, the previous value is not overridden.

Required? Yes

- **ProcessorType** : *processor_type*
  The value that is returned by **SYSTEMINFO**.dwProcessorType.For example, the value for the ARM CPU is 2577
- **UnsupportedPlatforms**: *platform_family_name*
  This lists known unsupported platform family names. If the name specified in the [**CEDevice.xxx**] section is different from that in the [**CEDevice**] section, both *platform_family_name* values are unsupported for the microprocessor specified by xxx. That is, the list of unsupported platform family names is appended to the previous list of unsupported names. Application Manager will not display the application for an unsupported platform. Also, a user will be warned during the setup process if the .CAB file is copied to an unsupported device.

Example

```
[CEDevice]
UnsupportedPlatforms = pltfrm1 ; pltfrm1 is unsupported
[CEDevice.SH3]
UnsupportedPlatforms = ; pltfrm1 is still unsupported
```

- **VersionMin**: *minor_version*
  Numeric value returned by **OSVERSIONINFO**.dwVersionMinor. The .CAB file is valid for the currently connected device if the version of this device is greater than or equal to **VersionMin**.
- **VersionMax**: *major_version*
  Numeric value returned by **OSVERSIONINFO**.dwVersionMajor. The .CAB file is valid for the currently connected device if the version of this device is less than or equal to **VersionMax**.
- **BuildMin**: *build_number*
  Numeric value returned by **OSVERSIONINFO**.dwBuildNumber. The .CAB file is valid for the currently connected device if the version of this device is greater than or equal to **BuildMin**.
- **BuildMax**: *build_number*
  Numeric value returned by **OSVERSIONINFO**.dwBuildNumber. The .CAB file is valid for the currently connected device if the version of this device is less than or equal to **BuildMax**.

Example

The following code example shows three [**CEDevice**] sections: one that gives basic information for any CPU and two that are specific to the SH3 and the MIPS microprocessors.

```
[CEDevice]                          ; A "template" for all platforms
UnsupportedPlatforms = pltfrm1      ; Does not support pltfrm1

; The following specifies version 1.0 devices only.
VersionMin = 1.0
VersionMax = 1.0

[CEDevice.ARM]                  ; Inherits all [CEDevice] settings
; This will create a .CAB file specific to ARM devices.
ProcessorType = 2577            ; ARM .cab file is valid for ARM microprocessors.
UnsupportedPlatforms =          ; pltfrm1 is still unsupported

; The following overrides the version settings so that no version checking is
performed.
VersionMin =
VersionMax =

[CEDevice.MIPS]                     ; Inherits all [CEDevice] settings
; This will create a .CAB file specific to "MIPS" devices.
ProcessorType = 4000            ; MIPS .CAB file is valid for MIPS
microprocessor.
UnsupportedPlatforms =pltfrm2 ; pltfrm1, pltfrm2 unsupported for MIPs .CAB
file.
```

Note: To create the two CPU-specific .CAB files for the SETUP.INF file in the previous example, run the CAB Wizard with the "/cpu arm mips" parameter.

[DefaultInstall]

This describes the default installation of your application. Note that under this section, you will list items expanded upon later in this description.

Required? Yes

- **Copyfiles**: *copyfile_list_section*
  Maps to files defined later in the .INF file, such as Files.App, Files.Font, and Files.Bitmaps.
- **AddReg**: *add_registry_section*
  Example: RegSettings.All
- **CEShortcuts**: *shortcut_list_section*
  String that identifies one more section that defines shortcuts to a file, as defined in the [**CEShortcuts**] section.
- **CESetupDLL**: *setup_DLL*
  Optimal string that specifies a SETUP.DLL file. It is written by the Independent Software Vendor (ISV) and contains customized functions for operations during installation and removal of the application. The file must be specified in the [**SourceDisksFiles**] section.
- **CESelfRegister**: *self_reg_DLL_filename*
  String that identifies files that self-register by exporting the **DllRegisterServer** and **DllUnregisterServer** Component Object Model (COM) functions. Specify these files in the [**SourceDiskFiles**] section. During installation, if installation on the device fails to call the file's exported **DllRegisterServer** function, the file's exported **DllUnregisterServer** function will not be called during removal.

Example

```
[DefaultInstall]
AddReg = RegSettings.All
CEShortcuts = Shortcuts.All
```

[SourceDiskNames]

This section describes the name and path of the disk on which your application resides.

Required? Yes

- **disk_ordinal**: *disk_label,,path*
  1=,"App files" , C:\Appsoft\RP32\...
  2=,"Font files",,C:\RpTools\...
  3=,"CE Tools" ,,C:\windows ce tools...
- **CESignature**: "$Windows CE$"

Example

```
[SourceDisksNames]                          ; Required section
1 = ,"Common files",,C:\app\common          ; Using an absolute path
[SourceDisksNames.SH3]
2 = ,"SH3 files",,sh3                       ; Using a relative path
[SourceDisksNames.MIPS]
2 = ,"MIPS files",,mips                     ; Using a relative path
```

[SourceDiskFiles]

This describes the name and path of the files in which your application resides.

Required? Yes

- **filename**: *disk_number[,subdir]*
  RPM.EXE = 1,c:\appsoft\...
  WCESTART.INI = 1
  RPMCE212.INI = 1
  TAHOMA.TTF = 2

Note: [**,subdir**] is relative to the location of the INF file.

Example

```
[SourceDisksFiles]      ; Required section
begin.wav = 1
end.wav = 1
sample.hlp = 1
[SourceDisksFiles.SH3]
sample.exe = 2        ; Uses the SourceDisksNames.SH3 identification of 2.
[SourceDisksFiles.MIPS]
sample.exe = 2        ; Uses the SourceDisksNames.MIPS identification of 2.
```

[DestinationDirs]

This describes the names and paths of the destination directories for the application on the target device. *Note Windows CE does not support directory identifiers.*

Required? Yes

- **file_list_section**: *0,subdir*
  String that identifies the destination directory. The following list shows the string substitutions supported by Windows CE. Use these only for the beginning of the path. \
  %CE1% \Program Files
  %CE2% \Windows
  %CE3% \My Documents
  %CE4% \Windows\Startup
  %CE5% \My Documents
  %CE6% \Program Files\Accessories
  %CE7% \Program Files\Communication
  %CE8% \Program Files\Games
  %CE9% \Program Files\Pocket Outlook
  %CE10% \Program Files\Office
  %CE11% \Windows\Start Menu\Programs
  %CE12% \Windows\Start Menu\Programs\Accessories
  %CE13% \Windows\Start Menu\Programs\Communications
  %CE14% \Windows\Start Menu\Programs\Games
  %CE15% \Windows\Fonts
  %CE16% \Windows\Recent
  %CE17% \Windows\Start Menu
  %InstallDir%
  Contains the path to the target directory selected during installation. It is declared in the [**CEStrings**] section
  %AppName%
  Contains the application name defined in the [**CEStrings**] section.

Example

```
[DestinationDirs]
Files.Common  = 0,%CE1%\My Subdir      ; \Program Files\My Subdir
Files.Shared  = 0,%CE2%                ; \Windows
```

[CopyFiles]

This section, under the [**DefaultInstall**] section, describes the default files to copy to the target device. Within the [**DefaultInstall**] section, files were listed that must be defined elsewhere in the INF file. This section identifies that mapping and may contain flags.

Required? Yes

- **copyfile_list_section**: *destination_filename,[source_filename]*
  The *source_filename* parameter is optional if it is the same as *destination_filename*.
- **copyfile_list_section**: *flags*
  The numeric value that specifies an action to be done while copying files. The following table shows values supported by Windows CE.

Flag	Value	Description
COPYFLG_WARN_IF_SKIP	0x00000001	Warn user if skipping a file is attempted after error.
COPYFLG_NOSKIP	0x00000002	Do not allow a user to skip copying a file.
COPYFLG_NO_OVERWRITE	0x00000010	Do not overwrite files in destination directory.
COPYFLG_REPLACEONLY	0x00000400	Copy the source file to the destination directory only if the file is already in the destination directory.
CE_COPYFLG_NO_DATE_DIALOG	0x20000000	Do not copy files if the target file is newer.
CE_COPYFLG_NODATECHECK	0x40000000	Ignore date while overwriting the target file.
CE_COPYFLG_SHARED	0x80000000	Create a reference when a shared DLL is counted.

Example

```
[DefaultInstall.SH3]
CopyFiles = Files.Common, Files.SH3
[DefaultInstall.MIPS]
CopyFiles = Files.Common, Files.MIPS
```

[AddReg]

This section, under the [**DefaultInstall**] section, is optional and describes the keys and values that the .CAB file adds to the device registry. Within the [**DefaultInstall**] section, a reference may have been made to this section, such as "AddReg=RegSettings.All". This section defines the options for that setting.

Required? No

- **add_registry_section**: *registry_root_string*
  String that specifies the registry root location. The following list shows the values supported by Windows CE.
  - HKCR Same as HKEY_CLASSES_ROOT
  - HKCU Same as HKEY_CURRENT_USER
  - HKLM Same as HKEY_LOCAL_MACHINE
- **add_registry_section**: *value_name*
  Registry value name. If empty, the "default" registry value name is used.
- **add_registry_section**: *flags*
  Numeric value that specifies information about the registry key. The following table shows the values that are supported by Window CE.

Flag	Value	Description
FLG_ADDREG_NOCLOBBER	0x00000002	If the registry key exists, do not overwrite it. Can be used with any of the other flags in this table.
FLG_ADDREG_TYPE_SZ	0x00000000	REG_SZ registry data type.
FLG_ADDREG_TYPE_MULTI_SZ	0x00010000	REG_MULTI_SZ registry data type. Value field that follows can be a list of strings separated by commas.
FLG_ADDREG_TYPE_BINARY	0x00000001	REG_BINARY registry data type. Value field that follows must be a list of numeric values separated by commas, one byte per field, and must not use the 0x hexadecimal prefix.
FLG_ADDREG_TYPE_DWORD	0x00010001	REG_DWORD data type. The noncompatible format in the Win32 Setup .INF documentation is supported.

Example

```
AddReg = RegSettings.All

[RegSettings.All]
HKLM,%reg_path%,,0x00000000,alpha            ; <default> = "alpha"
HKLM,%reg_path%,test,0x00010001,3            ; Test = 3
HKLM,%reg_path%\new,another,0x00010001,6     ; New\another = 6
```

[CEShortCuts]

This section, a Windows CE-specific section under the [**DefaultInstall**] section, is optional and describes the shortcuts that the installation application creates on the device. Within the [**DefaultInstall**] section, a reference may have been made to this section, such as "ShortCuts.All". This section defines the options for that setting.

Required? No

- **shortcut_list_section**: *shortcut_filename*
  String that identifies the shortcut name. It does not require the .LNK extension.
- **shortcut_list_section**: *shortcut_type_flag*
  Numeric value. Zero or empty represents a shortcut to a file; any non-zero numeric value represents a shortcut to a folder.
- **shortcut_list_section**: *target_file_path*
  String value that specifies the destination location. Use the target file name for a file, such as MyApp.exe, that must be defined in a file copy list. For a path, use a *file_list_section* name defined in the [**DestinationDirs**] section, such as *DefaultDestDir*, or the *%InstallDir%* string.
- **shortcut_list_section**: *standard_destination_path*
  Optional string value. A standard *%CEx%* path or *%InstallDir%*. If no value is specified, the *shortcut_list_section* name of the current section or the *DefaultDestDir* value from the [**DestinationDirs**] section is used.

Example

```
CEShortcuts = Shortcuts.All
[Shortcuts.All]
Sample App,0,sample.exe          ; Uses the path in DestinationDirs. Sample
App,0,sample.exe,%InstallDir%    ; The path is explicitly specified.
```

Sample .INF File

```
[Version]           ; Required section
Signature = "$Windows NT$"
Provider = "Intermec Technologies Corporation"
CESignature = "$Windows CE$"

;[CEDevice]
;ProcessorType =

[DefaultInstall]    ; Required section
CopyFiles = Files.App, Files.Fonts, Files.BitMaps, Files.Intl,
Files.TelecomNcsCE, Files.Windows, Files.Import, Files.Export, Files.Work,
Files.Database, Files.WinCE AddReg = RegSettings.All ;CEShortcuts =
Shortcuts.All

[SourceDisksNames]  ; Required section
1 = ,"App files" ,,c:\appsoft\...
2 = ,"Font files" ,,c:\WinNT\Fonts
3 = ,"CE Tools"  ,,c:\windows ce tools\wce400\700ie\mfc\lib\x86

[SourceDisksFiles]  ; Required section
rpm.exe   = 1,C:\Appsoft\program\wce400\WCEX86Rel700
wcestart.ini = 1
```

```
rpmce212.ini = 1
intermec.bmp = 1
rpmlogo.bmp = 1
rpmname.bmp = 1
import.bmp  = 1
export.bmp  = 1
clock.bmp  = 1
printer.bmp = 1
filecopy.bmp = 1
readme.txt  = 1
lang_eng.bin = 1
rpmdata.dbd = 1,database\wce1
tahoma.ttf  = 2
mfcce212.dll = 3
olece212.dll = 3
olece211.dll = 1,c:\windows ce tools\wce400\NMSD61102.11\mfc\lib\x86
rdm45wce.dll = 1,c:\rptools\rdm45wce\4_50\lib\wce400\wcex86rel
picfmt.dll  = 1,c:\rptools\picfmt\1_00\wce400\wcex86rel6110
fmtctrl.dll = 1,c:\rptools\fmtctrl\1_00\wce400\wcex86rel6110
ugrid.dll  = 1,c:\rptools\ugrid\1_00\wce400\wcex86rel6110
simple.dll  = 1,c:\rptools\pspbm0c\1_00\wce400\wcex86rel
psink.dll = 1,c:\rptools\psink\1_00\wce400\WCEX86RelMinDependency
pslpwce.dll =1,c:\rptools\pslpm0c\1_00\wce400\WCEX86RelMinDependency
npcpport.dll = 1,c:\rptools\cedk\212_03\installable drivers\printer\npcp
;dexcom.dll  = 1,c:\rptools\psdxm0c\1_00\x86
ncsce.exe  = 1,c:\rptools\ncsce\1_04
nrinet.dll  = 1,c:\rptools\ncsce\1_04

[DestinationDirs] ; Required section
;Shortcuts.All = 0,%CE3%  ; \Windows\Desktop
Files.App         = 0,%InstallDir%
Files.DataBase      = 0,%InstallDir%\DataBase
Files.BitMaps     = 0,%InstallDir%\Bitmaps
Files.Fonts       = 0,%InstallDir%\Fonts
Files.Intl        = 0,%InstallDir%\Intl
Files.TelecomNcsCE  = 0,%InstallDir%\Telecom\NcsCE
Files.Windows     = 0,%InstallDir%\Windows
Files.Import      = 0,%InstallDir%\Import
Files.Export      = 0,%InstallDir%\Export
Files.Work        = 0,%InstallDir%\Work
Files.WinCE       = 0,\storage_card\wince

[CEStrings]       ; Required section
AppName = Rp32
InstallDir = \storage_card\%AppName%

[Strings]         ; Optional section
;[Shortcuts.All]
;Sample App,0,sample.exe               ; Uses the path in DestinationDirs.
;Sample App,0,sample.exe,%InstallDir%     ; The path is explicitly specified.

[Files.App]
rpm.exe,,,0
rpm.ini,rpmce212.ini,,0
mfcce212.dll,,,0
olece212.dll,,,0
olece211.dll,,,0
rdm45wce.dll,,,0
picfmt.dll,,,0
```

```
fmtctrl.dll,,,0
ugrid.dll,,,0
simple.dll,,,0
psink.dll,,,0
pslpwce.dll,,,0
npcpport.dll,,,0
;dexcom.dll,,,0

[Files.DataBase]
rpmdata.dbd,,,0

[Files.Fonts]
tahoma.ttf,,,0

[Files.BitMaps]
intermec.bmp,,,0
rpmlogo.bmp,,,0
rpmname.bmp,,,0
import.bmp,,,0
export.bmp,,,0
clock.bmp,,,0
printer.bmp,,,0
filecopy.bmp,,,0

[Files.Intl]
lang_eng.bin,,,0

[Files.TelecomNcsCE]
ncsce.exe,,,0
nrinet.dll,,,0

[Files.Windows]
readme.txt,,,0

[Files.Import]
readme.txt,,,0

[Files.Export]
readme.txt,,,0

[Files.Work]
readme.txt,,,0

[Files.WinCE]
wcestart.ini,,,0

[RegSettings.All]
HKLM,”SOFTWARE\Microsoft\Shell\AutoHide”,,0x00010001,1
; Autohide the taskbar HKLM,”SOFTWARE\Microsoft\Shell\OnTop”,,0x00010001,0
  ; Shell is not on top
HKLM,”SOFTWARE\Microsoft\Clock”,SHOW_CLOCK,0x00010001,0
  ; Clock is not on taskbar
```

Using Installation Functions in SETUP.DLL

SETUP.DLL is an optional file that enables you to perform custom operations during installation and removal of your application. The following list shows the functions that are exported by SETUP.DLL.

Install_Init	Called before installation begins. Use this function to check the application version when reinstalling an application and to determine if a dependent application is present.
Install_Exit	Called after installation is complete. Use this function to handle errors that occur during application installation.
Uninstall_Init	Called before the removal process begins. Use this function to close the application, if the application is running.
Uninstall_Exit	Called after the removal process is complete. Use this function to save database information to a file and delete the database and to tell the user where the user data files are stored and how to reinstall the application.

Note; Use [**DefaultInstall**] > **CESelfRegister** (page 224) in the .INF file to point to SETUP.DLL.

After the CAB File Extraction

Cab files that need to cause a warm reset after cab extraction will need to create the __RESETMEPLEASE__.TXT file in the "\Windows" directory. The preferred method to create this file is within the DllMain portion of the SETUP.DLL file. It looks like this:

```
#include <windows.h>
#include <Tlhelp32.h>
#include <winioctl.h>
#include <ce_setup.h>   // in the public SDK dir

#define IOCTL_TERMINAL_RESET CTL_CODE (FILE_DEVICE_UNKNOWN,FILE_ANY_ACCESS,
2050, METHOD_NEITHER)

BOOL APIENTRY DllMain( HANDLE h, DWORD reason, LPVOID lpReserved )
{
  return TRUE;
} // DllMain


//*********************************************************************
// $DOCBEGIN$
// BOOL IsProcessRunning( TCHAR * pname );
//
// Description: Get process table snapshot, look for pname running.
//
// Arguments: pname - pointer to name of program to look for.
// for example, app.exe.
//
// Returns: TRUE - process is running.
//          FALSE - process is not running.
// $DOCEND$
//*********************************************************************
BOOL IsProcessRunning( TCHAR * pname )
{
  HANDLE hProcList;
```

```
  PROCESSENTRY32 peProcess;
  DWORD thDeviceProcessID;
  TCHAR lpname[MAX_PATH];

  if ( !pname || !*pname ) return FALSE;

  _tcscpy( lpname, pname );
  _tcslwr( lpname );

  hProcList = CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS, 0 );

  if ( hProcList == INVALID_HANDLE_VALUE ) {
    return FALSE;
  } // end if

  memset( &peProcess, 0, sizeof(peProcess) );
  peProcess.dwSize = sizeof(peProcess);

  if ( !Process32First( hProcList, &peProcess ) ) {
    CloseToolhelp32Snapshot( hProcList );
    return FALSE;
  } // end if

  thDeviceProcessID = 0;

  do {
    _tcslwr( peProcess.szExeFile );

    if ( _tcsstr( peProcess.szExeFile, lpname ) ) {
      thDeviceProcessID = peProcess.th32ProcessID;
      break;
    } // end if
  } while ( Process32Next( hProcList, &peProcess ) );

  if ( ( GetLastError() == ERROR_NO_MORE_FILES ) && ( thDeviceProcessID == 0
) ) {
    CloseToolhelp32Snapshot( hProcList );
    return FALSE;
  } // end if

  CloseToolhelp32Snapshot( hProcList );
  return TRUE;
} // IsProcessRunning

codeINSTALL_INIT Install_Init(
  HWND hwndParent,
  BOOL fFirstCall,
  BOOL fPreviouslyInstalled,
  LPCTSTR pszInstallDir )
{
  return codeINSTALL_INIT_CONTINUE;
}

codeINSTALL_EXIT Install_Exit (
  HWND hwndParent,
  LPCTSTR pszInstallDir,
  WORD cFailedDirs,
  WORD cFailedFiles,
  WORD cFailedRegKeys,
```

```
  WORD cFailedRegVals,
  WORD cFailedShortcuts )
{
  HANDLE h;
  TCHAR srcfile[MAX_PATH];
  TCHAR dstfile[MAX_PATH];

  if (cFailedDirs || cFailedFiles || cFailedRegKeys ||
    cFailedRegVals || cFailedShortcuts)
    return codeINSTALL_EXIT_UNINSTALL;

  if ( IsProcessRunning( L"autocab.exe" ) )
  {
    h = CreateFile( L"\\Windows\\__resetmeplease__.txt",
      (GENERIC_READ | GENERIC_WRITE), 0, NULL, CREATE_ALWAYS,
      FILE_ATTRIBUTE_HIDDEN, NULL );

    if ( h != INVALID_HANDLE_VALUE )
      CloseHandle( h );
    else
    {
      // Couldn't create the file.  If it failed because the file already
exists, it is not fatal.
      // Otherwise, notify user of the inability to reset the device and they
will have to
      // perform it manually after all of the installations are complete.
    } // end if
  }
  else
  {
    DWORD dret;

    h = CreateFile( L"SYI1:",
      (GENERIC_WRITE | GENERIC_READ), 0, NULL, OPEN_EXISTING,
      FILE_ATTRIBUTE_NORMAL, NULL );

    // Force a warm start NOW.
    if ( h != INVALID_HANDLE_VALUE )
    {
      DeviceIoControl( h, IOCTL_TERMINAL_RESET, NULL, 0, NULL, 0, &dret,
NULL);

      // Won't return, but we'll show clean up anyway
      CloseHandle( h );
    }
    else
    {
      // Couldn't access SYSIO. Notify user.
    } // end if
  } // end if

  return codeINSTALL_EXIT_DONE;
}

codeUNINSTALL_INIT
Uninstall_Init(
  HWND hwndParent,
  LPCTSTR pszInstallDir ) {
```

```
  // TODO: Perform the reverse of INSTALL_INIT here
  return codeUNINSTALL_INIT_CONTINUE;
}

codeUNINSTALL_EXIT
Uninstall_Exit(HWND hwndParent) {

  // TODO: Perform the reverse of INSTALL_EXIT here
  return codeUNINSTALL_EXIT_DONE;
}
```

The system software looks for the following directory structure and files on the installed media card whether it be an SD card or CF card or embedded flash file system. No other folders need exist.

```
\2577\autorun.exe
\2577\autorun.dat
\2577\autocab.exe
\2577\autocab.dat
\cabfiles\*.cab
```

Creating CAB Files with CAB Wizard

After you create the .INF file and the optional SETUP.DLL file, use the CAB Wizard to create the .CAB file. The command-line syntax for the CAB Wizard is as follows:

```
cabwiz.exe "inf_file" [/dest dest_directory] [/err error_file] [/cpu cpu_type
[cpu_type]]
```

A batch file, located in <program> directory, with the following commands, works well:

```
cabwiz.exe c:\appsoft\<program>\<inf_file_name>
cd \appsoft\<program>
```

"inf_file"	The SETUP.INF file path.
dest_directory	The destination directory for the .CAB files. If no directory is specified, the .CAB files are created in the "inf_file" directory.
error_file	The file name for a log file that contains all warnings and errors that are encountered when the .CAB files are compiled. If no file name is specified, errors are displayed in message boxes. If a file name is used, the CAB Wizard runs without the user interface (UI); this is useful for automated builds.
cpu_type	Creates a .CAB file for each specified microprocessor tag, which is a label used in the Win32 SETUP.INF file to differentiate between different microprocessor types. The */cpu* parameter, followed by multiple *cpu_type* values, must be the last qualifier in the command line.

Example

This example creates .CAB files for the ARM and MIPS microprocessors, assuming the Win32 SETUP.INF file contains the ARM and MIPS tags:

```
cabwiz.exe "c:\myfile.inf" /err myfile.err /cpu arm mips
```

Note: CABWIZ.EXE, MAKECAB.EXE, and CABWIZ.DDF (Windows CE files available on the Windows CE Toolkit) must be installed in the same directory on the desktop computer. Call CABWIZ.EXE using its full path for the CAB Wizard application to run correctly.

Troubleshooting the CAB Wizard

To identify and avoid problems that might occur when using the CAB Wizard, follow these guidelines:

- Use %% for a percent sign (%) character when using this character in an .INF file string, as specified in Win32 documentation. This will not work under the [**Strings**] section.
- Do not use .INF or .CAB files created for Windows CE to install applications on Windows-based desktop platforms.
- Ensure the MAKECAB.EXE and CABWIZ.DDF files, included with Windows CE, are in the same directory as CABWIZ.EXE.
- Use the full path to call CABWIZ.EXE.
- Do not create a .CAB file with the MAKECAB.EXE file included with Windows CE. You must use CABWIZ.EXE, which uses MAKECAB.EXE to generate the .CAB files for Windows CE.
- Do *not* set the read-only attribute for .CAB files.

Customization and Lockdown

Pocket PC (Windows Mobile) is a hardware specification created by Microsoft Corporation. Devices that wish to carry the Pocket PC logo must meet the minimum hardware requirements set in the Pocket PC specification. Manufacturers are free to add extra hardware functionality.

Pocket PC devices also use a specialized version of the CE operating system. This OS is built from Windows CE 3.0 but contains customizations, most notably the lack of a desktop and the addition of the Today Screen.

To carry the Pocket PC logo, all devices must be tested at an Independent Test Laboratory. The ITL testing is done based on Microsoft requirements. The test lab then reports the findings back to Microsoft Corporation and Intermec Technologies. If the 700 Series Computer passed all tests, Intermec is allowed to ship the device with the Pocket PC logo. Each time the operating system is modified, Intermec must resubmit to ITL testing.

This means we cannot change the operating system much and still be a Pocket PC device. For example, if we remove Word from the Start menu, the device would fail ITL testing and we would not be able to ship devices with the Pocket PC logo.

Although many customers want a Pocket PC device, some customers would prefer that their users not have access to all of the Pocket PC features. Intermec cannot customize the operating system in any way but a custom application can:

- Delete items from the Start menu, and Programs folder. These items are just shortcuts in the file system so the application is not really being deleted. Cold booting the device will bring these items back so the application will need to be run on every cold boot.
- Use the RegFlushKey() API to save a copy of the registry to a storage device. See the 700 Color Management Tools portion of the *Intermec Developer's Library CD* for more information on how to do this. Saving a copy of the registry restores most system settings in a cold boot situation.
- Use the SHFullScreen() API in conjunction with other APIs to make the application take up the entire display and prevent the start menu from being available.
- Remap keys and disable keys on the keypad.
- Create a custom SIP.
- Make changes to the registry to configure the device.

Should you want your 700 Series Computer to display a full screen, keep in mind that your computer is Pocket-PC certified by Microsoft Corporation. Check out resources on programming for the Pocket PC, using the following links. These instructions give full instructions on how to display full screen.

- Instructions on how to create a full screen application for eVC++ applications using an SHFullScreen() API:
  http://support.microsoft.com/support/kb/articles/Q266/2/44.ASP
- Instructions on how to create a full screen application for eVB applications also using the SHFullScreen() API:
  http://support.microsoft.com/support/kb/articles/Q265/4/51.ASP

FTP Server

FTP support is provided through the FTP Server application FTPDCE.EXE (MS Windows CE Versions) which is provided as part the base system.

FTPDCE is the Internet File Transfer Protocol (FTP) server process. The server can be invoked from an application or command line. Besides servicing FTP client requests the FTP Server also send a "network announcement" to notify prospective clients of server availability.

Note: You should consult the RFC959 specification for proper use of some of these commands at the following URL:

- http://www.ietf.org/rfc/rfc959.txt for the text version, or
- http://www.w3.org/Protocols/rfc959/ for an html version

Do the following to send commands:

1. Start an FTP client and connect to the device FTP server.
2. Log in with "intermec" as the user name and "cr52401" for the password.
3. From the FTP client, send the command.
4. Wait for a response.

Synopsis

ftpdce [*options*]

Options

–Aaddr	(where *addr* is in the form of a.b.c.d) Sets the single target address to which to send the network announcement. *Default is broadcast.*
–Bbyte	Sets the FTP data block size. Smaller sizes may be useful over slower links. *Default is 65536.*
–Cname	Sets the device name. Used by Intermec management software.
–Fvalue	Disables the default Intermec account. A value of "0" disables the account. *Default is "1".* Note that disabling the default account without providing a working access control list on the server will result in a device that will not accept any FTP connections.
–Hsec	Sets the interval between network announcements in seconds.A value of "0" turns the network announcement off. *Default is 30 seconds.*
–Iaddr	(where *addr* is in the form of a.b.c.d) Sets the preferred 6920 Communications Server *(optional).*
–Llog	(where *log* is either "0" or "1") Sets the state of logging. *Default is 0 (disabled).*
–Nsec	Specifies the number of seconds to wait before initially starting FTP server services.
–Pport	Sets the UDP port on which the network announcement will be sent. *Default port is 52401.*
–Qport	Sets the port on which the FTP Server will listen for connections. *Default port is 21.*
–Rdir	Sets the FTP mount point to this directory. Default is the root folder of the object store.
–Tscrip	Sets the script name for the 6920 Communications Server to process.
–Uurl	Sets the default URL for this device.
–Z"parms"	Sets extended parameters to be included in the network announcement.

Configurable Parameters Via the Registry Editor

The following parameters receive default values during the installation of the Intermec FTP Server components. A few of the parameters are visible in the registry by default, but most must be created in order to modify the default behavior of the FTP server.

BlockSize

Setting this parameter configures the Intermec FTP Server to transmit and receive Ethernet packets using the specified data block size. By default, the FTP server transmits and receives data using a 64K data block size. Adjusting this value may be useful in certain wireless TCP/IP installations.

Key	`HKLM\Software\Intermec\IFTP`
Value Type	REG_DWORD - data block size, in bytes.
Valid Range	0x100-0x10000 (256-65536 decimal).
Default	65536

DeviceName

This parameter configures the Intermec FTP Server to include the specified device name in the Intermec Device Network Announcement (IDNA). Adjusting this value may be useful in assigning a symbolic name to this device for asset tracking.

Key	`HKLM\Software\Intermec\IFTP`
Value Type	REG_SZ
Valid Range	None.
Default	None.

DeviceURL

This parameter configures the Intermec FTP Server to transmit the specified URL in the IDNA. This can be used by Intermec management software for asset management.

Key	`HKLM\Software\Intermec\IFTP`
Value Type	REG_SZ
Valid Range	None.
Default	None.

IDNATarget

This parameter configures the Intermec FTP Server to transmit the IDNA to a specific destination instead of a general UDP broadcast. This parameter is useful on networks that do not allow UDP broadcasts to be routed between subnets. The use of this parameter will restrict the reception of the IDNA to the target destination only.

Key	`HKLM\Software\Intermec\IFTP`
Value Type	REG_SZ
Valid Range	None.
Default	None.

ManifestName

This parameter configures the Intermec FTP Server to transmit the specified manifest name in the IDNA. This parameter is used by the Intermec 6920 Communications Server for communication transactions. See the 6920 Communications Server documentation for proper use of this parameter.

Key	`HKLM\Software\Intermec\IFTP`
Value Type	REG_SZ
Valid Range	None.
Default	iftp.ini

PauseAtStartup

This parameter configures the Intermec FTP Server to sleep for the specified number of seconds before making the FTP service available on the device.

Key	`HKLM\Software\Intermec\IFTP`
Value Type	REG_DWORD - stored in seconds.
Valid Range	None.
Default	0

Root

This parameter configures the Intermec FTP Server to set the root of the FTP mount point to the specified value. *Note that this must map to an existing directory or you will not be able to log into the FTP Server.*

Key	`HKLM\Software\Intermec\IFTP`
Value Type	REG_SZ
Valid Range	None.
Default	\

Transferring Files Over TCP/IP Networks

The File Transfer Protocol (FTP) server transfers files over TCP/IP networks. The FTPDCE.EXE program is a version that does not display a window, but can run in the background.

FTPDCE is the Internet File Transfer Protocol (FTP) server process. The server can be invoked from an application or command line. Besides servicing FTP client requests, the FTP Server also sends a "network announcement" to notify prospective clients of server availability.

Remarks

The FTP Server currently supports the following FTP requests:

CDUP	Changes to the parent directory of the current working directory.
CWD	Changes working directory.
DELE	Deletes a file.
HELP	Gives help information.
LIST	(This FTP request is the same as the ls -lgA command). Gives list files in a directory.
MKD	Makes a directory.
MODE	*(Always Uses Binary).* Specifies data transfer mode.
NLST	*(Not supported)* Gives a name list of files in directory (this FTP request is the same as the *ls* command).
NOOP	Does nothing.
PASS	Specifies a password.
PWD	Prints the current working directory.
QUIT	Terminates session.
RETR	Retrieves a file.
RMD	Removes a directory.
RNFR	Specifies rename-from file name.
RNTO	Specifies rename-to file name.
STOR	Stores a file.
SYST	Shows the operating system type of server system.
TYPE	*(Binary transfers only.)* Specifies the data transfer type with the Type parameter.
USER	Specifies user name.
XCUP	*(Not Normally Used)* Changes the parent directory of the current working directory.
XCWD	*(Not Normally Used)* Changes the current directory.
XMKD	*(Not Normally Used)* Creates a directory.
XPWD	*(Not Normally Used)* Prints the current working directory.
XRMD	*(Not Normally Used)* Removes a directory.

SITE	The following extended OEM commands are supported by the SITE request. For Microsoft FTP clients, you can send site commands by preceding the command with "quote" such as "quote site status."	
	ATTRIB	Gets or sets the attributes of a given file. (SITE ATTRIB) Usage **QUOTE SITE ATTRIB** [*+R* | *-R*] [*+A* | *-A*] [*+S* | *-S*] [*+H* | *-H*] [[*path*] *filename*] + Sets an attribute. – Clears an attribute. *R* Read-only file attribute. *A* Archive file attribute. *S* System file attribute. *H* Hidden file attribute. To retrieve the attributes of a file, only specify the file. The server response will be: *200-AD SHRCEIX filename* If the flag exists in its position shown above, it is set. Also, in addition to the values defined above, there is also defined: *C* Compressed file attribute. *E* Encrypted file attribute. *I* INROM file attribute. *X* XIP file attribute (execute in ROM, not shadowed in RAM).
	BOOT	Reboots the server OS. This will cause the system on which the server is executing to reboot. The FTP Server will shut down cleanly before reboot. All client connections will be terminated. Cold boot is default except for the PocketPC build in which the default is warm boot. (SITE BOOT) Usage: **QUOTE SITE BOOT** [*WARM* | *COLD*]
	COPY	Copies a file from one location to another. (SITE COPY) Usage: **QUOTE SITE COPY** [*source*] [*destination*] Example: `QUOTE SITE COPY '\Storage Card\one.dat' '\Storage Card\two.dat'`
	EXIT	Exits the FTP Server. This command will shut down the FTP Server thus terminating all client connections. (SITE EXIT) Usage: **QUOTE SITE EXIT**
	HELP	Gives site command help information. (SITE HELP) Usage: **QUOTE SITE HELP** [*command*]
	KILL	Terminates a running program. (SITE KILL) Usage: **QUOTE SITE KILL** [*program* | *pid*]
	LOG	Opens or closes the program log. (SITE LOG) Usage: **QUOTE SITE LOG** [*open* [*filename*]| *close*]
	PLIST	Lists the running processes (SITE PLIST) Usage: **QUOTE SITE PLIST**
	RUN	Starts a program running. If the program to run has spaces in path or filename, wrapping the name with single quotes is required. Usage: **QUOTE SITE RUN** [*program*] Example: `QUOTE SITE RUN '\Storage Card\app.exe'`

STATUS	Returns the current settings of the FTP Server. MAC, serial number, model, IP address, network announcement information as well as OS memory usage are returned. (SITE STATUS) Usage: **QUOTE SITE STATUS**
TIMEOUT	Toggles idle timeout between 120 to 1200 seconds (2 to 20 minutes). If this timer expires with no activity between the client and the server, the client connection will be disconnected. If the optional seconds argument is supplied, the server will set the connection timeout to the number of seconds specified. *Default is 120 seconds or 2 minutes.* (SITE TIMEOUT) Usage: **QUOTE SITE TIMEOUT** [*seconds*]
EKEY	Gives site command electronic key information. (SITE HELP) Usage: **QUOTE SITE EKEY** [*command*]
EVAL	Gives site command electronic value information. (SITE HELP) Usage: **QUOTE SITE EVAL** [*command*]
GVAL	Gives site command general value information. (SITE HELP) Usage: **QUOTE SITE GVAL** [*command*]
PVAL	Gives site command value information. (SITE HELP) Usage: **QUOTE SITE PVAL** [*command*]

The remaining FTP requests specified in RFC 959 are recognized, but not implemented.

The banner returned in the parenthetical portion of its greeting shows the version number of the FTP Server as well as the MAC address, serial number and OS of the machine hosting the server.

The FTP Server supports browsing from the latest Netscape and Microsoft web browsers. Drag-and-drop capability is available using this environment.

The FTPDCMDS subdirectory contains commands to use from the web browser.

- Click EXITME.BIN to execute a SITE EXIT command.
- Click REBOOTME.BIN to execute SITE BOOT command.
- Use the GET command on these files to have the FTP Server execute these commands.
  - **Security**:
    A customer configurable access control list may be installed on the 700 Series Computer. This list will allow customers to restrict access via the FTP Server to users they wish and is in addition to default Intermec accounts that are disabled using the *-F0* option at runtime.

    The access control list is named FTPDCE.TXT and is placed in the same directory on the 700 Series Computer as the FTPDCE.EXE server. The FTP Server will encrypt this file to keep the information safe from unauthorized users. This file is encrypted when the FTP Server is started so a file that is placed onto the 700 Series Computer after the FTP Server starts will require a restart of the FTP Server to take effect.

The format of the FTPDCE.TXT is as follows:

```
FTPDCE:user1!passwd1<cr><lf>user2!passwd2<cr><lf>user3
!passwd3<cr><lf>...
```

Note: The user accounts and passwords are case sensitive.
Once the access control list is encrypted on the 700 Series Computer, the FTP Server hides this file from users. Once an access control list is installed on the 700 Series Computer, a new one is not accepted by the FTP Server until the previous one is removed. Encrypted access control lists are not portable between 700 Series Computers.

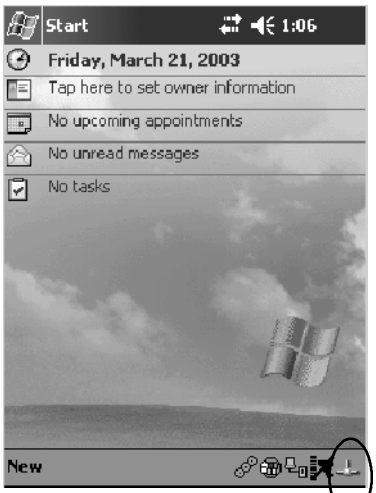
Stopping the FTP Server from Your Application

To allow application programmers the ability to programmatically shut down the FTP Server, the FTP Server periodically tests to see if a named event is signaled. The name for this event is "ITC_IFTP_STOP" (no quotes).

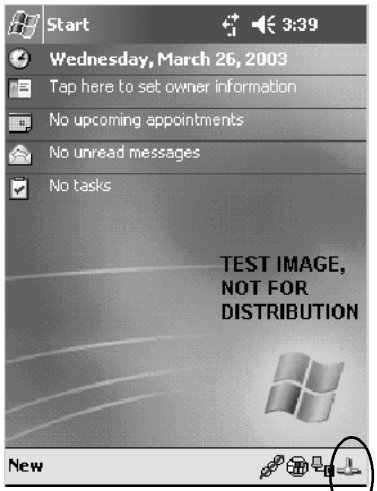
For examples on how to use events, consult the Microsoft Developer Network Library at http://www.msdn.com. The MSDN Library is an essential resource for developers using Microsoft tools, products, and technologies. It contains a bounty of technical programming information, including sample code, documentation, technical articles, and reference guides.

Autostart FTP

This automatically starts the FTP Server (FTPDCE.EXE) when the 700 Series Computer is powered on. This is provided with the NDISTRAY program (the Network Driver Interface Specification tray application), which displays the popup menu that currently allows you to load and unload the network drivers. Tap the antenna icon in the System Tray of the Today screen *(a sample antenna icon is circled below)* for this pop-up menu.



700 Color and 730 Screens



700CXL Screen

The default is to start the FTP Server at boot time, unless the following registry entry is defined and set to "0" which disables AutoFTP. "1" enables the AutoFTP. The entry can be set from the NDISTRAY pop-up menu by selecting either **AutoFTP On** or **AutoFTP Off**.

`HKEY_LOCAL_MACHINE\Software\Intermec\Ndistray\StartupIFTP`

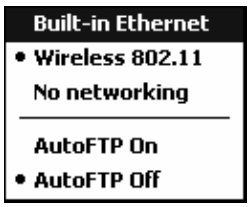
These new entries are located below the selections to load the network drivers. If the StartupIFTP registry key is not defined, the FTP Server is loaded by default, to provide "out-of-the-box" capability for customers who want to begin loading files to the 700 Series Computer without any prior configuration.

Note: If a network driver is unloaded using the NDISTRAY popup menu, and the FTP Server is running, the FTP Server is stopped.

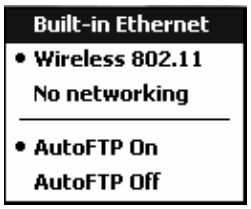
On a resume, if AutoFTP is enabled and the FTP Server is running, it is stopped and restarted. NDISTRAY uses a helper application named RESETIFTP to implement the restart on resume feature.

To do an AutoFTP Installation Check:

1. Ensure the FTP Server is running "out-of-the-box" the first time.
2. Tap **Start** > **Today** to access the Today screen, then tap the antenna icon in the System Tray to bring up the NDISTRAY pop-up menu. Select **AutoFTP Off** to disable AutoFTP. Do a warm boot and confirm the FTP Server is not running.



3. Tap **Start** > **Today** to access the Today screen, then tap the antenna icon in the System Tray to bring up the NDISTRAY pop-up menu. Select **AutoFTP On** to enable AutoFTP, reboot, confirm it is running.



4. Unload the network driver when the FTP Server is running and confirm that it is not running any more.
5. Load the FTP Server, establish a connection, then suspend and resume. The server should still run, but the FTP connection to the client should be dropped.

Kernel I/O Controls

This describes the KernelIoControl() functions available to application programmers. Most C++ applications will need to prototype the function as the following to avoid link and compile errors.

```
extern “C” BOOL KernelIoControl(DWORD dwIoControlCode, LPVOID lpInBuf, DWORD
nInBufSize, LPVOID lpOutBuf, DWORD nOutBufSize, LPDWORD lpBytesReturned);
```

IOCTL_HAL_GET_DEVICE_INFO

This IOCTL returns either the platform type or the OEMPLATFORM name based on an input value.

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_DEVICE_INFO, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Points to a DWORD containing either the SPI_GETPLATFORMTYPE or SPI_GETOEMINFO value.
lpInBufSize	Must be set to sizeof(DWORD).
lpOutBuf	Must point to a buffer large enough to hold the return data of the function. If SPI_GETPLATFORMTYPE is specified in *lpInBuf*, then the “PocketPC\0” Unicode string is returned. If SPI_GETOEMINFO is specified in *lpInBuf*, then the “Intermec 700\0” Unicode string is returned.
nOutBufSize	The size of *lpOutBuf* in bytes. Must be large enough to hold the string returned.
lpBytesReturned	The actual number of bytes returned by the function for the data requested.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_ITC_READ_PARM

Usage
#include "oemioctl.h"

Syntax
```
BOOL KernelIoControl( IOCTL_HAL_ITC_READ_PARM,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Points to this structure. See "*ID Field Values*" below.
	`struct PARMS {`
	`  BYTE id;`
	`  BYTE ClassId;`
	`};`

nInBufSize	Must be set to the size of the PARMS structure.
lpOutBuf	Must point to a buffer large enough to hold the return data of the function. If this field is set to NULL and *nOutBufSize* is set to zero when the function is called the function will return the number bytes required by the buffer.
nOutBufSize	The size of *lpOutBuf* in bytes.
lpBytesReturned	Number of bytes returned by the function for the data requested.

Return Values
Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the error value. Either ERROR_INVALID_PARAMETER or ERROR_INSUFFICIENT_BUFFER may be returned when this function is used to get the error.

ID Field Values
The *id* field of the PARMS structure may be one of the following values:

ID Field Values
ITC_NVPARM_ETHERNET_ID This IOCTL returns the Ethernet 802.11 MAC Address. Six bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_SERIAL_NUM This IOCTL returns the serial number of the device in BCD format. Six bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_MANF_DATE This IOCTL returns the device date of manufacture in the BCD YYYY/MM/DD format. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_SERVICE_DATE This IOCTL returns the device's date of last service in BCD YYYY/MM/DD format. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.

ID Field Values (continued)
ITC_NVPARM_DISPLAY_TYPE This IOCTL returns the device's display type. One byte is returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_EDG_IP This IOCTL returns the device Ethernet debug IP address. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_EDBG_SUBNET This IOCTL returns the device Ethernet debug subnet mask. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_ECN This IOCTL returns ECNs applied to the device in a bit array format. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_CONTRAST This IOCTL returns the device default contrast setting. Two bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_MCODE This IOCTL returns the manufacturing configuration code for the device. Sixteen bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
ITC_NVPARM_VERSION_NUMBER This IOCTL returns the firmware version for various system components. These values for the *ClassId* field of the PARMS structure are allowed when ITC_NVPARM_VERSION_NUMBER is used in the *id* field: • **VN_CLASS_KBD** Returns a five-byte string, including null terminator, that contains an ASCII value which represents the keypad microprocessor version in the system. The format of the string is *x.xx* with a terminating null character. • **VN_CLASS_ASIC** Returns a five-byte string, including null terminator, that contains an ASCII value which represents the version of the FPGA firmware in the system. The format of the string is *x.xx* with a terminating null character. • **VN_CLASS_BOOTSTRAP** Returns a five-byte string, including null terminator, that contains an ASCII value which represents the version of the Bootstrap Loader firmware in the system. The format of the string is *x.xx* with a terminating null character.
ITC_NVPARM_INTERMEC_SOFTWARE_CONTENT This IOCTL reads the manufacturing flag bits from the non-volatile data store that dictates certain software parameters. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer* that indicates if Intermec Content is enabled in the XIP regions. TRUE indicates that it is enabled. FALSE indicates that it is not enabled.
ITC_NVPARM_ANTENNA_DIVERSITY This IOCTL reads the state of the antenna diversity flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer* that indicates if there is a diversity antenna installed. TRUE indicates that it is installed. FALSE indicates that it is not installed.
ITC_NVPARM_WAN_RI This IOCTL reads the state of the WAN ring indicator flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer* that indicates the polarity of the WAN RI signal. TRUE indicates active high. FALSE indicates active low.

ID Field Values (continued)
ITC_NVPARM_RTC_RESTORE This IOCTL reads the state of the real-time clock restore flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the RTC is restored upon a cold boot. FALSE indicates that the RTC is not restored.
ITC_NVPARM_INTERMEC_DATACOLLECTION_SW This IOCTL reads the state of the data collection software enabled flag. A BOOLEAN DWORD is returned in the buffer pointer to by *lpOutBuffer* that indicates the data collection software is to install at boot time. FALSE indicates the data collection software should not install.
ITC_NVPARM_INTERMEC_DATACOLLECTION_HW This IOCTL reads the data collection hardware flags. A BYTE is returned in the buffer pointer to by *lpOutBuffer* that indicates the type of data collection hardware installed. The maximum possible value returned is ITC_DEVID_SCANHW_MAX. • **ITC_DEVID_SCANHW_NONE** No scanner hardware is installed. • **ITC_DEVID_OEM2D_IMAGER** OEM 2D imager is installed. • **ITC_DEVID_INTERMEC2D_IMAGER** Intermec 2D imager is installed. • **ITC_DEVID_SE900_LASER** SE900 laser is installed. • **ITC_DEVID_SE900HS_LASER** SE900HS laser is installed. • **ITC_DEVID_INTERMEC_EVIO** EVIO linear imager is installed. The high bit indicates whether the S6 scanning engine is installed. The bit mask for this is ITC_DEVID_S6ENGINE_MASK. A non-zero value indicates that the S6 scanning engine is installed.
ITC_NVPARM_WAN_INSTALLED This IOCTL reads the state of the WAN radio installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the WAN radio is installed. FALSE indicates that no WAN radio is installed.
ITC_NVPARM_WAN_FREQUENCY This IOCTL reads the state of the WAN radio frequency flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the WAN radio frequency is United States. FALSE indicates that the WAN radio frequency is European.
ITC_NVPARM_WAN_RADIOTYPE This IOCTL reads the WAN radio ID installed by manufacturing. A BYTE is returned in the buffer pointer to by *lpOutBuffer* which indicates the type of WAN radio hardware installed. The maximum possible value returned is ITC_DEVID_WANRADIO_MAX. The current definitions are: • **ITC_DEVID_WANRADIO_NONE** No WAN radio installed. • **ITC_DEVID_WANRADIO_SIERRA_SB555** CDMA Sierra Wireless radio. • **ITC_DEVID_WANRADIO_XIRCOM_GEM3503** GSM/GPRS Intel (Xircom) radio. • **ITC_DEVID_WANRADIO_SIEMENS_MC45** GSM/GPRS Siemens radio.
ITC_NVPARM_80211_INSTALLED This IOCTL reads the state of the 802.11b radio installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the 802.11b radio is installed. FALSE indicates that no 802.11b radio is installed.
ITC_NVPARM_80211_RADIOTYPE This IOCTL reads the 802.11b radio ID installed by manufacturing. A BYTE is returned in the buffer pointer to by *lpOutBuffer* that indicates the type of 802.11b radio hardware installed. The maximum possible value returned is ITC_DEVID_80211RADIO_MAX. The current definitions are: • **ITC_DEVID_80211RADIO_NONE** No 802.11b radio installed. • **ITC_DEVID_80211RADIO_INTEL_2011B** Intel 2011B radio installed.

ID Field Values (continued)
ITC_NVPARM_BLUETOOTH_INSTALLED This IOCTL reads the state of the Bluetooth radio installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the Bluetooth radio is installed. FALSE indicates that no Bluetooth radio is installed.
ITC_NVPARM_SERIAL2_INSTALLED This IOCTL reads the state of the serial 2 (COM2) device installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the serial 2 device is installed. FALSE indicates that no serial 2 device is installed.
ITC_NVPARM_VIBRATE_INSTALLED This IOCTL reads the state of the vibrate device installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the vibrate device is installed. FALSE indicates that no vibrate device is installed.
ITC_NVPARM_LAN9000_INSTALLED This IOCTL reads the state of the Ethernet device installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the Ethernet device is installed. FALSE indicates that no Ethernet device is installed.
ITC_NVPARM_SIM_PROTECT_HW_INSTALLED This IOCTL reads the state of the SIM card protection hardware installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the SIM card protection hardware is installed. FALSE indicates that no SIM card protection hardware is installed.
ITC_NVPARM_SIM_PROTECT_SW_INSTALLED This IOCTL reads the state of the SIM card protection software installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the SIM card protection software is installed. FALSE indicates that no SIM card protection software is installed.
ITC_NVPARM_SIM_PROTECT_SW_INSTALLED This IOCTL reads the state of the SIM card protection software installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the SIM card protection software is installed. FALSE indicates that no SIM card protection software is installed.

IOCTL_HAL_ITC_WRITE_SYSPARM

Describes and enables the registry save location.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_ITC_WRITE_SYSPARM,LPVOID
lpInBuf,DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	A single byte that may be one of the *id* values. See “*ID Field Values*” on the next page.
nInBufSize	Must be set to the size of the *lpInBuf* in bytes.
lpOutBuf	Must point to a buffer large enough to hold the data to be written to the non-volatile data store.
nOutBufSize	The size of *lpOutBuf* in bytes.
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the error value. Either ERROR_INVALID_PARAMETER or ERROR_INSUFFICIENT_BUFFER may be returned when this function is used to get the error.

ID Field Values

The *id* field of *lpInBuf* may be one of the following values:

ID Field Values

ITC_REGISTRY_SAVE_ENABLE
This function enables or disables the save registry to non-volatile media feature of the RegFlushKey() function. *lpOutBuf* must be set to zero (FALSE) if the feature is to be disabled or one (TRUE) if the feature is to be enabled.

ITC_ DOCK_SWITCH
This IOCTL sets a position of the dock switch. The dock switch may be set to either "modem" or "serial" positions. *lpOutBuf* must point to a buffer that contains a byte value of either DOCK_MODEM or DOCK_SERIAL as defined in OEMIOCTL.H; the value specifies the position the switch is to be set. The call appears as follows:

```
// port = DOCK_MODEM or DOCK_SERIAL as defined in oemioctl.h
BOOL SetDockSwitch( BYTE port)
{
 DWORD cmd = ITC_DOCK_SWITCH;
 DWORD cbRet;

 return KernelIoControl(IOCTL_HAL_ITC_WRITE_SYSPARM,&cmd, sizeof(cmd),
&port,sizeof(port),&cbRet)
}
```

ITC_ WAKEUP_MASK
This IOCTL sets a bit mask that represents the mask for the five programmable wakeup keys. The I/O key is not a programmable wakeup key. By default it is always the system resume key and all other keys are set to disable key wakeup. A zero in a bit position masks the wakeup for that key. A one in a bit position enables wakeup for that key. *lpOutBuf* must point to a buffer that contains a byte value of a wakeup mask consisting of the OR'ed constants as defined in OEMIOCTL.H. Only the following keys are programmable as wakeup events.

```
#define  SCANNER_TRIGGER 1
#define  SCANNER_LEFT    2
#define  SCANNER_RIGHT   4
#define  GOLD_A1         8
#define  GOLD_A2         0x10
```

ITC_AMBIENT_KEYBOARD *(does not apply to the 730 Computer)*
This IOCTL sets the threshold for the keypad ambient sensor. This can be a value from 0 (always off) to 255 (always on). *lpOutBuf* must point to a buffer that contains a byte value of the desired setting.

ITC_AMBIENT_FRONTLIGHT *(does not apply to the 730 Computer)*
This IOCTL sets the threshold for the frontlight ambient sensor. This can be a value from 0 (always off) to 255. *lpOutBuf* must point to a buffer that contains a byte value of the desired setting.

IOCTL_HAL_GET_DEVICEID

This IOCTL returns the device ID. There are two types of device IDs supported, which are differentiated based on the size of the *output* buffer. The UUID is returned if the buffer size is set to *sizeof(UNIQUE_DEVICEID)*, otherwise the oldstyle device ID is returned.

Usage

#include “pkfuncs.h”
#include “deviceid.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_DEVICEID,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL. STRICT_ID settings are not supported.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a UNIQUE_DEVICEID structure as defined by DEVICEID.H if the UUID is to be returned
nOutBufSize	The size of the UNIQUE_DEVICEID in bytes if the UUID is to be returned. A DEVICE_ID as defined by PKFUNCS.H is returned if the size in bytes is greater than or equal to *sizeof(DEVICE_ID)*.
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_GET_OAL_VERINFO

Returns the HAL version information of the Pocket PC image.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_OAL_VERINFO,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a VERSIONINFO structure as defined by OEMIOCTL.H. The fields should have these values: • `cboemverinfo` sizeof (tagOemVerInfo); • `verinfover` 1 • `sig;` “ITC\0” • `id;` ‘N’ • `tgtcustomer` “” • `tgtplat` SeaRay • `tgtplatversion` Current build version number • `tgtcputype[8];` “Intel\0” • `tgtcpu` “PXA255\0”; • `tgtcoreversion` “” • `date` Build time • `time` Build date
nOutBufSize	The size of VERSIONINFO in bytes.
lpBytesReturned	Returns *sizeof(PVERSIONINFO)*.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_GET_BOOTLOADER_VERINFO

Returns the HAL version information of the Pocket PC image.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_OAL_VERINFO,LPVOID
lpInBuf, DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
nInBufSize	Should be set to zero.
lpOutBuf	Must point to a VERSIONINFO structure as defined by OEMIOCTL.H. The fields should have these values: • `cboemverinfo` Sizeof (tagOemVerInfo); • `verinfover` 1 • `sig;` “ITC\0” • `id;` ‘B’ • `tgtcustomer` “” • `tgtplat` SeaRay • `tgtplatversion` Current build version number of the bootstrap loader • `tgtcputype[8];` “Intel\0”; • `tgtcpu` “PXA255\0” • `tgtcoreversion` “” • `date` Build time • `time` Build date
nOutBufSize	The size of VERSIONINFO in bytes.
lpBytesReturned	The number of bytes returned to *lpOutBuf*.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_WARMBOOT

Causes the system to perform a warm-boot. The object store is retained.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_WARMBOOT,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_HAL_COLDBOOT

Causes the system to perform a cold-boot. The object store is cleared.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_COLDBOOT,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_HAL_GET_RESET_INFO

This IOCTL code allows software to check the type of the most recent reset.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_RESET_INFO,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a HAL_RESET_INFO structure. See sample below.
nOutBufSize	The size of HAL_RESET_INFO in bytes.
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

Sample

```
typedef struct {
  DWORD ResetReason;                        // most recent reset type
  DWORD ObjectStoreState;                   // state of object store
} HAL_RESET_INFO, * PHAL_RESET_INFO;

// Reset reason types
#define HAL_RESET_TYPE_UNKNOWN            0
#define HAL_RESET_REASON_HARDWARE         1  // cold
#define HAL_RESET_REASON_SOFTWARE         2  // suspend
#define HAL_RESET_REASON_WATCHDOG         4
#define HAL_RESET_BATT_FAULT              8  // power fail
#define HAL_RESET_VDD_FAULT               16 // warm boot

// Object store state flags
#define HAL_OBJECT_STORE_STATE_UNKNOWN    0
#define HAL_OBJECT_STORE_STATE_CLEAR      1
```

IOCTL_HAL_GET_BOOT_DEVICE

This IOCTL code allows software to check which device CE booted from.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_BOOT_DEVICE,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.

lpOutBuf	Must point to a buffer large enough to hold a DWORD (4 bytes) that contains the boot device. The following boot devices are supported:
	`#define HAL_BOOT_DEVICE_UNKNOWN 0`
	`#define HAL_BOOT_DEVICE_ROM_XIP 1`
	`#define HAL_BOOT_DEVICE_ROM 2`
	`#define HAL_BOOT_DEVICE_PCMCIA_ATA 3`
	`#define HAL_BOOT_DEVICE_PCMCIA_LINEAR 4`
	`#define HAL_BOOT_DEVICE_IDE_ATA 5`
	`#define HAL_BOOT_DEVICE_IDE_ATAPI 6`

nOutBufSize	The size of *lpOutBuf* in bytes (4).
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_REBOOT

Causes the system to perform a warm-boot. The object store is retained.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_REBOOT,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_PROCESSOR_INFORMATION

Returns processor information.

Usage

#include “pkfuncs.h”

Syntax

```
BOOL KernelIoControl( IOCTL_PROCESSOR_INFORMATION,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
nInBufSize	Should be set to zero.
lpOutBuf	Should be a pointer to the PROCESSOR_INFO structure. The PROCESSOR_INFO structure stores information that describes the CPU more descriptively.

```
typedef __PROCESSOR_INFO {
WORD   wVersion;             // Set to value 1
WCHAR  szProcessorCore[40];  // “ARM\0”
WORD   wCoreRevision;        // 4
WCHAR  szProcessorName[40];  // “PXA255\0”
WORD   wProcessorRevision;   // 0
WCAHR  szCatalogNumber[100]; // 0
WCHAR  szVendor[100];        // “Intel Corporation\0”
DWORD  dwInstructionSet;     // 0
DWORD  dwClockSpeed;         // 400
}
```

nOutBufSize	Should be set to sizeof(PROCESSOR_INFO) in bytes.
lpBytesReturned	Returns sizeof(PROCESSOR_INFO);

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_GET_CPU_ID

Returns Xscale processor ID.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_GET_CPU_ID,LPVOID lpInBuf,
DWORD nInBufSize,LPVOID lpOutBuf,DWORD nOutBufSize,LPDWORD
lpBytesReturned );
```

Parameters

lpInBuf	Should point to a CPUIdInfo structure defined in OEMIOCTL.H.
lpInBufSize	Should be *sizeof(CPUIdInfo)*.
lpOutBuf	Should be NULL.
nOutBufSize	Should be set to 0.
lpBytesReturned	Returns *sizeof(PROCESSOR_INFO);*

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

Network Selection APIs

The Network Selection APIs change the network adapter configuration programmatically. Both drivers support the same IOCTL function numbers for loading and unloading the drivers.

Loading and unloading of the 802.11b driver is performed by the FWL1: device in the system by performing DeviceIOControl() calls to the driver.

Loading and unloading of the driver for the built-in Ethernet adapter is performed by the SYI1: device in the system by performing DeviceIOControl() calls to the driver.

- For loading an NDIS driver associated with an adapter, the IOCTL is IOCTL_LOAD_NDIS_MINIPORT.
- For unloading NDIS drivers associated with an adapter the IOCTL is IOCTL_UNLOAD_NDIS_MINIPORT.

Example

```
#include <winioctl.h>
#include "sysio.h"
void DoLoad(int nDevice) {
   LPTSTR devs[] = { _T("SYI1:"), _T("FWL1:") };
   HANDLE hLoaderDev;
   DWORD bytesReturned;
   hLoaderDev = CreateFile(devs[nDevice], GENERIC_READ|GENERIC_WRITE, 0,
   NULL, OPEN_EXISTING, 0, NULL);
   if (hLoaderDev != INVALID_HANDLE_VALUE) {
      if (!DeviceIoControl( hLoaderDev, IOCTL_LOAD_NDIS_MINIPORT, NULL, -1, NULL, 0,
      &bytesReturned, NULL)){
         MessageBox(NULL, TEXT("SYSIO IoControl Failed"), TEXT("Network
         loader"),MB_ICONHAND);
         if (hLoaderDev!=INVALID_HANDLE_VALUE) CloseHandle(hLoaderDev);
         hLoaderDev = INVALID_HANDLE_VALUE; // bad handle
      }else {
         CloseHandle(hLoaderDev);
      }
   }
}
void DoUnload(int nDevice) {
   LPTSTR devs[] = { _T("SYI1:"), _T("FWL1:") };
   HANDLE hLoaderDev;
   DWORD bytesReturned;
   hLoaderDev = CreateFile(devs[nDevice], GENERIC_READ|GENERIC_WRITE, 0,
   NULL, OPEN_EXISTING, 0, NULL);
   if (hLoaderDev != INVALID_HANDLE_VALUE) {
      if (!DeviceIoControl( hLoaderDev, IOCTL_UNLOAD_NDIS_MINIPORT, NULL, -1, NULL, 0,
      &bytesReturned, NULL)){
         MessageBox(NULL, TEXT("SYSIO IoControl Failed"),TEXT("Network
         loader"),MB_ICONHAND);
         if (hLoaderDev!=INVALID_HANDLE_VALUE) CloseHandle(hLoaderDev);
         hLoaderDev = INVALID_HANDLE_VALUE; // bad handle
      }else {
         CloseHandle(hLoaderDev);
      }
   }
}
```

The API provided by Intermec Technologies exposes a limited set of routines that allows a programmer to access and affect the 802.11b network interface card from within their application. The routines provided also reads/writes values to the CE registry that pertain to the 802.11b radio driver. By using the provided functions, a programmer can alter the 802.11b parameters of Network Name (SSID), WEP keys, infrastructure modes, radio channel, and power management modes. A programmer can also retrieve network connect status and signal strength indication from the RF network card.

The API is contained within the 80211API.DLL file that should be present in any load with the 802.11b networking installed.

NETWLAN.DLL PRISMNDS.DLL	This file is the 802.11b driver. It is present in all 700 CE loads that use the 802.11b network interface card.
80211API.DLL	This file is an Intermec authored file that provides the programmer with a set of API calls to configure or monitor status of the 802.11b network.
MOD80211.DLL	The CORE module for the 802.11b NIC. It provides the 802.11b status information displayed when the CORE application is running.
80211CONF.EXE	This is the "Control Panel" for configuring the 802.11b network parameters. Note that it is an EXE file and is actually called by CPL802.CPL (see below). It is also called by the CORE application when the "Configuration" button is pressed.
CPL802.CPL	A control panel application that does nothing but call 80211CONF.EXE.
80211SCAN.EXE	Internally manages the Scan List activity.
802PM.DLL	This handles profile management for radio configurable values.
URODDSVC.EXE	This handles radio configuration and security authentication based on a selected profile.

The Profile Manager supports up to four radio configuration profiles. These profiles are the same as those set by the Wireless Network control panel applet that runs on the Windows CE unit. You can configure different 802.11b profiles and switch between them using the 802.11 API. See the ConfigureProfile() function on page 280 for more information.

Basic Connect/Disconnect Functions

Below are functions available for the 700 Series Color Computer when enabled with the 802.11b radio module.

RadioConnect()

Connects to the available radio. Use this function if you plan on using a lot of API calls that talk directly to the radio. Note that the 802.11b radio must be enabled via NDISTRAY before you can connect to it.

Syntax	`UINT RadioConnect( );`
Parameters	None.
Return Values	ERROR_SUCCESS when successful, otherwise ERR_CONNECT_FAILED
Remarks	Call this function before you call any other function found within this API. It hunts out and connects to the 802.11b radio available on the system. Check extended error codes if it returns anything else for information.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_RadioConnect)();`
	`#else`
	`UINT RadioConnect();`
	`#endif`

RadioDisconnect()

Call this function when done using the 802.11 API to clean up a connection from a previous RadioConnect() call. If you do not call this function, you may leave memory allocated.

Syntax	`UINT RadioDisconnect( );`
Parameters	None.
Return Values	ERROR_SUCCESS when successful, otherwise ERR_CONNECT_FAILED.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_RadioDisconnect)();`
	`#else`
	`UINT RadioDisconnect();`
	`#endif`

RadioDisassociate()

Call this function to have the 802.11b radio disassociate from the current service set. The radio then enters an "off" mode until it is woken again by setting the Service Set Identifier (SSID). Also, the NDIS driver generates an NDIS media disconnect event.

Syntax	`UINT RadioDisassociate( );`
Parameters	None.
Return Values	ERROR_SUCCESS when successful, otherwise ERR_CONNECT_FAILED.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_RadioDisassociate)();`
	`#else`
	`UINT RadioDisassociate();`
	`#endif`

Query Information Functions

GetAssociationStatus()

Call this function to obtain the radio's current association status with a service set.

Syntax	`UINT GetAssociationStatus( ULONG & );`	
Parameters	NDIS_RADIO_ASSOCIATED	Indicates the radio is associated with an access point
	NDIS_RADIO_SCANNING	Indicates the radio is looking for an access point with which to associate
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	Data is only valid if the function returns ERROR_SUCCESS. Also, if ERROR_SUCCESS is returned, your ULONG reference is populated by one of the parameters listed above.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetAssociationStatus)(ULONG &);`
	`#else`
	`UINT GetAssociationStatus(ULONG &);`
	`#endif`	

GetAuthenticationMode()

Call this function to obtain the radio's current authentication mode.

Syntax	`UINT GetAuthenticationMode( ULONG & );`	
Parameters	NDIS_RADIO_AUTH_MODE_OPEN	802.11b Open Authentication. Indicates that the radio is using an open system.
	NDIS_RADIO_AUTH_MODE_SHARED	802.11b Shared Authentication. Indicates that the radio is using a shared key.
	NDIS_RADIO_AUTH_MODE_AUTO	Auto switch between Open/Shared. Indicates automatic detection is used when available.
	NDIS_RADIO_AUTH_MODE_ERROR	Defined as error value. Indicates the authentication mode was not determined at this time or is unknown.
	NDIS_RADIO_AUTH_MODE_WPA	WPA Authentication
	NDIS_RADIO_AUTH_MODE_WPA_PSK	WPA Preshared Key Authentication
	NDIS_RADIO_AUTH_MODE_WPA_NONE	WPA None
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	Data is only valid if ERROR_SUCCESS is returned. Also, if ERROR_SUCCESS is returned, your USHORT reference is populated with one of the parameters listed above.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetAuthenticationMode)(ULONG &);`
	`#else`
	`UINT GetAuthenticationMode(ULONG &);`
	`#endif`	

GetBSSID()

Call this function to get the current MAC address (BSSID) of the service set. In ESS mode, this is the MAC address of the access point the radio is associated with. In IBSS mode, this is a randomly generated MAC address, and serves as the ID for the IBSS.

Syntax	`UINT GetBSSID( TCHAR * );`
Parameters	Pointer to a character array, which is populated with the current BSSID after a successful call.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your TCHAR array is populated with the BSSID of the current service set: `xx-xx-xx-xx-xx-xx`

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetBSSID)(TCHAR *);`
	`#else`
	`UINT GetBSSID(TCHAR *);`
	`#endif`

GetDiversity()

Call this function to get the current diversity setting of your 802.11b radio. This uses an optional NDIS5.1 OID to query the radio, of which a large number of 802.11b devices do not support. This may be inaccurate.

Syntax	`UINT GetDiversity(USHORT *);`	
Parameters	ANT_PRIMARY	The primary antenna is selected.
	ANT_SECONDARY	The secondary antenna is selected.
	ANT_DIVERSITY	The radio is in diversity mode, and uses both antennas
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your USHORT reference is populated with one of the parameters listed above.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetDiversity)(USHORT *);`
	`#else`
	`UINT GetDiversity(USHORT *);`
	`#endif`	

GetLinkSpeed()

Call this function to get the current link speed of the 802.11b radio.

Syntax	`UINT GetLinkSpeed( int & );`
Parameters	This function accepts an int reference, and your int is populated with the current link speed, in Mbps, rounded to the nearest whole integer, for example: 1, 2, 5, 11, etc.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	Data returned is valid if ERROR_SUCCESS is returned.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetLinkSpeed)(int &);`
	`#else`
	`UINT GetLinkSpeed(int &);`
	`#endif`

GetMac()

Call this function to get the MAC address of the 802.11b radio.

Syntax	`UINT GetMac( TCHAR * );`
Parameters	Pointer to a character array, which is populated with the MAC address after a successful call.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your TCHAR array is populated with the formatted MAC address of the adapter, as follows: `xx-xx-xx-xx-xx-xx`

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetMac)(TCHAR *);`
	`#else`
	`UINT GetMac(TCHAR *);`
	`#endif`

Note: Be sure to call RadioConnect() *before* calling this function for this function to work properly.

GetNetworkMode()

Call this function to get the current Network Mode (SSID) for the 802.11b radio.

Syntax	`UINT GetNetworkMode( ULONG & );`	
Parameters	NDIS_NET_MODE_IBSS	802.11 Ad-Hoc Mode.
	NDIS_NET_MODE_ESS	802.11 Infrastructure Mode.
	NDIS_NET_MODE_UNKNOWN	Anything Else/Unknown Error
	NDIS_NET_AUTO_UNKNOWN	Automatic Selection. *Use of this option is not supported or recommended.*
	NDIS_NET_TYPE_OFDM_5G	5 Gigahertz 54 Mbps
	NDIS_NET_TYPE_OFDM_2_4G	802.11g 2.4 Gigahertz
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of the parameters listed above.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetNetworkMode)(ULONG &);`
	`#else`
	`UINT GetNetworkMode(ULONG &);`
	`#endif`	

GetNetworkType()

Call this function to get the current network type of the radio. Do not confuse this with GetNetworkMode().

Syntax	`UINT GetNetworkType( ULONG & );`	
Parameters	NDIS_NET_TYPE_FH	Indicates this is a frequency hopping radio.
	NDIS_NET_TYPE_DS	Indicates that this is a direct sequence radio.
	NDIS_NET_TYPE_UNDEFINED	Indicates this radio type is unknown or undefined.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of the parameters listed above.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetNetworkType)(ULONG &);`
	`#else`
	`UINT GetNetworkType(ULONG &);`
	`#endif`	

GetSSID()

Call this function to get the desired SSID of the 802.11b radio.

Syntax	`UINT GetSSID( TCHAR * );`
Parameters	Pointer to a character array, which is populated with the current SSID when successful.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your TCHAR array is populated with the desired SSID.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetSSID)(TCHAR *);`
	`#else`
	`UINT GetSSID(TCHAR *);`
	`#endif`

Note: Call RadioConnect() *before* this function for this function to work properly.

GetPowerMode()

Call this function to get the current power savings mode of the radio.

Syntax	`UINT GetPowerMode( ULONG & );`	
Parameters	NDIS_RADIO_POWER_MODE_CAM	Continuous Access Mode *(ie: always on)*.
	NDIS_RADIO_POWER_MODE_PSP	Power Saving Mode.
	NDIS_RADIO_POWER_UNKNOWN	Unknown power mode.
	NDIS_RADIO_POWER_AUTO	Auto. *(Available for 730 Mobile Computers)*
	NDIS_RADIO_POWER_MODE_FAST_PSP	Fast PSP, good savings, fast
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of the parameters listed above.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetPowerMode)(ULONG &);`
	`#else`
	`UINT GetPowerMode(ULONG &);`
	`#endif`	

Note: Do not use Automatic Switching mode at this time.

GetRSSI()

Call this function to get the current RSSI (Radio Signal Strength Indicator), in Dbm.

Syntax	`UINT GetRSSI( ULONG & );`
Parameters	References a ULONG that is populated with the current RSSI after a successful call.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your ULONG reference contains the RSSI. Valid RSSI range is from –100 Dbm to –30 Dbm.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetRSSI)(ULONG &);`
	`#else`
	`UINT GetRSSI(ULONG &);`
	`#endif`

GetTXPower()

Call this function to get the current transmit power of the radio.

Syntax	`UINT GetTXPower( ULONG & );`	
Parameters	NDIS_POWER_LEVEL_63	63 mW
	NDIS_POWER_LEVEL_30	30 mW
	NDIS_POWER_LEVEL_15	15 mW
	NDIS_POWER_LEVEL_5	5 mW
	NDIS_POWER_LEVEL_1	1 mW
	NDIS_POWER_LEVEL_UNKNOWN	Unknown Value or Error.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with the TX power in milliwatts (mW). Valid ranges are from 5 mW to 100 mW.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetTXPower)(ULONG &);`
	`#else`
	`UINT GetTXPower(ULONG &);`
	`#endif`	

GetWepStatus()

Call this function to get the current state of the radio's WEP and encryption levels.

Syntax	`UINT GetWepStatus( ULONG & );`	
Parameters	NDIS_ENCRYPTION_1_ENABLED	WEP is enabled; TKIP and AES are not enabled, and a transmit key may or may not be available. *(same as NDIS_RADIO_WEP_ENABLED)*
	NDIS_ENCRYPTION_DISABLED	Indicates that AES, TKIP, and WEP are disabled, and a transmit key is available. *(Same as NDIS_RADIO_WEP_DISABLED)*
	NDIS_ENCRYPTION_NOT_SUPPORTED	Indicates encryption (WEP, TKIP, AES) is not supported. *(Same as NDIS_RADIO_WEP_NOT_SUPPORTED)*
	NDIS_ENCRYPTION_1_KEY_ABSENT	Indicates that AES, TKIP, and WEP are disabled, and a transmit key is not available. *(Same as NDIS_RADIO_WEP_ABSENT)*
	NDIS_ENCRYPTION_2_ENABLED	Indicates that TKIP and WEP are enabled; AES is not enabled, and a transmit key is available.
	NDIS_ENCRYPTION_2_KEY_ABSENT	Indicates that there are no transmit keys available for use by TKIP or WEP, TKIP and WEP are enabled; and AES is not enabled.
	NDIS_ENCRYPTION_3_ENABLED	Indicates that AES, TKIP, and WEP are enabled, and a transmit key is available.
	NDIS_ENCRYPTION_3_KEY_ABSENT	Indicates that there are no transmit keys available for use by AES, TKIP, or WEP, and AES, TKIP, and WEP are enabled.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of the parameters listed above.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetWepStatus)(ULONG &);`
	`#else`
	`UINT GetWepStatus(ULONG &);`
	`#endif`	

GetRadioIpAddress()

Call this function to obtain a formatted string indicating whether DHCP is enabled, and what is the current adapters IP address.

Syntax	`UINT GetRadioIpAddress( TCHAR * );`
Parameters	Pointer to a character array that contains the formatted string of the IP address and static/DHCP information.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your TCHAR array contains a string formatted as follows: `IP: DHCP Enabled\nxxx.xxx.xxx.xxx\n` or `IP: DHCP Disabled\nxxx.xxx.xxx.xxx\n`

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetRadioIpAddress)(TCHAR *);`
	`#else`
	`UINT GetRadioIpAddress(TCHAR *);`
	`#endif`

GetCCXStatus()

Call this function to get information about the current CCX status of the adapter.

Syntax	`UINT GetCCXStatus( ULONG & );`	
Parameters	NDIS_NETWORK_EAP_MODE_OFF	Disable EAP mode.
	NDIS_NETWORK_EAP_MODE_ON	Enable EAP mode.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of parameters listed above.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetCCXStatus)(ULONG &);`
	`#else`
	`UINT GetCCXStatus(ULONG &);`
	`#endif`	

Set Information Functions

AddWep()

Call this function to add a WEP key to the radio. Call this function multiple times when adding more than one WEP key. Save the "default" key for last. For example, when adding four keys, and the second key is the default transmit key, add keys 1, 3 and 4 *before* you add key 2.

Note: Add the default transmit key *last*.

Syntax	`UINT AddWep( ULONG, BOOL, TCHAR * );`	
Parameters	ULONG	Specifies the key index to be set. Valid values are 0–3.
	BOOL	When set to TRUE, specifies that this key is the default transmit key.
	TCHAR	Pointer to a character array that specifies the key data in either HEX (length of 10 or 26) or ASCII (length of 5 or 13). This string must be null-terminated.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	When adding WEP keys to the radio, turn off encryption before you add the keys, then turn encryption back on afterwards. Also, be sure to add the TRANSMIT KEY last.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_AddWep)(ULONG, BOOL, TCHAR *);`
	`#else`
	`UINT AddWep(ULONG, BOOL, TCHAR *);`
	`#endif`	

EnableWep()

Enables or disables WEP encryption on the radio (TRUE/FALSE).

Syntax	`UINT EnableWep( BOOL );`
Parameters	Set BOOL to TRUE to enable WEP encryption, or FALSE to disable WEP encryption.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.

Remarks	Call this function with TRUE as the parameter to enable WEP encryption. Call this function with the FALSE parameter to disable WEP encryption. This call is an alias for EncryptionStatus(). See the following:
	`EnableWEP(TRUE) = EncryptionStatus(NDIS_ENCRYPTION_1_ENABLED)`
	`EnableWEP(FALSE) = EncryptionStatus(NDIS_ENCRYPTION_DISABLED)`

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_EnableWep)(BOOL);`
	`#else`
	`UINT EnableWep(BOOL);`
	`#endif`

EncryptionStatus()

Call this function to set the desired encryption status.

Syntax	`UINT EncryptionStatus( UINT mode );`	
Parameters	NDIS_ENCRYPTION_1_ENABLED	WEP is enabled; TKIP and AES are not enabled, and a transmit key may or may not be available. *(same as NDIS_RADIO_WEP_ENABLED)*
	NDIS_ENCRYPTION_DISABLED	Indicates that AES, TKIP, and WEP are disabled, and a transmit key is available. *(Same as NDIS_RADIO_WEP_DISABLED)*
	NDIS_ENCRYPTION_NOT_SUPPORTED	Indicates that encryption (WEP, TKIP, and AES) is not supported. *(Same as NDIS_RADIO_WEP_NOT_SUPPORTED)*
	NDIS_ENCRYPTION_1_KEY_ABSENT	Indicates that AES, TKIP, and WEP are disabled, and a transmit key is not available. *(Same as NDIS_RADIO_WEP_ABSENT)*
	NDIS_ENCRYPTION_2_ENABLED	Indicates that TKIP and WEP are enabled; AES is not enabled, and a transmit key is available.
	NDIS_ENCRYPTION_2_KEY_ABSENT	Indicates that there are no transmit keys available for use by TKIP or WEP, TKIP and WEP are enabled; and AES is not enabled.
	NDIS_ENCRYPTION_3_ENABLED	Indicates that AES, TKIP, and WEP are enabled, and a transmit key is available.
	NDIS_ENCRYPTION_3_KEY_ABSENT	Indicates that there are no transmit keys available for use by AES, TKIP, or WEP, and AES, TKIP, and WEP are enabled.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_EncryptionStatus)(UINT mode);`
	`#else`
	`UINT EncryptionStatus(UINT mode);`
	`#endif`	

SetAuthenticationMode()

Call this function to set the desired authentication mode.

Syntax	`UINT SetAuthenticationMode( ULONG );`	
Parameters	NDIS_RADIO_AUTH_MODE_OPEN	802.11b Open Authentication. Indicates that the radio is using an open system.
	NDIS_RADIO_AUTH_MODE_SHARED	802.11b Shared Authentication. Indicates that the radio is using a shared key.
	NDIS_RADIO_AUTH_MODE_AUTO	Auto switch between Open/Shared. Indicates automatic detection is used when available.
	NDIS_RADIO_AUTH_MODE_ERROR	Defined as error value. Indicates the authentication mode was not determined at this time or is unknown.
	NDIS_RADIO_AUTH_MODE_WPA	WPA Authentication
	NDIS_RADIO_AUTH_MODE_WPA_PSK	WPA Preshared Key Authentication
	NDIS_RADIO_AUTH_MODE_WPA_NONE	WPA None
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_SetAuthenticationMode)(ULONG);`
	`#else`
	`UINT SetAuthenticationMode(ULONG);`
	`#endif`	

SetChannel()

This function is currently not implemented. Ad-hoc networks automatically select a channel or use the already existing channel.

Syntax	`UINT SetChannel( USHORT );`
Parameters	USHORT value that should populate with the desired channel (1–14).
Return Values	None.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_SetChannel)(USHORT);`
	`#else`
	`UINT SetChannel(USHORT);`
	`#endif`

SetNetworkMode()

Call this function to set the desired Network Mode.

Syntax	`UINT SetNetworkMode( ULONG );`	
Parameters	NDIS_NET_MODE_IBSS	802.11 Ad-Hoc Mode.
	NDIS_NET_MODE_ESS	802.11 Infrastructure Mode.
	NDIS_NET_MODE_UNKNOWN	Anything Else/Unknown Error
	NDIS_NET_AUTO_UNKNOWN	Automatic Selection. *Use of this option is not supported or recommended.*
	NDIS_NET_TYPE_OFDM_5G	5 Gigahertz 54 Mbps
	NDIS_NET_TYPE_OFDM_2_4G	802.11g 2.4 Gigahertz
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_SetNetworkMode)(ULONG);`
	`#else`
	`UINT SetNetworkMode(ULONG);`
	`#endif`	

SetPowerMode()

Call this function to set the desired power mode.

Syntax	`UINT SetPowerMode( ULONG mode );`	
Parameters	NDIS_RADIO_POWER_MODE_CAM	Continuous Access Mode *(ie: always on)*.
	NDIS_RADIO_POWER_MODE_PSP	Power Saving Mode.
	NDIS_RADIO_POWER_UNKNOWN	Unknown power mode.
	NDIS_RADIO_POWER_AUTO	Auto. *(Available for 730 Computers)*
	NDIS_RADIO_POWER_MODE_FAST_PSP	Fast PSP, good savings, fast
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_SetPowerMode)(ULONG mode);`
	`#else`
	`UINT SetPowerMode(ULONG mode);`
	`#endif`	

SetSSID()

Call this function with a pointer to a null-terminated TCHAR array containing the desired SSID to set the desired SSID of the adapter.

Syntax	`UINT SetSSID( TCHAR * );`
Parameters	Pointer to a character array that contains the desired SSID. This should be null-terminated.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If an "ANY" network is desired, pass in _T("ANY").

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_SetSSID)(TCHAR *);`
	`#else`
	`UINT SetSSID(TCHAR *);`
	`#endif`

SetCCXStatus()

Call this function to set the desired CCX / Network EAP status.

Syntax	`UINT SetCCXStatus( ULONG );`	
Parameters	NDIS_NETWORK_EAP_MODE_OFF	Disable Network EAP / CCX
	NDIS_NETWORK_EAP_MODE_ON	Enable Network EAP / CCX
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_SetCCXStatus)(ULONG);`
	`#else`
	`UINT SetCCXStatus(ULONG);`
	`#endif`	

SetMixedCellMode()

Call this function to set the desired mixed cell mode.

Syntax	`UINT SetMixedCellMode( ULONG );`	
Parameters	NDIS_MIXED_CELL_OFF	Disable Mixed Cell
	NDIS_MIXED_CELL_ON	Enable Mixed Cell
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_SetMixedCellMode)(ULONG);`
	`#else`
	`UINT SetMixedCellMode(ULONG);`
	`#endif`	

RemoveWep()

Call this function with a key index of 0–3 to remove the WEP key at that index.

Syntax	`UINT RemoveWep( ULONG );`
Parameters	ULONG value that specifies the key index to set. Valid values are 0–3.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	On disassociation with all BSSIDs of the current service set, WEP key is removed by the adapter.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_RemoveWEP)(ULONG);`
	`#else`
	`UINT RemoveWEP(ULONG);`
	`#endif`

Helper Functions

ConfigureProfile()

If using the Intermec 802.11b Profile Management system, you can program the API to configure the radio to a specific profile by passing the profile name.

Syntax	`UINT ConfigureProfile( TCHAR * );`
Parameters	Pointer to a character array that contains the profile name. This should be null-terminated.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	Call this function with a pointer to a null-terminated TCHAR array that contains the name of the profile you wish to configure. This function reads profile data from the profile manager, sets that profile as the default active profile, and configures the radio appropriately. If needed, the supplicant and any other related services are automatically started and stopped.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_ConfigureProfile)(TCHAR *);`
	`#else`
	`UINT ConfigureProfile(TCHAR *);`
	`#endif`

EnableZeroConfig()

This enables or disables the Wireless Zero Configuration Wizard from Microsoft. After calling this function, a warm-boot is required for the change to take effect.

Note: Enabling this function effectively disables all the SET commands in this API.

Syntax	`UINT EnableZeroConfig( USHORT );`
Parameters	TRUE Enable Wireless Zero Config FALSE Disable Wireless Zero Config
Return Values	ERROR_SUCCESS when successful, ERR_ZERO_CONFIG_CHANGE_FAILED when the query failed.
Remarks	Call this function to set the desired Zero Config status.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_EnableZeroConfig)(USHORT);`
	`#else`
	`UINT EnableZeroConfig(USHORT);`
	`#endif`

isZeroConfigEnabled()

Call this function to determine whether Zero Config is currently enabled.

Syntax	`UINT isZeroConfigEnabled( );`
Parameters	None.
Return Values	TRUE if ZeroConfig is enabled, and FALSE if it is disabled.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_isZeroConfigEnabled)();`
	`#else`
	`UINT isZeroConfigEnabled();`
	`#endif`

isOrinoco()

Call this function to determine whether the current radio is an ORiNOCO, Lucent, or WaveLAN radio.

Syntax	`UINT isOrinoco( );`
Parameters	None.
Return Values	TRUE if this is an ORiNOCO radio, and FALSE if it is not.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_isOrinoco)();`
	`#else`
	`UINT isOrinoco();`
	`#endif`

isSupplicantRunning()

Call this function to determine whether the security supplicant is running.

Syntax	`UINT isSupplicantRunning( );`
Parameters	None.
Return Values	TRUE if the security supplicant is running, FALSE if it is not running.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_isSupplicantRunning)();`
	`#else`
	`UINT isSupplicantRunning();`
	`#endif`

StartScanList()

If a scan list is configured on the system, this causes the API to begin the process of scanning for an available network. This call can take quite a while to process *(depending upon the length of the scan list and how long it takes to find a valid network)*, you may wish to call it from a separate thread.

Syntax	`UINT StartScanList( );`
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	Call this function to start the scan list functionality of the system.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_StartScanList)();`
	`#else`
	`UINT StartScanList();`
	`#endif`

StartSupplicant()

Call this function to start the supplicant service if it is installed on the system.

Syntax	`UINT StartSupplicant( );`
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_StartSupplicant)();`
	`#else`
	`UINT StartSupplicant();`
	`#endif`

StopSupplicant()

Call this function to stop the supplicant service.

Syntax	`UINT StopSupplicant( );`
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_StopSupplicant)();`
	`#else`
	`UINT StopSupplicant();`
	`#endif`

isDHCPEnabled()

Call this function to determine whether DHCP is enabled on the current adapter.

Syntax	`UINT isDHCPEnabled( );`
Parameters	None.
Return Values	TRUE if DHCP is enabled, FALSE if it is not.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_isDHCPEnabled)();`
	`#else`
	`UINT isDHCPEnabled();`
	`#endif`

RenewDHCP()

Call this function to force a DHCP renewal on the current network adapter.

Syntax	`UINT RenewDHCP( );`
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	You should not have to call this function on Microsoft PocketPC 2003 or Microsoft Windows CE 4.2 .NET and later devices.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_RenewDHCP)();`
	`#else`
	`UINT RenewDHCP();`
	`#endif`

GetCurrentDriverName()

Call this function to populate the TCHAR array with the driver name.

Syntax	`UINT GetCurrentDriverName( TCHAR * );`
Parameters	Pointer to a TCHAR array which contains the name of the driver when successful.
Return Values	ERROR_SUCCESS when successful.
Remarks	This function is called with a pointer to a TCHAR array that is large enough to hold the name of the driver PLUS the null terminator.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_GetCurrentDriverName)(TCHAR *);`
	`#else`
	`UINT GetCurrentDriverName(TCHAR *);`
	`#endif`

ResetRadioToSystemSave()

Call this function to force the radio to reset to the last desired active profile.

Syntax	`UINT ResetRadioToSystemSave( );`
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	None.

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_ResetRadioToSystemSave)();`
	`#else`
	`UINT ResetRadioToSystemSave();`
	`#endif`

EnableSuppLogging()

Call this function to set the desired supplicant logging mode.

Syntax	`UINT EnableSuppLogging( ULONG );`	
Parameters	NDIS_SUPP_LOGGING_ON	Supplicant Logging Enabled
	NDIS_SUPP_LOGGING_OFF	Supplicant Logging Disabled
Return Values	ERROR_SUCCESS when successful.	
Remarks	None.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_EnableSuppLogging)(ULONG);`
	`#else`
	`UINT EnableSuppLogging(ULONG);`
	`#endif`	

SwitchPacketDriver()

Call this function to switch between available packet drivers on the system.

Syntax	`UINT SwitchPacketDriver( USHORT );`	
Parameters	INTERMEC_PACKET_DRIVER	Intermec Packet Driver (ZNICZIO)
	NDISUIO_PACKET_DRIVER	Microsoft Packet Driver (NDISUIO)
Return Values	ERROR_SUCCESS when successful.	
Remarks	After switching to a new packet driver, perform a warm boot for changes to take effect.	

Definitions	`#ifdef DYNAMIC_LOADING`
	`typedef UINT (*PFN_SwitchPacketDriver)(USHORT);`
	`#else`
	`UINT SwitchPacketDriver(USHORT);`
	`#endif`	

Deprecated Functions

The following functions are deprecated. While these are not removed from the API, these are no longer supported. Their parameters are no longer applicable and the return value for all of these functions is: ERR_FUNCTION_DEPRECATED

Function	Syntax
GetRTSThreshold(Deprecated)	`UINT GetRTSThreshold( USHORT & );`
GetMedia(Deprecated)	`UINT GetMedia( ULONG & );`
GetMedium(Deprecated)	`UINT GetMedium( ULONG & );`
GetNicStats(Deprecated)	`UINT GetNicStats( NDIS_802_11_STATISTICS & );`
SetRTSThreshold(Deprecated)	`UINT SetRTSThreshold( USHORT & );`
SetTXRate(Deprecated)	`UINT SetTXRate( UCHAR );`
EncryptWepKeyForRegistry(Deprecated)	`UINT EncryptWepKeyForRegistry( TCHAR * szDest, TCHAR * szSource );`
SetDiversity(Deprecicated)	`UINT SetDiversity( USHORT );`

Notifications

Use the following information to programmatically control the vibrator, to write an application to turn on the vibrator when a message is received via the WLAN radio link, and turn it off when the user hits a key.

Vibrator support is implemented in the NLED driver as a false LED. The vibrator is LED 5 and is identified with an CycleAdjust of –1. The vibrate option is only available in the notifications panel when the vibrator is present in the system.

Regarding an applications interface to NLED.DLL, LEDs must be available for use by applications. This is possible via two functions exported by the COREDLL.DLL file. To use the LED functions, declare these as extern ”C” as follows:

```
extern ”C” BOOL WINAPI NLEDGetDeviceInfo(UINT nInfoId,
void *pOutput);
extern ”C” BOOL WINAPI NLEDSetDevice( UINT nDeviceId, void
*pInput);
```

The LEDs are enumerated for access through the data structures associated with these APIs:

- Notification LED 0
- Radio On LED 1 *(does not apply to the 730 Computer)*
- Alpha Lock LED 2
- Scanner LED 3
- Low Battery 4
- Vibrator 5 *(does not apply to the 730 Computer)*

NLEDGetDeviceInfo

Usage
#include “nled.h”

Syntax
```
BOOL NLEDGetDeviceInfo ( UINT nInfoId, void *pOutput );
```

Parameters

nInfoId	Integer specifying the information to return. These values are defined:	
	NLED_COUNT_INFO	Indicates the *pOutput* buffer specifies the number of LEDs on the device.
	NLED_SUPPORTS_INFO_ID	Indicates the *pOutput* buffer specifies information about the capabilities supported by the LED.
	NLED_SETTINGS_INFO_ID	Indicates the *pOutput* buffer contains information about the LED current settings.
pOutput	Pointer to the buffer to which the information is returned. The buffer points to various structure types defined in “nled.h”, depending on the value of *nId*, as detailed in the following table:	
	Value of *nID*	**Structure in *pOutput***
	LED_COUNT_INFO	NLED_COUNT_INFO
	NLED_SUPPORTS_INFO	NLED_SUPPORTS_INFO
	NLED_SETTINGS_INFO	NLED_SETTINGS_INFO

NLEDSetDevice

Usage
#include “nled.h”

Syntax
```
BOOL NLEDSetDevice ( UINT nDeviceId, void *pInput );
```

Parameters

nDeviceId	Integer specifying the device identification. The following is defined:	
	NLED_SETTINGS_INFO_ID	Contains information about the desired LED settings.
pInput	Pointer to the buffer that contains the NLED_SETTINGS_INFO structure.	

Reboot Functions

There are several methods, via Kernel I/O Control functions, that an application program can use to force the 700 Series Computer to reboot.

IOCTL_HAL_REBOOT

IOCTL_HAL_REBOOT performs a warm-boot. See page 260.

IOCTL_HAL_COLDBOOT

Invoking the KernelIOControl function with IOCTL_HAL_COLDBOOT forces a cold reboot. This resets the 700 Series Computer and reloads Windows CE as if a power-up had been performed. The contents of the Windows CE RAM-based object store are discarded. See page 257.

IOCTL_HAL_WARMBOOT

This function is supported on the 700 Series Computers. It performs a warm boot of the system, preserving the object store. See page 257.

Remapping the Keypad

Note; Use caution when remapping the keypad. Improper remapping may render the keypad unusable. Data within the 700 Series Computer could also be lost, should any problems occur.

Applications have the ability to remap keys on the 700 Color Numeric Keypad and 700 Color Alphanumeric Keypad. This will allow applications to enable keys that would otherwise not be available, such as the [F1] function key. Also, to disable keys that should not be available, such as the alpha key because no alpha entry is required. Care should be exercised when attempting to remap the keypad because improper remapping may cause the keypad to become unusable. This can be corrected by cold booting the device which will cause the default keymap to be loaded again.

Note that remapping the keys in this way affects the key mapping for the entire system, not just for the application that does the remapping.

There are three "planes" supported for the 700 Color Numeric Keypad and Alphanumeric Keypad. Keys that are to be used in more than one shift plane must be described in each plane.

Unshifted Plane

The unshifted plane contains values from the keypad when not pressed with other keys, such as the following:

Press the Keys		
Numeric Keypad	**Alphanumeric Keypad**	**To Enter This**
1 Caps Send	M1	1
5 JKL A3	T5	5
9 WXYZ	Y9	9

Gold Plane

The gold plane contains values from the keypad when a key is simultaneously pressed with the [**Gold**] key on the numeric keypad or the [**Gold/White**] key on the alphanumeric keypad, such as the following:

Press the Keys		
Numeric Keypad	**Alphanumeric Keypad**	**To Enter This**
[Gold] 1 Caps Send	[Gold/White] Esc Send	Send
[Gold] 5 JKL A3	[Gold/White] C A3	A3
[Gold] 9 WXYZ	[Gold/White] P PgDn	PgDn

Alpha (Blue) Plane

The alpha plane contains values from the keypad when the keypad has been placed in alpha mode by pressing the blue alpha key, such as the following:

Press the Keys		
Numeric Keypad	**Alphanumeric Keypad**	**To Enter This**
[Alpha] Alpha 1 Caps Send	[Alpha] Alpha Cap Lock	Caps
[Alpha] Alpha 5 JKL A3	[Alpha] Alpha J PgUp	j
[Alpha] Alpha 9 WXYZ	[Alpha] Alpha W7	w

Key Values

Key values for each plane are stored in the registry. All units ship with a default key mapping already loaded in the registry. Applications that wish to change the default mapping need to read the appropriate key from the registry into an array of Words, modify the values required and then write the updated values back into the registry. The registry access can be done with standard Microsoft API calls, such as RegOpenKeyEx(), RegQueryValueEx(), and RegSetValueEx().

Numeric Keypad

For the 700 Color Numeric Keypad, the following registry keys contain the plane mappings:

- The unshifted plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\Vkey`
- The gold plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\VkeyGold`
- The alpha plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\VkeyAlpha`

Alphanumeric Keypad

For the 700 Color Alphanumeric Keypad, the following registry keys contain the plane mappings:

- The unshifted plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\ALPHA\Vkey`
- The gold plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\ALPHA\VkeyGold`
- The alpha plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\ALPHA\VkeyAlpha`

How Key Values Are Stored in Registry

To know which fields to update in the registry, you must know what Scan Codes are assigned to each physical key (see the "*Keypad Scan Codes and Meanings"* table on the next page). The Scan Code is used at the lowest level of the system to let the keypad driver know which physical key has been pressed. The keypad driver takes that scan code and looks it up in a table (a copy of the one stored in the registry) to determine which values to pass on to the operating system.

Each registry key is just an array that describes to the keypad driver what value needs to be passed for each physical key. The key values are indexed by the scan code, this is a zero-based index. For example in the unshifted plane, the [4] key has a scan code of 0x06. This means that the seventh word under the "Vkey" registry key will have the value for the [4] key. Taking a sample of the "Vkey" registry key shows the following values:

```
00,00,0B,05,02,03,C1,07,04,03,BE,00,34,00,00,00,. . .
```

The value is 34,00. The values are in reverse byte order because that is the way the processor handles data. When writing an application, nothing needs to be done to swap the bytes, as this will happen automatically when the data is read into a byte value. This is something you just need to be aware of this when looking at the registry. Knowing this, we can see that the value that the keypad driver will pass to the system is a hex 34. Looking that up on an UNICODE character chart, we see that it maps to a "4". If you wanted the key, labeled "4", to output the letter "A" instead, you would need to change the seventh word to "41" (the hexadecimal representation of "A" from the UNICODE chart), then put the key back into the registry.

Note: Do not remap scan codes 0x01, 0x41, 0x42, 0x43, 0x44. Remapping these scan codes could render your 700 Series Computer unusable until a cold-boot is performed.

If you wish to disable a certain key, remap its scan code to 0x00.

Change Notification

Just changing the registry keys will not immediately change the key mappings. To notify the keypad driver that the registry has been updated, signal the "ITC_KEYBOARD_CHANGE" named event using the CreateEvent() API.

Advanced Keypad Remapping

It is also possible to map multiple key presses to one button and to map named system events to a button. The multiple key press option could be useful to cut down on the number of keys needed to press in a given situation or to remap which key behaves like the action key. Mapping events to a button could be useful to change which buttons will fire the scanner, control volume, and allow for suspending and resuming the device. If you need help performing one of these advanced topics please contact Intermec Technical Support.

Scan Codes

At the lowest driver level, the 700 Color Numeric Keypad and the 700 Color Alphanumeric Keypad identifies keys as scan codes. These scan codes are sent via the keypad microcontroller, and cannot be changed without modifying the keypad firmware.

Numeric Keypad

The following scan codes pertain to the 700 Color Numeric keypad:

Numeric Keypad Scan Codes and Meanings

Press this Key	Meaning	ScanCode
	Reserved	0x00
I/O	I/O button	0x01
	Scanner Handle Trigger	0x02
	Scanner Left	0x03
	Scanner Right	0x04
4 GHI A2	4/GHI/A2	0x06
	None	0x07
	Left arrow/Back Tab	0x08
	None	0x09
BkSp /	BkSp// (forward slash)	0x0A
	[Gold] key	0x0B
	None	0x0C
Esc –	Esc/– (minus sign)	0x0D
	Down arrow/Volume decrease	0x0E
1 Caps Send	1/Caps/Send	0x0F
7 PQRS PgUp	7/PQRS/PgUp	0x10
Alpha	[Alpha] key	0x11
	None	0x12
	Up arrow/Volume increase	0x13
	Right arrow/Tab	0x14
2 ABC End	2/ABC/End	0x15
8 TUV *	8/TUV/* (asterisk)	0x16
0 Win	0/Win	0x17
5 JKL A3	5/JKL/A3	0x18
	None	0x19
Action +	Action/+ (plus symbol)	0x1A
3 DEF	3/DEF/backlight	0x1B
9 WXYZ	9/WXYZ/PgDn	0x1C

Numeric Keypad Scan Codes and Meanings (continued)

Press this Key	Meaning	ScanCode
Enter @	Enter/@ (at symbol)	0x1D
6 MNO A4	6/MNO/A4	0x1E
	None	0x1F–0x40
	Charge Detect	0x41
	LCD frontlight	0x42
	Ambient light	0x42
	Threshold crossed	0x42
	Headset detected	0x43
	Keypad Backlight	0x44
	Ambient Light	0x44
	Threshold Crossed	0x44

Alphanumeric Keypad

The following scan codes pertain to the 700 Color Alphanumeric keypad:

Alphanumeric Keypad Scan Codes and Meanings

Press this Key	Meaning	ScanCode
	Reserved	0x00
I/O	I/O button	0x01
	Scanner Handle Trigger	0x02
	Scanner Left	0x03
	Scanner Right	0x04
A A1	A/A1 key	0x05
B A2	B/A2 key	0x06
Esc Send	Escape/Send	0x07
	Left arrow/Back Tab	0x08
	Up arrow/Volume increase	0x09
	Down arrow/Volume decrease	0x0A
	Right arrow/Tab	0x0B
Action End	Action/End	0x0C
E Win	E/Win	0x0D
F =	F/= (equal sign)	0x0E
G *	G/* (asterisk)	0x0F
C A3	C/A3	0x10
H /	H// (forward slash)	0x11
D A4	D/A4	0x12

Alphanumeric Keypad Scan Codes and Meanings (continued)

Press this Key	Meaning	ScanCode
J PgUp	J/PgUp	0x13
K @	K/@ (as symbol)	0x14
L -	L/– (minus sign)	0x15
M 1	M/1	0x16
N 2	N/2	0x17
I ☼	I/backlight	0x18
P PgDn	P/PgDn	0x19
Q ,	Q/, (comma)	0x1A
R +	R/+ (plus sign)	0x1B
S 4	S/4	0x1C
T 5	T/5	0x1D
O 3	O/3	0x1E
Cap Lock	Caps/Lock	0x1F
BkSp	BkSp	0x20
V .	V/. (period)	0x21
W 7	W/7	0x22
X 8	X/8	0x23
U 6	U/6	0x24
	Gold/White	0x25
	NumLock	0x26
Space	Space	0x27
Z 0	Z/0	0x28
Enter	Enter	0x29
Y 9	Y/9	0x2A
	None	0x2B–0x40
	Charge Detect	0x41
	LCD frontlight	0x42
	Ambient light	0x42
	Threshold crossed	0x42
	Headset detected	0x43
	Keypad Backlight	0x44
	Ambient Light	0x44
	Threshold Crossed	0x44

Sample View of Registry Keys

The following is a sample view of the current default key mapping for the 700 Color Numeric Keypad. See the registry on your device for the latest key mappings.

```
[HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD]
"ResumeMask"=dword:7
"Vkey"=hex:00,00,0B,05,02,03,C1,07,04,03,BE,00,34,00,00,00,\
  25,00,00,00,08,00,03,02,00,00,1B,00,28,00,31,00,\
  37,00,01,02,00,00,26,00,27,00,32,00,38,00,30,00,\
  35,00,00,00,01,03,33,00,39,00,0D,00,36,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,07,05,01,05,03,05,02,05

"VkeyGold"=hex: 00,00,0B,05,02,03,C1,07,04,03,BE,00,34,00,00,00,\
  09,01,00,00,BF,00,03,02,00,00,BD,00,75,00,72,00,\
  21,00,01,02,00,00,76,00,09,00,73,00,38,01,5B,00,\
  35,00,00,00,BB,01,09,05,22,00,32,01,36,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,07,05,01,05,03,05,02,05

"VkeyAlpha"=hex:  00,00,0B,05,02,03,C1,07,04,03,BE,00,47,00,00,00,\
  25,00,00,00,08,00,03,02,00,00,1B,00,28,00,02,02,\
  50,00,01,02,00,00,26,00,27,00,41,00,54,00,20,00,\
  4A,00,00,00,01,03,44,00,57,00,0D,00,4D,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,07,05,01,05,03,05,02,05
```

A Configurable Settings

This appendix contains information about the Data Collection, Intermec Settings, SNMP, Unit Information, Utilities, and Wireless Network control panel applets that may be on the 700 Series Color Mobile Computer.

SNMP, Intermec Settings, and Data Collection settings that can appear under **Settings** are dependent on what hardware configuration is done for each 700 Series Computer at the time of shipment. These settings will currently only appear if a scanner or an imager option is present.

Likewise, other control panel applets that are specifically related to the 802.11b radio module will appear when a 802.11b radio module is installed in a 700 Series Computer. Control panel applets that are specific for Wireless Printing, CDMA/1xRTT, and GSM/GPRS radio modules will only appear when each respective hardware configuration is done on the 700 Series Computer. *See Chapter 4, "Network Support," for more information about the radio modules or the wireless printing.*

Information about using reader commands and configuration bar codes to configure some of your settings is also in this appendix.

Note: Information about the settings you can configure with the Intermec Settings control panel applet is described in the *Intermec Computer Command Reference Manual* (P/N: 073529). The online manual is available from the Intermec web site at www.intermec.com.

Configuration Parameters

A configuration parameter changes the way the 700 Series Color Mobile Computer operates, such as configuring a parameter to have the 700 Series Computer emit a very loud beep in a noisy environment. Use any of the following methods to execute configuration parameters:

- Change Data Collection and SNMP parameters via control panel applets later in this appendix.
- Send parameters from an SNMP management station. See "*SNMP Configuration on the 700 Series Computer*" starting on page 191.
- Scan EasySet bar codes. You can use the EasySet bar code creation software from Intermec Technologies Corporation to print configuration labels. Scan the labels to change the scanner configuration and data transfer settings.

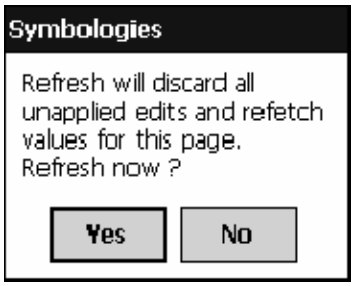
Changing a Parameter Setting

Menus of available parameters for each group are listed. Use the scroll bars to go through the list. Expand each menu (+) to view its parameter settings. Tap a parameter to select, or expand a parameter (+) to view its subparameters.

Note that each parameter or subparameter is shown with its default setting or current setting in (< >) brackets. Tap a parameter or subparameter to select that parameter, then do any of the following to change its setting: Tap **Apply** to apply any changes. *Note that these illustrations are from a Symbologies parameter.*

- Typing a new value in an entry field.
- Choosing a new value from the drop-down list.
- Selecting a different option. The selected option contains a bullet.
- Tap **Defaults**, then **Apply** to restore factory-default settings. Tap **Yes** when you are prompted to verify this action.

- Tap **Refresh** to discard changes and start again. Tap **Yes** when you are prompted to verify this action.



About Configuration Parameters

You can find this information about each configuration parameter:

- **Name and Purpose**:
  Describes the parameter and its function.
- **Action**:
  Describes what to do with a parameter once that parameter is selected.
- **SNMP OID**:
  Lists the SNMP OID for the parameter.
- **Syntax** or **Options**:
  Syntax lists the two-character code for the parameter, if the parameter is configurable by scanning a bar code or by sending parameters through a network. Both **Syntax** and **Options** list acceptable values for the parameter.

Data Collection Control Panel Applet

Note: This applet is not available in units with PSM Build 3.00 or newer. To determine your PSM Build version, tap **Start** > **Programs** > **File Explorer** > the **PSMinfo** text file.

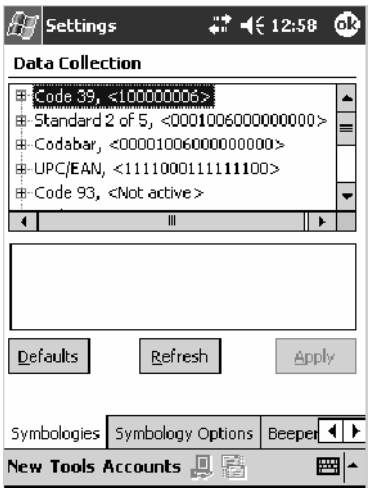
Intermec Settings

If your unit has PSM Build 3.00 or newer, then you may have the Intermec Settings control panel applet in place of the Data Collection applet. Information about the settings you can configure with the Intermec Settings applet is described in the *Intermec Computer Command Reference Manual*. The online manual is available from the Intermec web site at www.intermec.com.

See "*Scanner Control and Data Transfer*" in the *Intermec Windows CE/ Pocket PC Software Developer's Kit (SDK) User's Manual* shipped with the Software Developer's Kit (SDK) for information about data collection functions. *Note that icons are shown to the left.*

Data Collection

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Data Collection** to access its control panel applet.



Use the left and right arrows to scroll through the tabs along the bottom of the control panel applet, then tap a tab to access its menus. These tabs represent the following groups of settings or parameters:

- **Symbologies** *(starting on page 301)*
- **Symbology Options** *(starting on page 322)*
- **Beeper/LED** *(starting on page 330)*
- **Imager** *(starting on page 336)*
- **Virtual Wedge** *(starting on page 341)*

Symbologies

You can change bar code symbology parameter settings in your 700 Series Computer via the **Data Collection** control panel applet. The following parameters are for bar code symbologies. Additional information about the more common bar code symbologies are in Appendix B, "*Bar Codes*." *Note that these parameters are listed in the order of their appearance within this tab.*

Most of these symbologies apply to both the imager and the laser scanner tools. However, when using an imager, the Macro PDF *(page 312)*, Micro PDF417 *(page 314)*, Matrix 2 of 5 *(page 316)*, Telepen *(page 317)*, and Code 11 *(page 318)* symbologies are not supported. Likewise, when using a laser scanner, the QR Code *(page 319)*, Data Matrix *(page 320)*, and MaxiCode *(page 321)* symbologies are not supported.

Note: The 730 Computer uses the EV10 APS linear imager which supports 1D symbologies.

The following table shows which bar code symbologies are supported by an imager, a laser scanner, or the EV10 APS Linear Imager

Bar Code Symbology	Imager	Laser Scanner	EV10 APS Linear Imager
Code 39	X	X	X
Interleaved 2 of 5	X	X	X
Standard 2 of 5	X	X	X
Matrix 2 of 5		X	X
Code 128	X	X	X
Code 93	X	X	X
Codabar	X	X	X
MSI		X	X
Plessey		X	X
UPC	X	X	X
EAN/EAN 128	X	X	X
Code 11		X	X
PDF417	X	X	X
Micro PDF417		X	X
Telepen		X	X
Data Matrix	X		
QR Code	X		
MaxiCode	X		

Code 39

Code 39 is a discrete, self-checking, variable length symbology. The character set is uppercase A–Z, 0–9, dollar sign ($), period (.), slash (/), percent (%), space ( ), plus (+), and minus (-).

Action

Tap (+) to expand the **Code 39** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

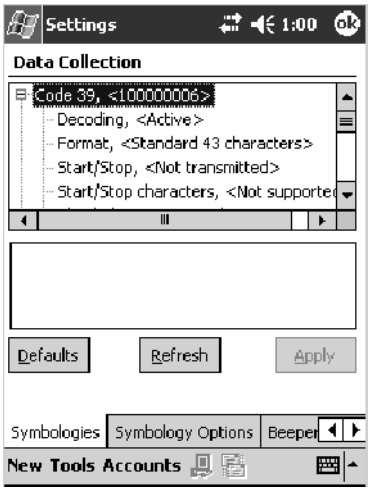
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.3.1

Options

Decoding	0 1	Not active Active (default)
Format	0 1	Standard 43 characters (default) Full ASCII
Start/Stop	0 1	Not transmitted (default) Transmitted
Start/Stop characters *(Not supported when using an imager)*	0 1 2	$ (dollar sign) only * (asterisk) only (default) $ and * (dollar sign and asterisk)
Check digit	0 1 2 3 4 5 6	Not used (default) Mod 43 transmitted Mod 43 not transmitted French CIP transmitted French CIP not transmitted Italian CPI transmitted Italian CPI not transmitted
Bar code length	0 1	Any length (default) Minimum length
Minimum length	001–254	Minimum length 1–254 (default is 6)

Note: If **Bar code length** = "1" then **Minimum length** is entered.



Standard 2 of 5

Standard 2 of 5 is a discrete and self-checking symbology that uses the bars to encode information and the spaces to separate the individual bars.

Action

Tap (+) to expand the **Standard 2 of 5** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

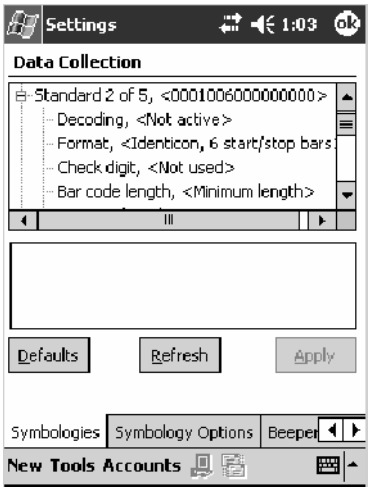
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.4.1

Options

Decoding	0 1	Not active (default) Active
Format	0 1	Identicon, 6 start/stop bars (default) Computer Identics, 4 start/stop bars
Check digit	0 1 2	Not used (default) Mod 10 transmitted Mod 10 not transmitted
Bar code length	0 1 2	Any length Minimum length (default) Fixed lengths
Minimum length	001–254	Minimum length 1–254 (default is 6)
Fixed length 1	000–254	Fixed bar code length 0–254 (default is 0)
Fixed length 2	000–254	Fixed bar code length 0–254 (default is 0)
Fixed length 3	000–254	Fixed bar code length 0–254 (default is 0)

Note: If **Bar code length** = "1" then **Minimum length** is entered. If **Bar code length** = "2" then **Fixed length 1**, **Fixed length 2**, or **Fixed length 3** is entered.



Codabar

Codabar is a self-checking, discrete symbology.

Action

Tap (+) to expand the **Codabar** parameter, select a setting to be changed, then select an option from the drop-down list to change this setting.

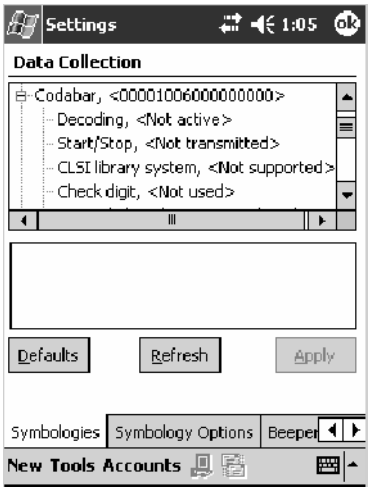
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.5.1

Options

Decoding	0 1	Not active (default) Active
Start/Stop	0 1 2 3 4	Not transmitted (default) abcd transmitted ABCD transmitted abcd/tn*e transmitted DC1–DC4 transmitted
CLSI library system *(Not supported when using an imager)*	0 1	Not active (default) Active
Check digit	0 1 2	Not used (default) Transmitted Not transmitted
Bar code length	0 1 2	Any length Minimum length (default) Fixed lengths
Minimum length	003–254	Minimum length 3–254 (default is 6)
Fixed length 1	000–254	Fixed bar code length 0–254 (default is 0)
Fixed length 2	000–254	Fixed bar code length 0–254 (default is 0)
Fixed length 3	000–254	Fixed bar code length 0–254 (default is 0)

Note: If **Bar code length** = "1" then **Minimum length** is entered. If **Bar code length** = "2" then **Fixed length 1**, **Fixed length 2**, or **Fixed length 3** is entered.



UPC/EAN

UPC/EAN are fixed-length, numeric, continuous symbologies that use four element widths.

Action

Tap (+) to expand the **UPC/EAN** parameter, select the setting to be changed, then select an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.6.1

Options

UPC A	0 1	Not Active Active (default)
UPC E	0 1	Not Active Active (default)
EAN 8	0 1	Not Active Active (default)
EAN 13	0 1	Not Active Active (default)
Add-on digits	0 1	Not required (default) Required
Add-on 2 digits	0 1	Not active (default) Active
Add-on 5 digits *(Not supported when using an imager)*	0 1	Not active (default) Active
UPC A check digit	0 1	Not transmitted Transmitted (default)
UPC E check digit	0 1	Not transmitted Transmitted (default)
EAN 8 check digit	0 1	Not transmitted Transmitted (default)
EAN 13 check digit	0 1	Not transmitted Transmitted (default)
UPC A number system	0 1	Not transmitted Transmitted (default)
UPC E number system	0 1	Not transmitted Transmitted (default)
UPC A re-encoding	0 1	UPC A transmitted as UPC A UPC A transmitted as EAN 13 (default)
UPC E re-encoding	0 1	UPC E transmitted as UPC E (default) UPC E transmitted as UPC A
EAN 8 re-encoding	0 1	EAN 8 transmitted as EAN 8 (default) EAN 8 transmitted as EAN 13

Code 93
Code 93 is a variable length, continuous symbology that uses four element widths.

Action
Tap the **Code 93** parameter, then select an option to change this parameter setting. Tap (+) to access the **Code 93 Lengths** parameter.

SNMP OID
1.3.6.1.4.1.1963.15.3.3.1.1.7.1

Options
0 Not active (default)
1 Active

Code 93 Length
Sets the Code 93 bar code length.

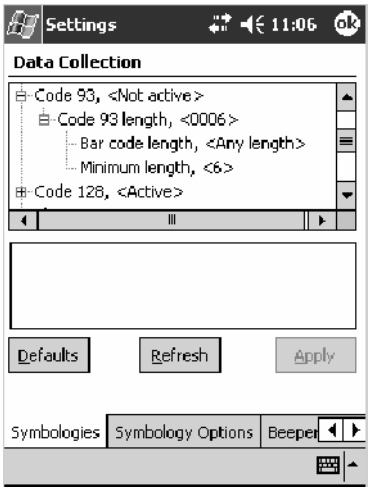
Action
Tap (+) to expand the **Code 93** parameter, then tap (+) to expand the **Code 93 Lengths** parameter. Tap the setting to be changed, then tap an option to change this setting.

SNMP OID
1.3.6.1.4.1.1963.15.3.3.1.1.19.1

Options

Bar code length	0 1	Any length Minimum length (default)
Minimum length	001–254	Minimum length 1–254 (default is 6)

Note: If **Bar code length** = "1" then **Minimum length** is entered.



Code 128

Code 128 is a variable-length, continuous, high-density, alphanumeric symbology that uses multiple element widths and supports the extended ASCII character set.

Action

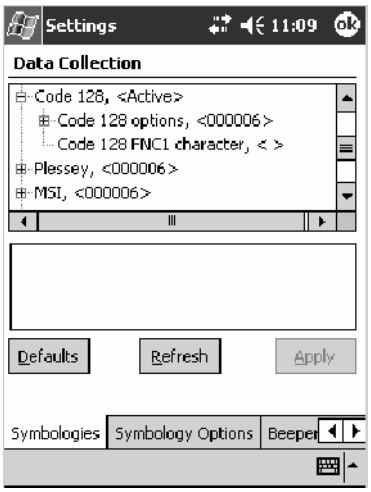
Tap the **Code 128** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.9.1

Options

0 Not active (default)
1 Active



This illustration is from a 700 Series Computer using a laser scanner.

Code 128 Options

Set the following for the Code 128 parameter. *Note that the **EAN 128]C1** and **CIP 128 French Pharmaceutical** options are not available when you use an imager with your 700 Series Computer.*

Action

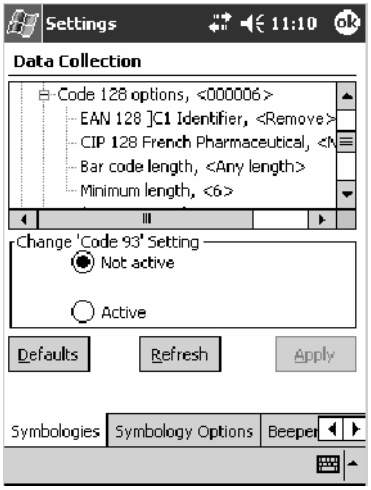
Tap (+) to expand the **Code 128 Options** parameter, select a setting, then select an option to change this setting.

SNMP OID

None.

Options

EAN 128]C1 Identifier *(Not supported when using an imager)*	0 1	Remove (default) Include
CIP 128 French Pharmaceutical *(Not supported when using an imager)*	0 1	Not active (default) Active
Bar code length	0 1	Any length (default) Minimum length
Minimum length	001–254	Minimum length 1–254 (default is 6)



This illustration is from a 700 Series Computer using a laser scanner.

Code 128 FNC1 Character

The Code 128 FNC1 character (EAN 128 norms) can be any ASCII character and is used as a separator when multiple identifiers and their fields are concatenated. *Note that this is not available when you use an imager with your 700 Series Computer.*

Non-printable ASCII characters can be entered using the following syntax where *HH* is the hexadecimal value of the character.

`\xHH`

For example, the GS character, whose hexadecimal value is 1D, would be entered as `\x1D`. In addition,the following characters have their own identifiers:

- BEL \a
- BS \b
- FF \f
- LF \n
- CR \r
- HT \t
- VT \v

Action

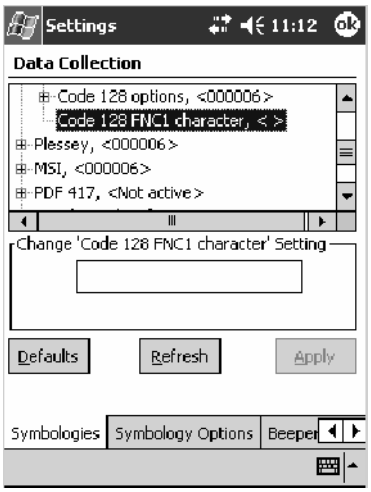
Tap (+) to expand the **Code 128** parameter, then type the ASCII characters to be set for the **Code 128 FNC1 character** parameter.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.21.1

Options

Any ASCII character (default is the GS function character — ID hex)



Plessey

Plessey is a pulse-width modulated symbology like most other bar codes. It includes a start character, data characters, an eight-bit cyclic check digit, and a termination bar. The code is continuous and not self-checking. You need to configure two parameters for Plessey code: Start Code and Check Digit. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

Tap (+) to expand the **Plessey** parameter, select the setting to be changed, then select an option to change this setting or select an option from the drop-down list.

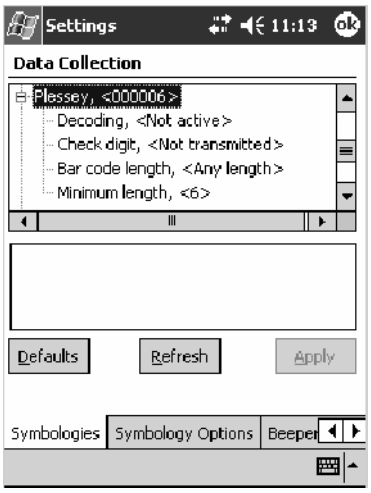
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.10.1

Options

Decoding	0 1	Not active (default) Active
Check digit	0 1	Not transmitted (default) Transmitted
Bar code length	0 1	Any length Minimum length (default)
Minimum length	001–254	Minimum length 1–254 (default is 6)

Note: If **Bar code length** = "1" then **Minimum length** is entered.



MSI

MSI is a symbology similar to Plessey code (page 310) that includes a start pattern, data characters, one or two check digits, and a stop pattern. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

Tap (+) to expand the **MSI** parameter, select the setting to be changed, then select an option to change this setting or select an option from the drop-down list.

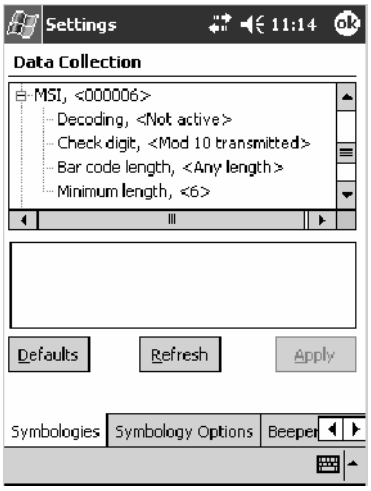
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.15.1

Options

Decoding	0 1	Not active (default) Active
Check digit	0 1 2 3	Mod 10 transmitted (default) Mod 10 not transmitted Double Mod 10 transmitted Double Mod 10 not transmitted
Bar code length	0 1	Any length Minimum length (default)
Minimum length	001–254	Minimum length 1–254 (default is 6)

Note: If **Bar code length** = "1" then **Minimum length** is entered.



PDF417

PDF417 is a stacked two-dimensional symbology that provides the ability to scan across rows of code. Each row consists of start/stop characters, row identifiers, and symbol characters, which consist of four bars and four spaces each and contain the actual data. This symbology uses error correction symbol characters appended at the end to recover loss of data.

Because the virtual wedge translates incoming data into keypad input, the size of the keypad buffer limits the effective length of the label to 128 characters. Longer labels may be truncated. For PDF417 labels of more than 128 characters, you can develop an application that bypasses the keypad buffer.

Action

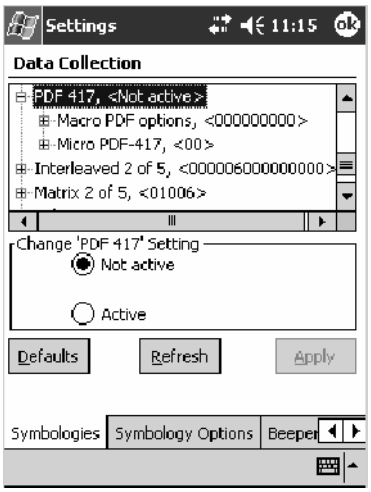
Tap the **PDF417** parameter, then select an option to change this parameter setting. Tap (+) to access either the **Macro PDF options** parameter or the **Micro PDF417** parameter.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.17.1

Options

0 Not active
1 Active (default)



This illustration is from a 700 Series Computer using a laser scanner.

Macro PDF options

Macro PDF is used when a long message requires more than one PDF417 label. *Note that this is not available when you use an imager with your 700 Series Computer.*

- Select **Buffered** to store a multi-label PDF417 message in the Sabre buffer, thus transmitting the entire message when all labels are read.

- Select **Unbuffered** for multi-label PDF417 messages that are too long for the Sabre buffer (memory overflow). Each part of the PDF417 label is transmitted separately, and the host application must then assemble the message using the macro PDF control header transmitted with each label. *Control Header is only present in macro PDF codes and is always transmitted with unbuffered option.*

Action

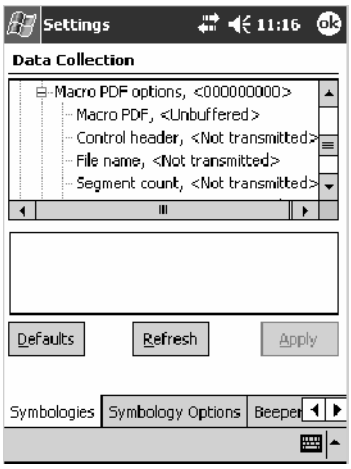
Tap (+) to expand the **PDF417** parameter, tap (+) to expand the **Macro PDF** parameter, select a setting to be changed, then select an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.22.1

Options

Macro PDF	0 1	Unbuffered Buffered (default)
Control header	0 1	Not transmitted (default) Transmitted
File name	0 1	Not transmitted (default) Transmitted
Segment count	0 1	Not transmitted (default) Transmitted
Time stamp	0 1	Not transmitted (default) Transmitted
Sender	0 1	Not transmitted (default) Transmitted
Addressee	0 1	Not transmitted (default) Transmitted
File size	0 1	Not transmitted (default) Transmitted
Checksum	0 1	Not transmitted (default) Transmitted



Micro PDF417

Micro PDF417 is a multi-row symbology derived from and closely based on PDF417 *(page 312)*. A limited set of symbology sizes is available, together with a fixed level of error correction for each symbology size. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

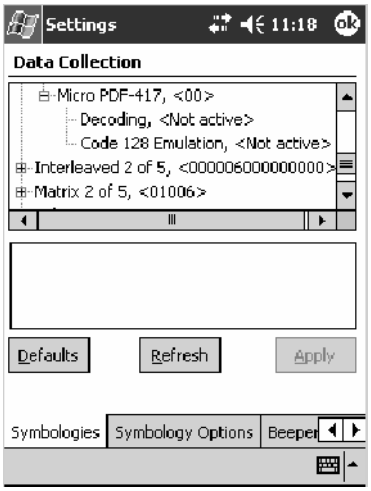
Tap (+) to expand the **PDF417** parameter, tap (+) to expand the **Micro PDF417** parameter, select a setting to be changed, then select an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.27.1

Options

Decoding	0	Not active (default)
	1	Active
Code 128 Emulation	0	Not active (default)
	1	Active



Interleaved 2 of 5

Interleaved 2 of 5 (I 2 of 5) is a high-density, self-checking, continuous, numeric symbology used mainly in inventory distribution and the automobile industry.

Note: An Interleaved 2 of 5 bar code label must be at least three characters long for the 700 Series Computer to scan and decode correctly.

Action

Tap (+) to expand the **Interleaved 2 of 5** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

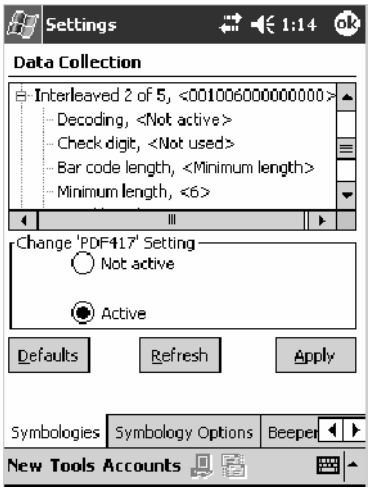
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.23.1

Options

Decoding	0 1	Not active (default) Active
Check digit	0 1 2 3 4	Not used (default) Mod 10 transmitted Mod 10 not transmitted French CIP transmitted French CIP not transmitted
Bar code length	0 1 2	Any length Minimum length (default) Fixed lengths
Minimum length	003–254	Minimum length 3–254 (default is 6)
Fixed length 1	003–254	Fixed bar code length 3–254 (default is 3)
Fixed length 2	003–254	Fixed bar code length 3–254 (default is 3)
Fixed length 3	003–254	Fixed bar code length 3–254 (default is 3)

Note: If **Bar code length** = "1" then **Minimum length** is entered. If **Bar code length** ="2" then **Fixed length 1**, **Fixed length 2**, or **Fixed length 3** is entered.



Matrix 2 of 5

Matrix 2 of 5 is a numerical symbology. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

Tap (+) to expand the **Matrix 2 of 5** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

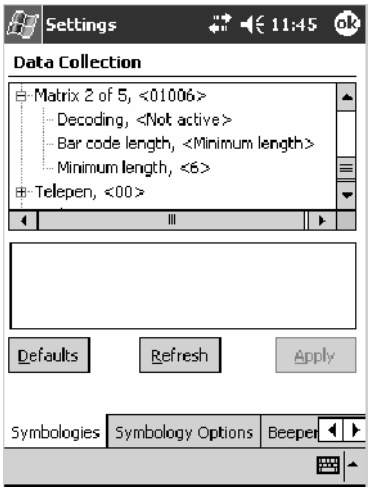
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.24.1

Options

Decoding	0 1	Not active (default) Active
Bar code length	0 1	Any length Minimum length (default)
Minimum length	001–254	Minimum length 1–254 (default is 6)

Note: If **Bar code length** = "1" then **Minimum length** is entered.



Telepen

Telepen is an alphanumeric, case-sensitive, full ASCII symbology. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

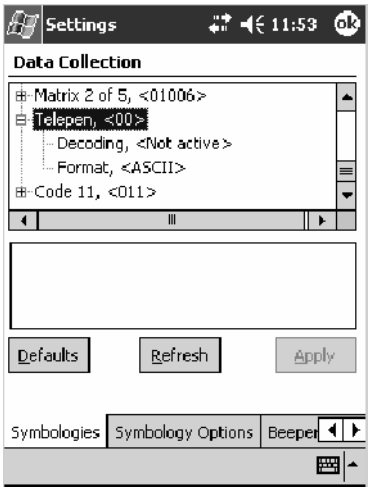
Tap (+) to expand the **Telepen** parameter, select the setting to be changed, then tap an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.25.1

Options

Decoding	0 1	Not active (default) Active
Format	0 1	ASCII (default) Numeric



Code 11

Code 11 is a high density, discrete numeric symbology that is extensively used in labeling telecommunications components and equipment. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

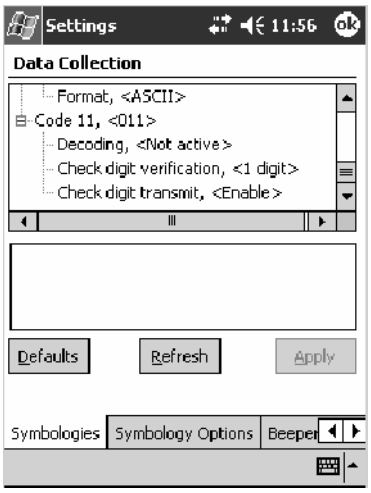
Tap (+) to expand the **Code 11** parameter, select the setting to be changed, then tap an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.26.1

Options

Decoding	0 1	Not active (default) Active
Check digit verification	1 2	1 digit (default) 2 digits
Check digit transmit	0 1	Disable Enable (default)



QR Code

QR Code (Quick Response Code) is a two-dimensional matrix symbology containing dark and light square data modules. It has position detection patterns on three of its four corners and features direct encodation of the Japanese Kana-Kanji character set. It can encode up to 2509 numeric or 1520 alphanumeric characters and offers three levels of error detection. *Note that this is not available when you use a laser scanner with your 700 Series Computer or if you are using a 730 Computer.*

Action

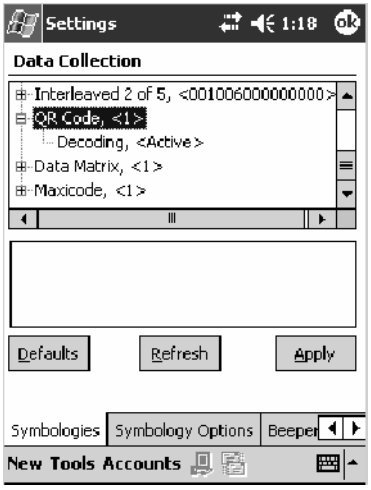
Tap (+) to expand the **QR Code** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.35.1

Options

Decoding	0	Not active
	1	Active (default)



Data Matrix

A two-dimensional matrix symbology, which is made of square modules arranged within a perimeter finder pattern. The symbology utilizes Error Checking and Correcting (ECC) algorithm with selectable levels for data error recovery and Cyclic Redundancy Check algorithm to validate the data. The character set includes either 128 characters conforming to ISO 646 (ANSI X3.4 - 1986) or 256 extended character set. Maximum capacity of a symbol is 2335 alphanumeric characters, 1556 8-bit byte characters or 3116 numeric digits. *Note that this is not available when you use a laser scanner with your 700 Series Computer or if you are using a 730 Computer.*

Action

Tap (+) to expand the **Data Matrix** parameter, select the setting to be changed, then tap an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.34.1

Options

Decoding	0	Not active
	1	Active (default)

MaxiCode

MaxiCode is a fixed-size 2-D matrix symbology which is made up of offset rows of hexagonal elements arranged around a unique circular finder pattern. ASCII data is encoded in six-bit symbol characters. The symbol contains 33 rows which are alternately 30 and 29 elements wide. There are five different code sets. A single MaxiCode symbol can encode up to 93 characters of data. *Note that this is not available when you use a laser scanner with your 700 Series Computer or if you are using a 730 Computer.*

Action

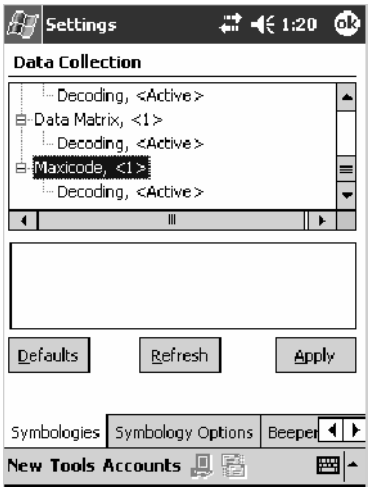
Tap (+) to expand the **MaxiCode** parameter, select the setting to be changed, then tap an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.33.1

Options

Decoding	0	Not active
	1	Active (default)



Symbology Options

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > the **Data Collection** icon to access its control panel applet.

Use the right and left arrows to scroll to the **Symbology Options** tab, then tap this tab to access its parameters. The following are parameters for bar code symbology options. *Note that these are listed in the order of their appearance within the Symbology Options tab.*

Symbology ID

Identifies the bar code symbology in which data is encoded by prepending a user-specified symbology identifier to the data. You can prepend one of these types of character strings to identify the symbology:

- **User-defined ASCII Character** *(Option 1)*:
  A user-defined symbology identifier is a single ASCII character. You can assign a custom identifier character to each bar code symbology. *Note that this is not available when you use an imager with your 700 Series Computer.*
- **AIM ISO/IEC Standard** *(Option 2 — Required to define symbology IDs)*:
  The AIM Standard has a three-character structure which indicates the symbology and optional features. See the *AIM ISO/IEC Standard* for information.

Action

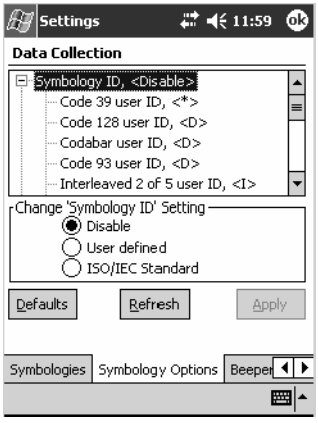
Select **Symbology ID**, then select an option to change this parameter setting. Tap (+) to expand the **Symbology ID** parameter, then select any of the user ID parameters listed. *See the top of the next page for a sample screen of the Code 39 user ID.*

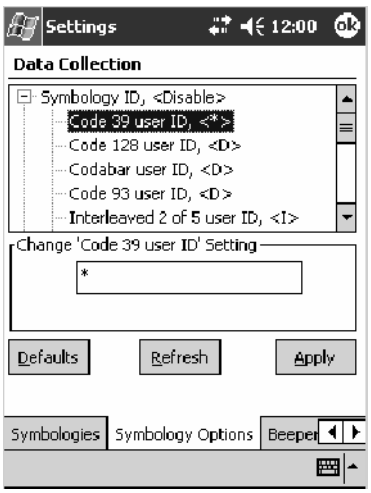
SNMP OID

1.3.6.1.4.1.1963.15.3.3.4.1.22.1

Options

0 Disable (default)
1 User defined *(disabled when using an imager)*
2 ISO/IEC Standard





Code 39 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Code 39 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Code 39 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.3.1

Options: *x* where *x* is a single ASCII character. Default is asterisk (*).

Code 128 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Code 128 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Code 128 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.5.1

Options: *x* where *x* is a single ASCII character. Default is asterisk (*).

Codabar User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Codabar bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Codabar user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.2.1

Options: *x* where *x* is a single ASCII character. Default is D.

Code 93 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Code 93 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Code 93 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.4.1

Options: *x* where *x* is a single ASCII character. Default is asterisk (*).

Interleaved 2 of 5 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Interleaved 2 of 5 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Interleaved 2 of 5 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.10.1

Options: *x* where *x* is a single ASCII character. Default is I (not lowercase L).

PDF417 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify PDF417 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **PDF417 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.12.1

Options: *x* where *x* is a single ASCII character. Default is an asterisk (*).

MSI User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify MSI bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **MSI user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.11.1

Options: *x* where *x* is a single ASCII character. Default is D.

Plessey User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Plessey bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Plessey user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.13.1

Options: *x* where *x* is a single ASCII character. Default is D.

Standard 2 of 5 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Standard 2 of 5 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Standard 2 of 5 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.23.1

Options: *x* where *x* is a single ASCII character. Default is D.

UPC A User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify UPC-A (Universal Product Code) bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **UPC A user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.6.1

Options: *x* where *x* is a single ASCII character. Default is A.

UPC E User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify UPC-E bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **UPC E user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.7.1

Options: *x* where *x* is a single ASCII character. Default is E.

EAN 8 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify EAN-8 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **EAN 8 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.8.1

Options: *x* where *x* is a single ASCII character. Default is \xFF.

EAN 13 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify EAN-13 (European Article Numbering) bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **EAN 13 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.9.1

Options: *x* where *x* is a single ASCII character. Default is F.

Matrix 2 of 5 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Matrix 2 of 5 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Matrix 2 of 5 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.24.1

Options: *x* where *x* is a single ASCII character. *Default is D.*

Telepen User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Telepen bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Telepen user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.25.1

Options: *x* where *x* is a single ASCII character. Default is an asterisk (*).

Code 11 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Code 11 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Code 11 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.16.1

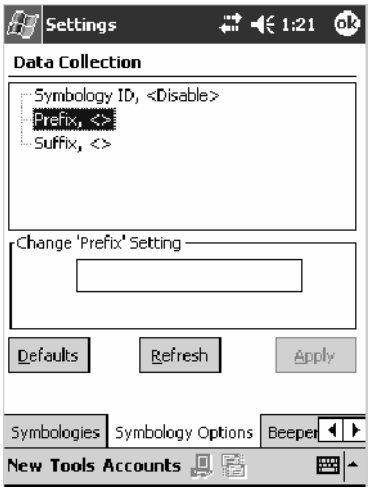
Options: *x* where *x* is a single ASCII character. Default is asterisk (*).

Prefix
Prepends a string of up to 20 ASCII characters to all scanned data.

Action
Tap the **Prefix** parameter, then enter a prefix value to change this parameter setting.

SNMP OID
1.3.6.1.4.1.1963.15.3.3.4.1.29.1

Options
Acceptable values are up to 20 ASCII characters. Embedded null (<NUL >) characters are not allowed. Default is no characters (disabled).

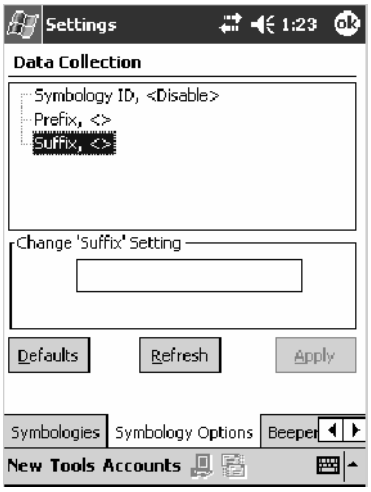


Suffix
Appends a string of up to 20 ASCII characters to all scanned data.

Action
Tap the **Suffix** parameter, then enter a suffix value to change this parameter setting.

SNMP OID
1.3.6.1.4.1.1963.15.3.3.4.1.30.1

Options
Acceptable values are up to 20 ASCII characters. Embedded null (<NUL >) characters are not allowed. Default is no characters (disabled).



Beeper/LED

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > the **Data Collection** icon to access its control panel applet.

Use the right and left arrows to scroll to the **Beeper/LED** tab, then tap this tab to access its parameters.

Most of these functions are not available when using an imager. The following table shows which functions are supported either by an imager or by a laser scanner.

Beeper Function	Imager	Laser Scanner
Beeper	X	
Beeper Volume		X
Beeper Frequency		X
Good Read Beeps		X
Good Read Beep Duration		X

The following are parameters for features on the 700 Series Computer. *Note that these are listed in the order of their appearance.*

Beeper

Sets the volume for the good read beep. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

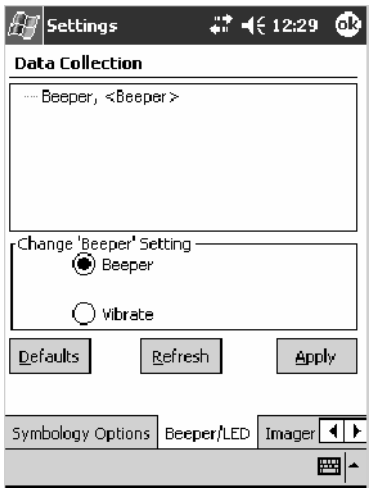
Tap the **Beeper** parameter, then select an option to change this parameter setting.

SNMP OID

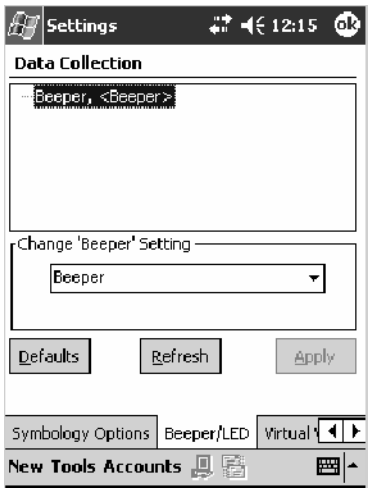
1.3.6.1.4.1.1963.15.3.1.4.1.6.1

Options

1 Beeper (default)
4 Vibrate *(not supported on 730 Computers)*



700 Color with Imager Screen



730 Screen

Beeper Volume

Sets the volume for the good read beep. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

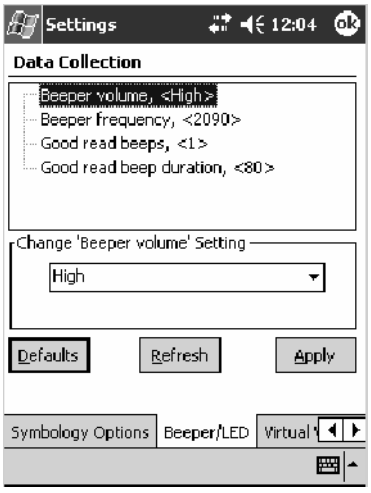
Tap the **Beeper volume** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.1.4.1.6.1

Options

0 Low
1 High (default)
2 Medium
3 Off
4 Vibrate



Disabling the Volume

To disable the beeper, tap **Start** > **Settings** > the **Personal** tab > **Sounds & Notifications** > the **Volume** tab, drag the **System volume** slider bar to the left "Silent" position, then tap **ok** to exit this applet. See Chapter 1, "*Introduction*" for more information.

Beeper Frequency

Sets the frequency for the good read beep. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

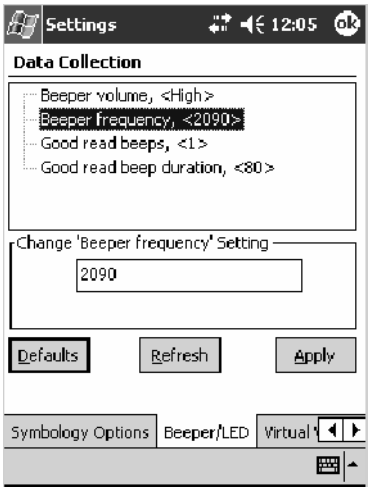
Tap the **Beeper frequency** parameter, then enter a frequency value to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.1.4.1.7.1

Options

1000–4095 (default is 2090)



Good Read Beeps

Sets the number of good read beeps. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

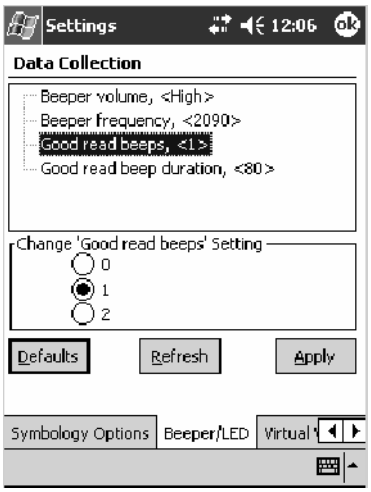
Tap the **Good read beeps** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.1.4.1.8.1

Options

0 No beeps
1 One beep (default)
2 Two beeps



Good Read Beep Duration

Sets the duration of the good read beep. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

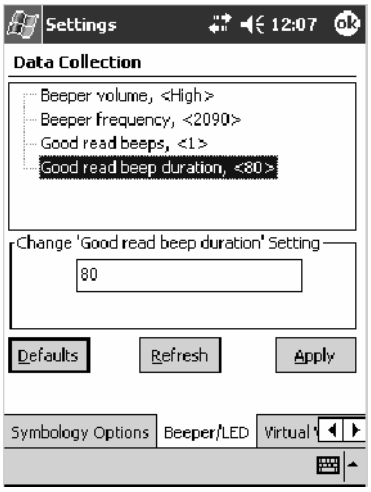
Tap the **Good read beep duration** parameter, then enter a duration value to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.1.4.1.9.1

Options

0–2550 Beep duration in milliseconds. (default is 80)



Imager

Note: These instructions do *not* apply to the 730 Computer.

Data Collection

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Data Collection** to access its control panel applet.

Use the right and left arrows to scroll to the **Imager** tab, then tap this tab to access its parameters.

The following are parameters for the imager. Note that these are listed in the order of their appearance within the Imager tab.

Aimer LED Duration

The Aimer LED Duration controls the time the Aimer LED is turned on when the scan button is pressed. After this time, images are captured for decoding. The purpose is to position the Aimer LED on the bar code symbol before attempting to decode the bar code. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

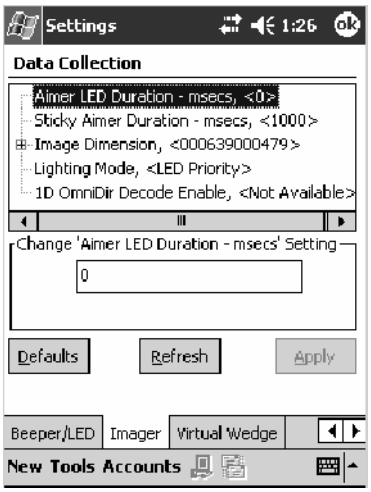
Tap the **Aimer LED Duration** parameter, then enter a value to change this setting. Note that values must be in 50 ms increments, such as 500, 650, or 32500. Values not entered in 50 ms increments are rounded down. For example, 2489 ms is rounded down to 2450 ms, 149 ms is rounded down to 100 ms, etc..

SNMP OID

1.3.6.1.4.1.1963.15.3.3.3.1.1.21.1

Options

0–65500 ms (Default is 0)



Sticky Aimer Duration

The Sticky Aimer Duration controls the time the Aimer LED stays on after the a bar code read completes or after the trigger button is released. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

Tap the **Sticky Aimer Duration** parameter, then enter a value to change this setting. Note that values must be in 50 ms increments, such as 500, 650, or 32500. Values not entered in 50 ms increments are rounded down. For example, 2489 ms is rounded down to 2450 ms, 149 ms is rounded down to 100 ms, etc..

SNMP OID

1.3.6.1.4.1.1963.15.3.3.3.1.1.24.1

Options

0–65535 ms (Default is 1000)

Image Dimension

The image dimensions control the vertical size of the image for decoding. This can restrict the image to one bar code when otherwise, there might be more than one bar code in the image to be decoded. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

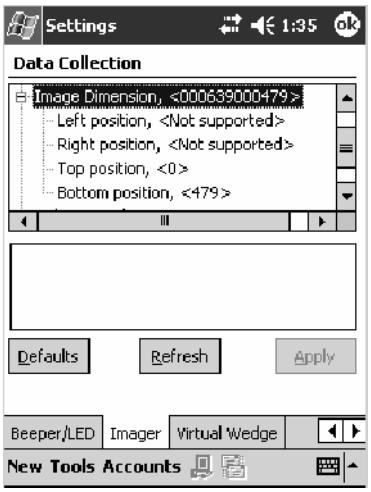
Tap the **Image dimension** parameter, select the position to be changed, then tap an option or enter a value to change this position.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.3.1.1.22.1

Options

Left position	0	Not supported
Right position	0	Not supported
Top position	0–478	Position in pixels (Default is 0)
Bottom position	0–479	Position in pixels (Default is 479)



Lighting Mode

The Lighting Mode sets the lighting mode of the imager. When set to "LED Priority," the imager depends more on ambient lighting to illuminate the bar code for reading. When set to "Aperture Priority," the imager uses its built-in LED to illuminate the bar code for reading. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

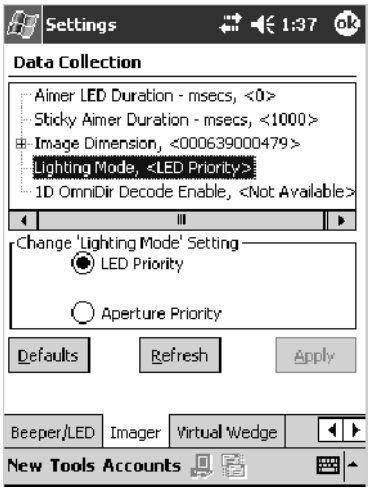
Tap the **Lighting Mode** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.3.1.1.23.1

Options

0 LED Priority (default)
1 Aperture Priority



1D OmniDir Decode Enable

The 1D OmniDir Decode Enable affects the scanning abilities of the IT4000 Imager. With 1D omni directional enabled, the imager is able to decode images and bar code labels regardless of the orientation of the label. With 1D omni directional disabled, the imager only decodes labels in the direction of the aimer LED. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

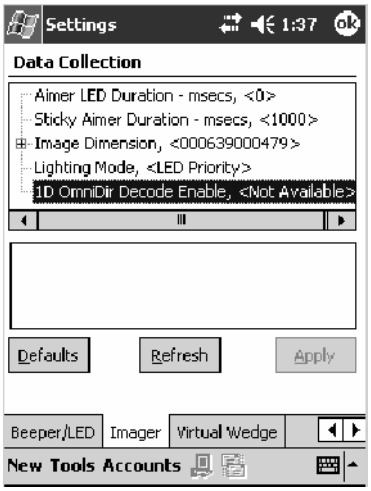
Tap the **1D OmniDir Decode Enable** parameter,then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.3.1.1.25.1

Options

0 Disabled
1 Enabled (default)



Virtual Wedge

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Data Collection** to access its control panel applet.

Use the right and left arrows to scroll to the **Virtual Wedge** tab, then tap this tab to access its parameters.

The following are parameters for the virtual wedge scanner. *Note that these are listed in the order of their appearance within the Virtual Wedge tab.*

Virtual Wedge

Enables or disables the virtual wedge for the internal scanner. The virtual wedge retrieves scanned Automatic Data Collection (ADC) data and sends it to the keypad driver so that the 700 Series Computer can receive and interpret the data as keypad input.

Because the virtual wedge translates incoming data into keypad input, the size of the keypad buffer limits the effective length of the label to 128 characters. Longer labels may be truncated. For labels of more than 128 characters, you need to develop an application that bypasses the keypad buffer.

Action

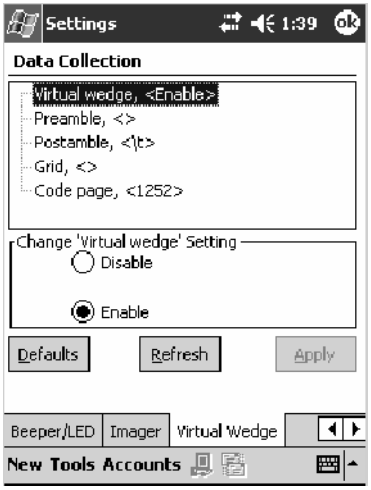
Tap the **Virtual Wedge** parameter, then tap an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.2.1.1.2.1

Options

0 Disable
1 Enable (default)



Preamble

Sets the preamble that precedes any data you scan with the 700 Series Computer. Common preambles include a data location number or an operator number.

Action

Tap the **Preamble** parameter, then enter a preamble value to change this parameter setting.

SNMP OID

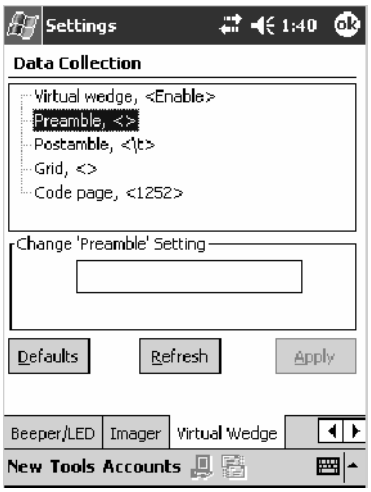
1.3.6.1.4.1.1963.15.3.2.1.1.3.1

Syntax

`ADdata`

where *data* is any acceptable values up to 31 ASCII characters. Embedded null (<NUL >) characters are not allowed. Below are the non-printing characters you can use for Virtual Wedge Preambles. *Default is blank.*

\a	Alert (bell)
\b	Backspace
\f	Form Feed
\n	New line/line feed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\xnnnn	*nnnn* is up to four HEX digits. Use leading zeros to fill out to four digits to ensure proper conversion. For example, to prepend the character M to scanned data, set the Preamble to either 1) M, or 2) x004D, where 4D is the HEX equivalent for an uppercase M.



Note: When you enter the AD command without data, the preamble is disabled. If you want to use quotation marks or the following combinations of characters as part of the appended data, separate those characters from the AD command with quotes. If you do not use quotes as described here, the 700 Series Computer will interpret the characters as another configuration command:
AD
AE
AF
KC
BV
EX
DF

Example

To use the two-character string BV as a preamble, scan this command (as a Code 39 label) or send this command through the network: `$+AD"BV"`

Postamble

Sets the postamble that is appended to any data you scan with the 700 Series Computer. Common postambles include cursor controls, such as tabs or carriage return line feeds.

Action

Tap the **Postamble** parameter, then enter a postamble value to change this parameter setting.

SNMP OID

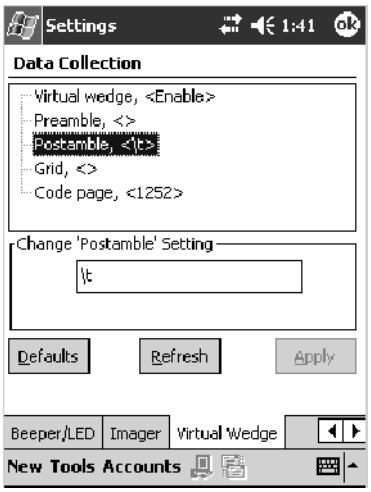
1.3.6.1.4.1.1963.15.3.2.1.1.4.1

Syntax

`AEdata`

where *data* is any acceptable values up to 31 ASCII characters. Embedded null (<NUL >) characters are not allowed. Below are the non-printing characters you can use for Virtual Wedge Postambles:

\a	Alert (bell)
\b	Backspace
\f	Form Feed
\n	New line/line feed
\r	Carriage return
\t	Horizontal tab (default)
\v	Vertical tab
\xnnnn	*nnnn* is up to four HEX digits. Use leading zeros to fill out to four digits to ensure proper conversion. For example, to prepend the character M to scanned data, set the Preamble to either 1) M, or 2) x004D, where 4D is the HEX equivalent for an uppercase M.



Note: When you enter the AE command without data, the postamble is disabled. If you want to use quotation marks or the following combinations of characters as part of the appended data, separate those characters from the AE command with quotes. If you do not use quotes as described here, the 700 Series Computer will interpret the characters as another configuration command.
AD
AE
AF
KC
BV
EX
DF

Example

To use the two-character string BV as a postamble, scan this command (as a Code 39 label) or send this command through the network: `$+AE"BV"`

Grid

Sets the virtual wedge grid, which filters the data coming from this 700 Series Computer. The data server supports data filtering, which allows you to selectively send scanned data. The virtual wedge grid is similar to the "format" argument of the C Runtime Library scan function.

Action

Tap the **Grid** parameter, then enter a grid value to change this parameter setting.

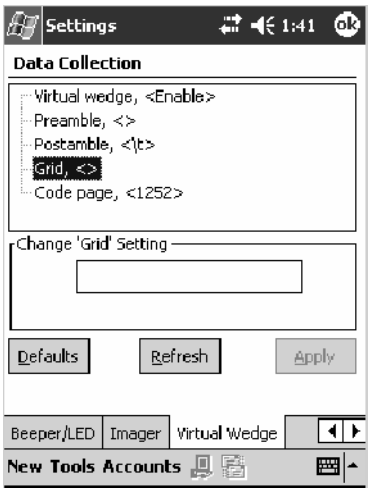
SNMP OID

1.3.6.1.4.1.1963.15.3.2.1.1.5.1

Syntax

`AF<symID> filter-expression= > editing-expression`
where:

- *<symID>*
  The AIM symbology ID *(optional)*.
- *filter-expression*
  Any character string that includes valid filter expression values. Go to the *SDK User's Manual* provided with your *Windows CE/PocketPC SDK* for a list of valid filter expression values.
- *editing-expression*
  Any character string that includes valid editing expression values. Go to the *SDK User's Manual* provided with your *Windows CE/PocketPC SDK* for a list of valid editing expression values.



Code Page

Sets the virtual wedge code page. The code page controls the translation from the character set of the raw collected data to Unicode, which is the character set expected by Windows CE applications. The default code page is 1252, which is the Windows Latin 1 (ANSI) character set.

Action

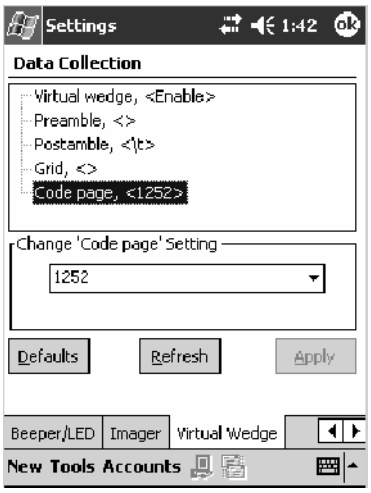
Tap the **Code Page** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.2.1.1.6.1

Options

The only acceptable value for the code page parameter is "*1252*," which is the default.



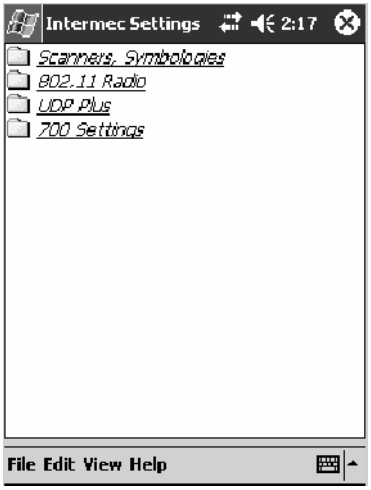
Intermec Settings Control Panel Applet

You may have the Intermec Settings control panel applet. Information about the settings you can configure with this applet is described in the *Intermec Computer Command Reference Manual*. The online manual is available from the Intermec web site at www.intermec.com.

See "*Scanner Control and Data Transfer*" in the *Intermec Windows CE/ Pocket PC Software Developer's Kit (SDK) User's Manual* shipped with the Software Developer's Kit (SDK) for information about data collection functions.

Intermec Settings

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Intermec Settings** to access its control panel applet.



SNMP Control Panel Applet

Note: This applet is not available in units with PSM Build 3.00 or newer. To determine your PSM build version, tap **Start** > **Programs** > **File Explorer** > the **PSMinfo** text file.

Intermec Settings

If your unit has PSM Build 3.00 or newer, then you may have the Intermec Settings control panel applet in place of the SNMP applet. Information about the settings you can configure with the Intermec Settings applet is described in the *Intermec Computer Command Reference Manual*. The online manual is available from the Intermec web site at www.intermec.com.

Simple Network Management Protocol (SNMP) parameters include identification information, security encryption, security community strings, and traps.

SNMP

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **SNMP** to access its control panel applet.

Tap a tab to access its menus. These tabs represent three groups of settings or parameters:

- **Security** *(starting on the next page)*
- **Traps** *(starting on page 355)*
- **Identification** *(starting on page 357)*

Security

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **SNMP** > the **Security** tab to access its parameters.

The following are parameters that affect encryption and community strings. *Note that these are listed in the order of their appearance within the Security tab.*

Read Only Community

Sets the read-only community string for this 700 Series Computer, which is required for processing of SNMP get and get next requests.

Action

Tap the **Read Only Community** parameter, then enter a community string to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.2.0

Options

The read-only community string can be up to 128 ASCII characters. Default is Public.

Read/Write Community

Sets the read/write community string, which is required for processing of SNMP set requests by this 700 Series Computer. An SNMP packet with this name as the community string will also process SNMP get and next requests.

Action

Tap the **Read/Write Community** parameter, then enter a community string to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.3.0

Options

The read/write community string can be up to 128 ASCII characters. Default is Private.

Read Encryption

Sets the packet-level mode of security for SNMP read-only requests. If you enable read encryption, all received SNMP get and get next packets have to be encrypted or the packet will not be authorized. If encryption is enabled, you can only use software provided by Intermec Technologies.

Note: To enable security encryption, you also need to set the Security Encryption Key (page 354).

Action

Tap the **Read Encryption** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.4.0

Options

1	On	SNMP get and get next packets must be encrypted
2	Off	SNMP packets do not have to be encrypted (default)

Write Encryption

Sets the packet-level mode of security for SNMP read/write requests. If you enable write encryption, all SNMP packets that are received with the read/write community string have to be encrypted or the packet will not be authorized. You need to use software from Intermec Technologies that supports encryption.

Note: To enable security encryption, you also need to set the Security Encryption Key (page 354).

Action

Tap the **Write Encryption** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.5.0

Options

1	On	SNMP packets must be encrypted
2	Off	SNMP packets do not have to be encrypted (default)

Encryption Key

Identifies the key that this 700 Series Computer uses to encrypt or decipher SNMP packets. Encryption is used only by software provided by Intermec Technologies. If encryption is enabled, SNMP management platforms will not be able to communicate with the 700 Series Computer. The encryption key is returned encrypted.

Action

Tap the **Encryption Key** parameter, then enter a security encryption key value to change this parameter setting.

Note: You also need to set either **Read Encryption** (page 352) or **Write Encryption** (page 353) or both.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.6.0

Options

The encryption key can be from 4 to 20 ASCII characters. Default is NULL.

Traps

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **SNMP** > the **Traps** tab to access its parameters.

The following are authentication and threshold parameters for traps. *Note that these are listed in the order of their appearance within the Traps tab.*

Authentication

Determines whether to send authentication traps. When trap authentication is enabled, an authentication trap is sent if an SNMP packet is received by the master agent with an invalid community string.

Action

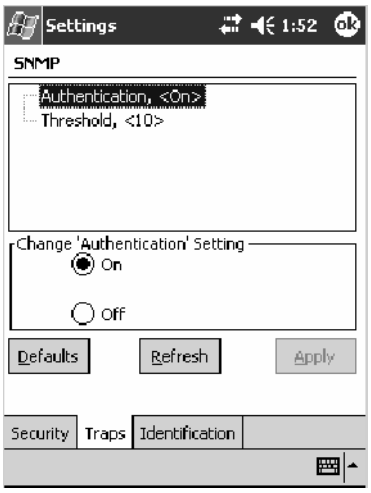
Tap the **Authentication** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.2.2.0

Options

1 On (default)
2 Off

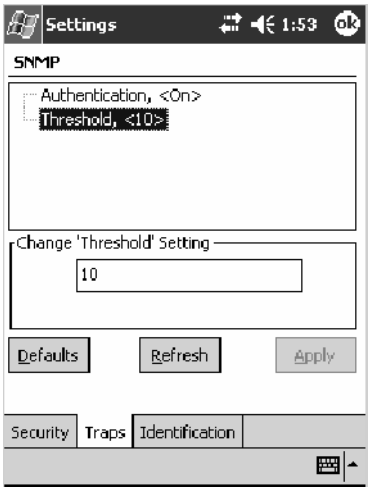


Threshold
Determines the maximum number of traps per second that the master agent generates. If the threshold is reached, the trap will not be sent.

Action
Tap the **Threshold** parameter, then enter a threshold value to change this parameter setting.

SNMP OID
1.3.6.1.4.1.1963.10.5.2.3.0

Options
Any positive integer value. Default is 10.



Identification

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **SNMP** > the **Identification** tab to access its parameters.

The following are parameters for contact, location, and name information for support purposes. *Note that these are listed in the order of their appearance within the Identification tab.*

Contact

Sets the contact information for the person responsible for this 700 Series Computer.

Action

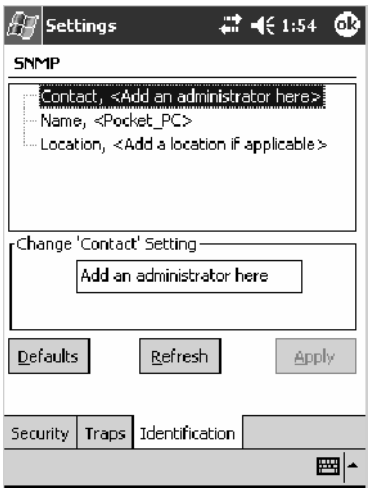
Tap the **Contact** parameter, then enter the name of your contact representative to change this parameter setting.

SNMP OID

1.3.6.1.2.1.1.4.0

Options

The identification contact may be up to 255 ASCII characters. Default is no characters or blank.



Name
Sets the assigned name for this 700 Series Computer.

Action
Tap the **Name** parameter, then enter the name of your 700 Series Computer to change this parameter setting.

SNMP OID
1.3.6.1.2.1.1.5.0

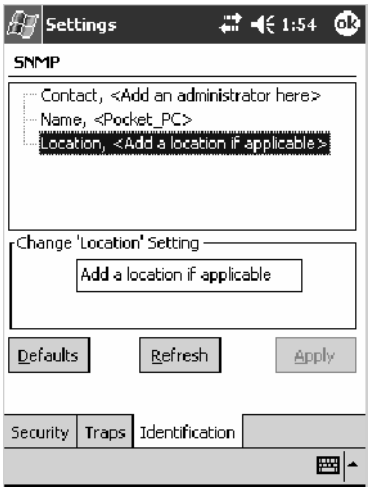
Options
The identification name may be up to 255 ASCII characters. Default is no characters or blank.

Location
Sets the identification location for this 700 Series Computer, such as "Shipping."

Action
Tap the **Location** parameter, then enter the location of where your 700 Series Computer to change this parameter setting.

SNMP OID
1.3.6.1.2.1.1.6.0

Options
The identification location may be up to 255 ASCII characters. Default is no characters or blank.



Unit Information Control Panel Applet

Note: This applet is not available in units with PSM Build 3.00 or newer. To determine your PSM build version, tap **Start** > **Programs** > **File Explorer** > the **PSMinfo** text file.

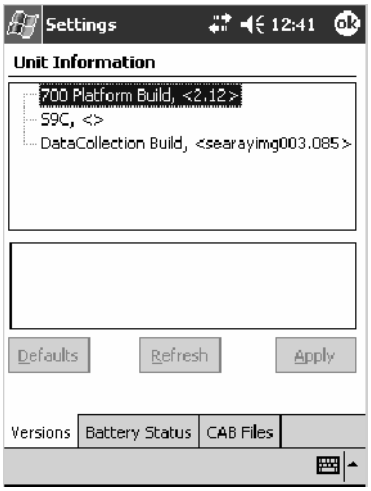
Intermec Settings

I*f your unit has PSM Build 3.00 or newer*, then you may have the Intermec Settings control panel applet in place of the Unit Information applet. Information about the settings you can configure with the Intermec Settings applet is described in the *Intermec Computer Command Reference Manual*. The online manual is available from the Intermec web site at www.intermec.com.

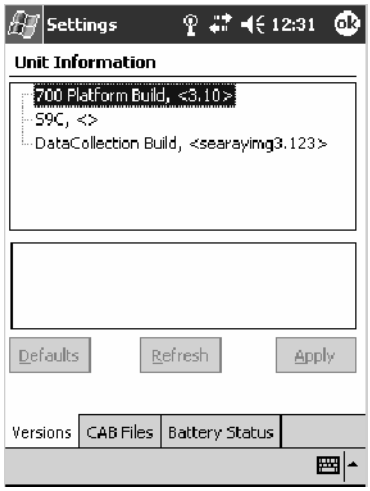
Unit Information is a read-only control panel applet that provides information about your 700 Series Computer, such as software version builds, available CAB files, and the internal battery status.

Unit Information

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Unit Information** to access its control panel applet.



700 Color Screen



730 Screen

Tap a tab to access its menus. These tabs represent three groups of settings or parameters:

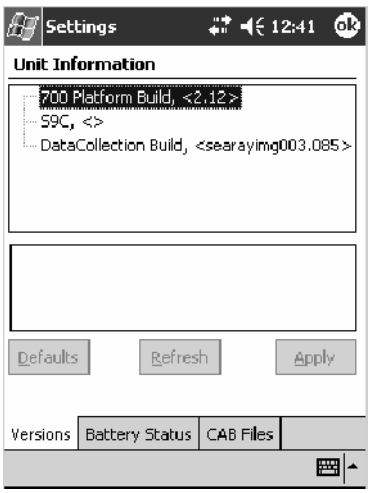
- **Versions** *(starting on the next page)*
- **Battery Status** *(starting on page 362)*
- **CAB Files** *(starting on page 363)*

Versions

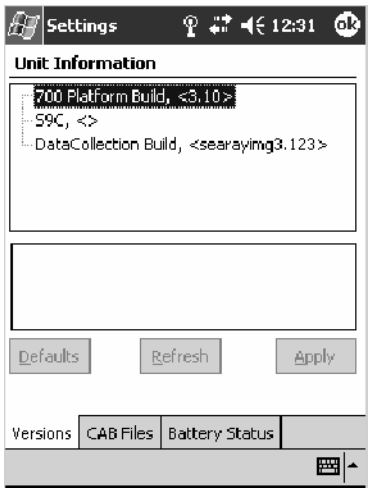
You can view the latest software build version on your 700 Series Computer by accessing the **Unit Information** control panel applet.

Unit Information

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Unit Information** > the **Versions** tab to view the latest software build version. Tap **ok** to exit this information.



700 Color Screen



730 Screen

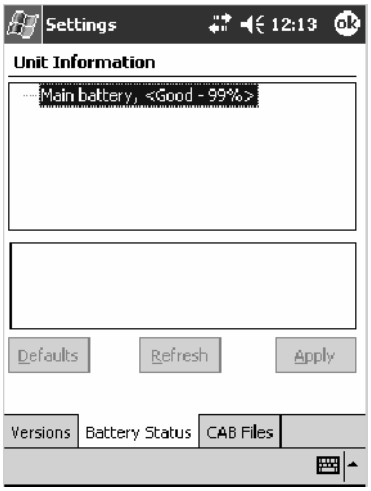
Below are some of the software applications you may find on this screen:

- **700 Platform Build**:
  Shows the latest development or released version of the software build for the 700 Series Computer.
- **S9C**:
  Provides the name and version of the scanner file built into this 700 Series Computer, along with the current CPU version.
- **DataCollection Build**:
  Shows the latest development or released version of the software build for the Data Collection control panel applet.

Battery Status

You can view the battery status for your 700 Series Computer by accessing the **Unit Information** control panel applet.

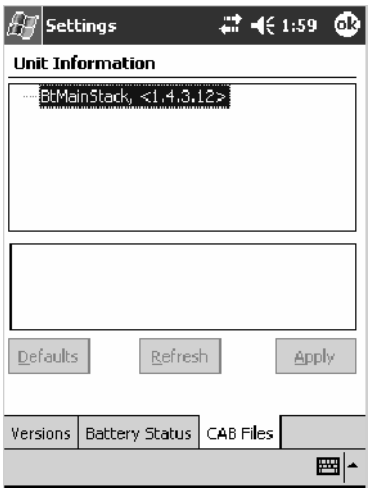
To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Unit Information** > the **Battery Status** tab to view the current status. Tap **ok** to exit this information.



CAB Files

You can view the latest developer or released version of each CAB file from Intermec Technologies Corporation that are installed in your 700 Series Computer via the **Unit Information** control panel applet. *Custom CAB files are not displayed in this applet.* See the *Software Tools User's Manual* for more information about these files.

To access the information from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Unit Information** > the **CAB Files** tab to view the current CAB file versions. Tap **ok** to exit this information.



When a CAB file is built, a registry entry is created with a build number for that file. This CAB Files control panel applet looks for a registry key for each CAB file installed. When the registry entry is found, the CAB file name and version number information are displayed. If a CAB file has not been installed, then its information is not displayed.

Below is a list of CAB files from Intermec Technologies that are available for your 700 Series Computer with their latest developer or released version of the software build. Should you need to add any of these to your 700 Series Computer, contact an Intermec representative.

- **BtMainStack**:
  Installation of the Main Bluetooth Stack is handled automatically as part of the operating system boot-up procedure. *See Chapter 4, "Network Support," for more information about Bluetooth wireless printing.*
- **Comm Port Wedge**:
  The software build for the Comm Port Wedge. *Note that the Comm Port Wedge CAB file is available on the Intermec Developer's Library CD.*
- **NPCPTest**:
  This installs a Norand® Portable Communications Protocol (NPCP) Printing test application which will print to an Intermec® 4815, 4820, or 6820 Printer. *See Chapter 5, "Printer Support," for more information.*

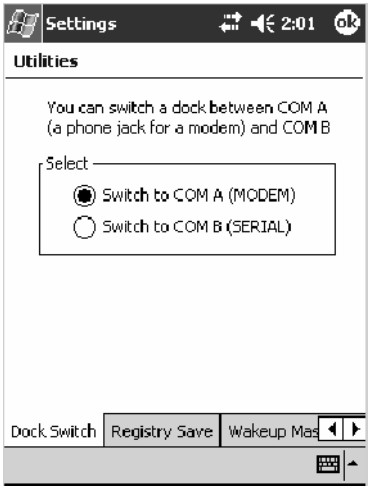
- **S9C Upgrade**:
  Installs the files needed to upgrade the S9C scanner firmware. *See the 700 Color Management Tools portion of the Intermec Developer's Library CD for more information about upgrading the firmware.*
- **SDK**:
  Installs the Intermec Software Developer's Kit (SDK). *See the SDK User's Manual for more information.*
- **WinCfg**:
  Configures the NRINET.INI file, launches the NRINet client, and loads and unloads the LAN and WLAN device drivers.
- **Wireless Printing Sample**:
  Installs a sample application that developers can use for reference when they are developing their own Wireless Printing applications. The source code for this application is included as part of the Wireless Printing SDK on the *Intermec Developer's Library CD*. *See the SDK User's Manual for more information.*
- **ActiveX Control Tools**:
  This lists some of the CAB files that may be available with which to install ActiveX Control Tools. *See the SDK Online Help for more information.*
  - **AXCommunication**:
    Communication controls that transmit or receive messages from input connections.
  - **AXFileTransfer**:
    File transfer controls that transmit and receive files using the Trivial File Transfer Protocol (TFTP).
  - **AXReaderCommand**:
    Reader command functions that modify and retrieve configuration information from your 700 Series Computer.
  - **AXVWedge**:
    The virtual wedge control that retrieves scanned ADC data and sends it to the keyboard driver to interpret data as keyboard input.

Utilities Control Panel Applet

The Utilities control panel applet examines and modifies settings and operational modes of specific hardware and software on the 700 Color Computer, including the dock switch, registry storage, wakeup mask, and application launch keys.

Utilities

To access the settings from the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Utilities** to access its control panel applet.



Use the left and right arrows to scroll through the tabs along the bottom of the control panel applet, then tap a tab to access its menus. These tabs represent the following groups of settings or parameters:

- **Dock Switch** *(next page)*
- **Registry Save** *(page 367)*
- **Wakeup Mask** *(page 368)*
- **App Launch** *(page 369)*

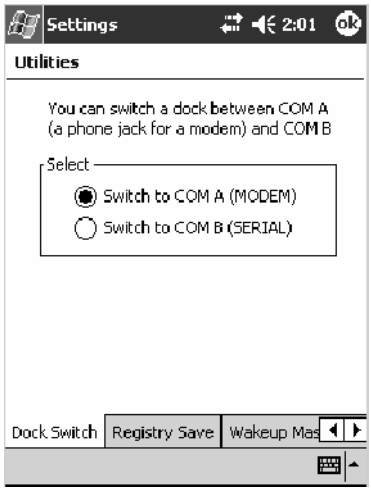
Dock Switch

From the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Utilities** > the **Dock Switch** tab to access the Dock Switch control panel applet.

Use this applet to control the position of the dock switch. This can be set either to a COM A (phone jack for a modem) position or to a COM B (serial) position.

If switched to COM B and suspended the terminal will have the following behavior:

- If the 700 Series Computer is on charge, the dock switch will remain switched to COM B.
- If the 700 Series Computer is off charge, the dock switch will switch to COM A and remain in this position until the 700 Series Computer resumes charge.

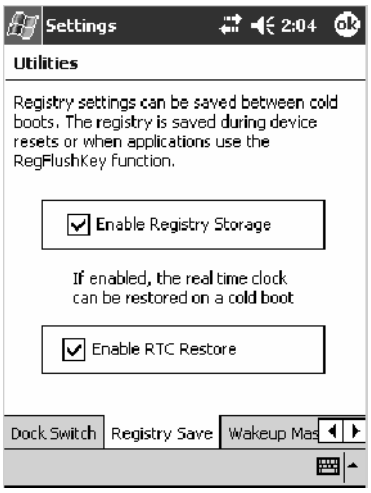


Registry Save

From the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Utilities** > the **Registry Save** tab to access the Registry Save control panel applet.

For Windows Mobile 2003, the only medium available for saving the registry is the Flash File System (PSM). Registry data is stored in the "\Flash_File_Store\Registry" path. Check **Enable Registry Storage** to enable this function.

To ensure that the 700 Series Computer restores the real-time clock after a cold-boot, check the **Enable RTC Restore** option. *Note that this does not apply to the 730 Computer.*



700 Color or 700CXL Screen

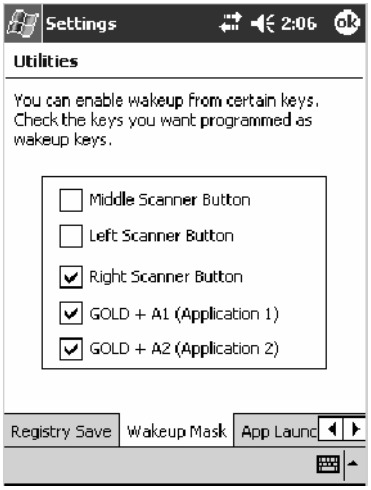
730 Screen

Wakeup Mask

From the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Utilities** > the **Wakeup Mask** tab to access the Wakeup Mask control panel applet.

This applet programs three scanner buttons and the A1 and A2 application keys to be "wakeup" or resume keys. That is, to prompt the 700 Series Computer to "wake up" or resume activity after going to "sleep" as a result of being inactive after a length of time. This information will remain between warm and cold boots.

Check the appropriate box, then tap **ok** to apply your settings.



Based on your setting, do the following to "wake up" the 700 Series Computer.

If you select:	Then do this on Numeric Keyboard	Then do this on Alphanumeric Keyboard
Middle Scanner Button	Squeeze the button on the Scan Handle	Squeeze the button on the Scan Handle
Left Scanner Button	Squeeze the left scanner button	Squeeze the left scanner button
Right Scanner Button	Squeeze the right scanner button	Squeeze the right scanner button
GOLD + A1 (Application 1)	Press [**Gold**]	Press [**Gold/White**]
GOLD + A2 (Application 2)	Press [**Gold**]	Press [**Gold/White**]

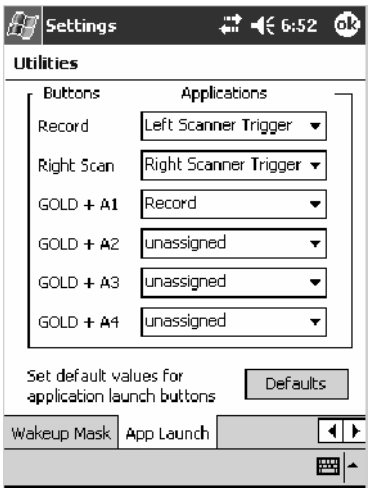
App Launch

From the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Utilities** , then scroll to the right to tap the **App Launch** tab to access the Application Launch control panel applet.

This applet programs or maps two scanner buttons and four application keys to start up to six applications. *Note that the left scanner button also acts as the record button.*

For 700 Series Computers with either a laser scanner or an imager, applications are as follows and default mappings are shown in the following illustration:

- Left Scanner Trigger
- Right Scanner Trigger
- Record *(see Note)*
- Calendar *(see Note)*
- Contacts *(see Note)*
- Tasks *(see Note)*



For 700 Series Computers without either a laser scanner or an imager, the default maps the Record, Calendar, Contacts, and Tasks applications the top four and the A3 and A4 buttons are ”unassigned” or available for two more applications.

Note: Record, Calendar, Contacts, and Tasks are Pocket PC applications. See Chapter 2, “*Windows Mobile 2003*,” for more information about these applications.

- To assign an application to a button, select an application from the applicable drop-down list box.
- To assign a new application, select the “Add new application” option, which brings up an Open File dialog and browse SD or CF storage cards for new applications.

- To disable or unmap a currently mapped application from a corresponding button, select "unassigned" from the applicable drop-down list.
- To restore these buttons to their defaults, tap **Defaults** in the lower right corner.

Note; You cannot map an application to more than one button. Should you assign the same application to two buttons, a verification prompt will appear after the second button to confirm whether you want to remap the application. If you tap **Yes**, the applet changes the first button to "unassigned" and map the application to the second button.

Note: All changes are activated immediately upon selection.

Wireless Network Control Panel Applet

Note: See Chapter 4, "*Network Support*," for information about the 802.11b radio module.

About the Wireless Network

Your wireless adapter (network interface card) connects to wireless networks of two types: access point networks and peer-to-peer networks.

- Access point networks get you onto your corporate network and the internet. Your 700 Series Computer establishes a wireless connection to an access point, which links you to the rest of the network. When you connect to a network via an access point, you are using the 802.1x infrastructure mode.
- Peer-to-peer networks are private networks shared between two or more people, even with no access point.

Each wireless network is assigned a name (or Service Set Identifier - SSID) to allow multiple networks to coexist in the same area without infringement.

With multiple networks within range of each other, security is a necessity, to avoid outside eavesdropping. To make a network safe, the following are required:

- Authentication by both the network and the user
- Authentication is cryptographically protected
- Wireless connection is encrypted

There are two basic mechanisms for providing secure encryption over a wireless network: preconfigured secrets called WEP keys and authentication using the 802.1x protocol.

Terminology

Below are terms you may encounter as you configure your wireless network:

- **CKIP** (Cisco Key Integrity Protocol)
  This is a light version of the TKIP protocol developed by Cisco.
- **EAP** (Extensible Authentication Protocol)
  802.1x uses this protocol to perform authentication. This is not necessarily an authentication mechanism, but is a common framework for transporting actual authentication protocols. Intermec Technologies provides a number of EAP protocols for you to choose the best for your network.

- **TKIP** (Temporal Key Integrity Protocol)
  This protocol is part of the IEEE 802.11i encryption standard for wireless LANs., which provides per-packet key mixing, a message integrity check and a re-keying mechanism, thus fixing the flaws of WEP.

  This protocol provides stronger encryption than WEP, by dynamically updating the encryption keys every 10,000 packets. It eliminates attacks on WEP, which is based on a cryptographic algorithm called RC4, leaving WLANs open to various security attacks.

  TKIP is an interim addition to WEP that addresses some security concerns. It provides per-packet key mixing, a MIC (Message Integrity Check), and a rekeying mechanism designed to fix WEP flaws.

- **WEP** (Wired Equivalent Privacy) **encryption**
  With preconfigured WEP, both the client 700 Series Computer and access point are assigned the same key, which can encrypt all data between the two devices. WEP keys also authenticate the 700 Series Computer to the access point — unless the 700 Series Computer can prove it knows the WEP key, it is not allowed onto the network.

  WEP keys are only needed if they are expected by your clients. There are two types available: 64-bit (5-character strings, 12345) (default) and 128-bit (13-character strings, 1234567890123). Enter these as either ASCII (12345) or Hex (0x3132333435).

- **WPA** (Wi-Fi Protected Access)
  This is an enhanced version of WEP that does not rely on a static, shared key. It encompasses a number of security enhancements over WEP, including improved data encryption via TKIP and 802.1x authentication with EAP.

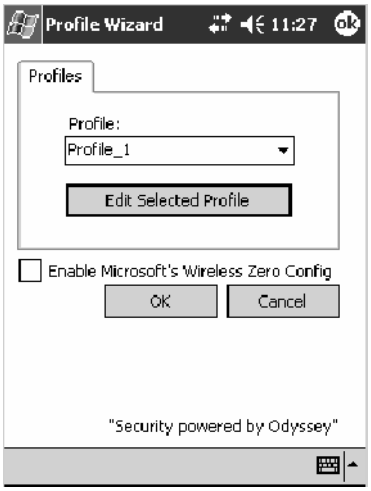
Configuring Your Wireless Network

To start 802.11b communications on the 700 Series Computer, tap **Start** > **Settings** > the **System** tab > **Wireless Network** to access the Profile Wizard for the 802.11b radio module.

A profile contains all the information necessary to authenticate you to the network, such as login name, password or certificate, and protocols by which you are authenticated.

You can have up to four profiles for different networks. For example, you may have different login names or passwords on different networks, or you may use a password on one network, and a certificate on another.

Use the Profiles page to select and configure between the networking environments assigned to this 802.11b radio.

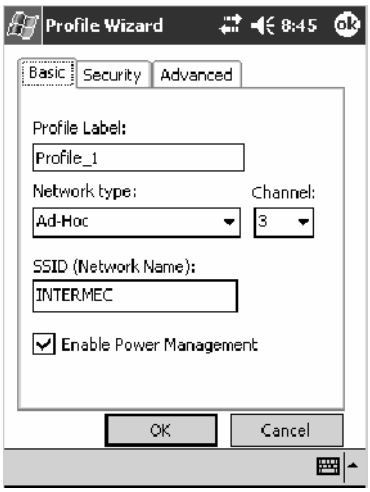


- **Profile**:
  Tap the drop-down list to choose between four different profiles assigned to this unit, then tap **Edit Select Profile**, make the changes needed for this profile *(starting on the next page)*, then tap **OK** to return to the Profiles page.
- **Enable Microsoft's Wireless Zero Config**
  Check this box to enable Microsoft's Wireless Zero Config application. This effectively disables the Intermec software solution for 802.11b, including configuration via the CORE application and the Wireless Network control panel applet.

Basic

Use the Basic page to set the network type, name, and manage battery power for this profile. Tap **ok** or **OK** to return to the Profiles page.

- **Profile Label**:
  Enter a unique name for your profile.
- **Network type**:
  Tap the drop-down list to select either "Infrastructure" if your network uses access points to provide connectivity to the corporate network or internet; or "Ad-Hoc" to set up a private network with one or more participants.
- **Channel**:
  If you selected "Ad-Hoc" for the network type, select the channel on which you are communicating with others in your network. There are up to 11 channels available.
- **SSID (Network Name)**:
  This assumes the profile name *unless another name is entered in this field*. If you want to connect to the next available network or are not familiar with the network name, enter "ANY" in this field. Consult your LAN administrator for network names.
- **Enable Power Management**:
  Check this box to conserve battery power (default), or clear this box to disable this feature.



Security

The following are available from the **8021x Security** drop-down list. *Note that the last four methods are available if you have purchased the security package. Contact your Intermec representative for information.*

- None *(next page)*
- PEAP *(page 379)*
- TLS *(page 383)*
- TTLS *(page 387)*
- LEAP *(page 390)*

None

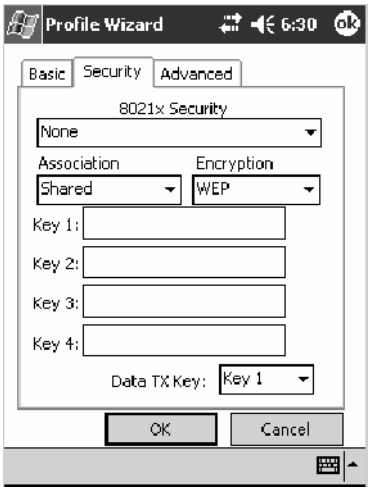
Use "None" to disable 802.1x Security and enable either WEP or WPA-PSK encryption.

To Disable 802.1x Security

1. Set **8021x Security** as "None."
2. Set **Association** to "Open."
3. Set **Encryption** to "None."

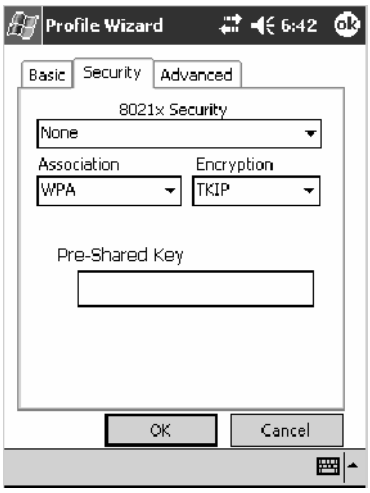
To Enable WEP Encryption

1. Set **8021x Security** as "None."
2. Set **Association** to either "Open" if WEP keys are not required; or "Shared" when WEP keys are required for association.
3. Set **Encryption** to "WEP." See page 372 for information about WEP encryption.
4. If you had set **Association** to "Shared," then select a data transmission key from the **Data TX Key** drop-down list near the bottom of this screen, then enter the encryption key for that data transmission in the appropriate **Key #** field.



To Enable WPA Encryption Using a Preshared Key

1. Set **8021x Security** as "None."
2. Set **Association** to "WPA." See page 372 for information about WPA encryption.
3. Skip **Encryption** as it is automatically set to "TKIP." See page 372 for more information about TKIP.
4. Enter the temporal key as ASCII (12345) in the **Pre-Shared Key** field.



PEAP (Protected EAP)

This protocol is suitable for performing secure authentication against Windows domains and directory services. It is comparable to EAP-TTLS (see page 387), both in its method of operation and its security, though not as flexible. This does not support the range of inside-the-tunnel authentication methods supported by EAP-TTLS. Microsoft and Cisco both support this protocol.

Use "PEAP" to configure the use of PEAP as an authentication protocol and to select "Open," "WPA," or "Network EAP" as an association mode.

To Enable PEAP with an Open Association

1. Set **8021x Security** as "PEAP."
2. Set **Association** to "Open."
3. Skip **Encryption** as it is automatically set to "WEP." See page 372 for information about WEP encryption.
4. Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol; or leave **Use following password** as selected to automatically use the protocol without entering a password.
5. Tap **Additional Settings** to assign an inner PEAP authentication and set options for server certificate validation and trust. See page 382 for more information.

To Enable PEAP with WPA Encryption

1. Set **8021x Security** as "PEAP."
2. Set **Association** to "WPA." See page 372 for information about WPA encryption.
3. Skip **Encryption** as it is automatically set to "TKIP." See page 372 for more information about TKIP.
4. Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
5. Tap **Additional Settings** to assign an inner PEAP authentication and set options for server certificate validation and trust. See page 382 for more information.

To Enable PEAP with Network EAP

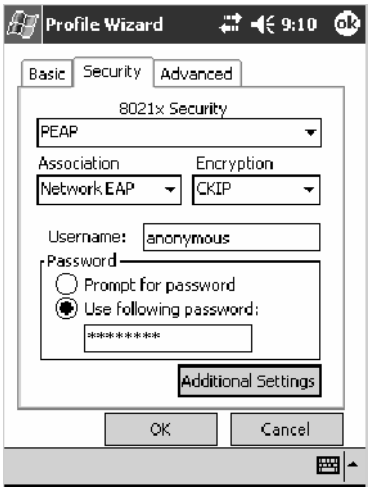
1 Set **8021x Security** as "PEAP."

2 Set **Association** to "Network EAP." See page 371 for information about EAP.

3 Set **Encryption** to either "WEP" or "CKIP." See page 371 for information about CKIP and page 372 for information about WEP encryption.

4 Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.

5 Tap **Additional Settings** to assign an inner PEAP authentication and set options for server certificate validation and trust. See page 382 for more information.



Additional Settings

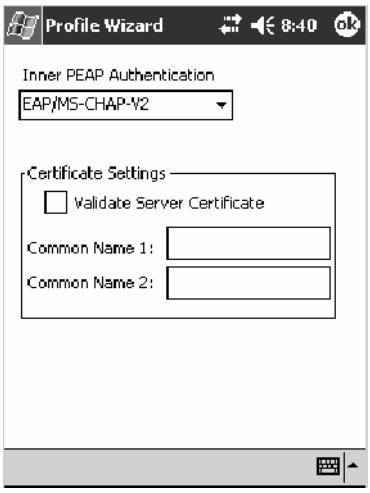
1 Select an authentication method from the **Inner PEAP Authentication** drop-down list.

EAP/MS-CHAP-V2	Authenticates against a Windows Domain Controller and other non-Windows user databases. This is Microsoft's implementation of PEAP.
EAP/Token Card	Use with token cards. The password value entered is never cached. This is Cisco's implementation of PEAP.
EAP/MD5-Challenge	Message Digest 5. A secure hashing authentication algorithm.

2 Check **Validate Server Certificate** to verify the identity of the authentication server based on its certificate when using TTLS, PEAP, and TLS.

3 Enter the **Common Names** of trusted servers. *Note that if these fields are left blank, the server certificate trust validation is not performed or required.*

4 Click **ok** to return to the Security page.



TLS (EAP-TLS)

EAP-TLS is a protocol that is based on the TLS (Transport Layer Security) protocol widely used to secure web sites. This requires both the user and authentication server have certificates for mutual authentication. While cryptically strong, this requires corporations that deploy this to maintain a certificate infrastructure for all their users.

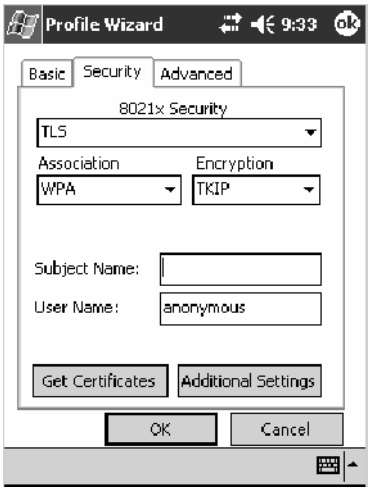
Use "TLS" to configure the use of EAP-TLS as an authentication protocol, and select either "Open" or "WPA" as an association mode.

To Enable TLS with an Open Association

1. Set **8021x Security** as "TLS."
2. Set **Association** to "Open."
3. Skip **Encryption** as it is automatically set to "WEP." See page 372 for information about WEP encryption.
4. Enter your unique **Subject Name** and **User Name** to use this protocol.
5. Tap **Get Certificates** to obtain or import server certificates. See page 385 for more information.
6. Tap **Additional Settings** to set options for server certificate validation and trust. See page 386 for more information.

To Enable TLS with WPA Encryption

1. Set **8021x Security** as "TLS."
2. Set **Association** to "WPA." See page 372 for information about WPA encryption.
3. Skip **Encryption** as it is automatically set to "TKIP."See page 372 for more information about TKIP.
4. Enter your unique **Subject Name** and **User Name** as credentials for this profile.
5. Tap **Get Certificates** to obtain or import server certificates. See page 385 for more information.
6. Tap **Additional Settings** to set options for server certificate validation and trust. See page 386 for more information.



To Get Certificates

Certificates are pieces of cryptographic data that guarantee a public key is associated with a private key. They contain a public key and the entity name that owns the key. Each certificate is issued by a certificate authority.

Use this page to import a certificate onto the 700 Series Computer.

Root Certificates

1 Tap the <<< button next to the **Import Root Certificate** field to select the root certificate (DER-encoded .CER file) to import.

2 Click **Import Root Cert** to install the selected certificate.

User Certificate

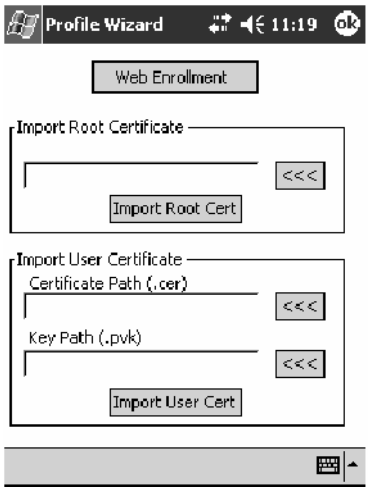
1 Tap the <<< button next to the **Certificate Path** field to select the user certificate (DER-encoded .CER file without the private key) to import.

2 Tap the <<< button next to the **Key Path** field to select the private key (.PVK file) which corresponds to the user certificate chosen in step 1.

3 Tap **Import User Cert** to install the selected certificate.

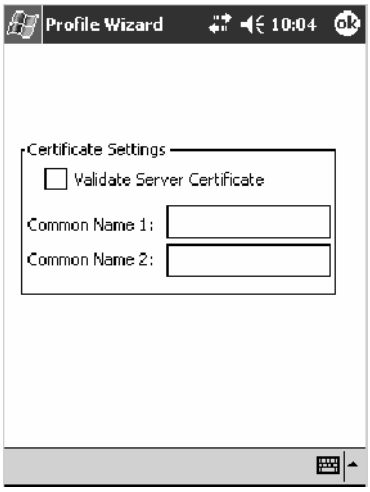
Web Enrollment

Tap **Web Enrollment** to obtain a user certificate over the network from an IAS Server. Tap **ok** to return to the Security page.



Additional Settings

1. Check **Validate Server Certificate** to verify the identity of the authentication server based on its certificate when using TTLS, PEAP, and TLS.
2. Enter the **Common Names** of trusted servers. *Note that if these fields are left blank, the server certificate trust validation is not performed or required.*
3. Click **ok** to return to the Security page.



TTLS (EAP-Tunneled TLS)

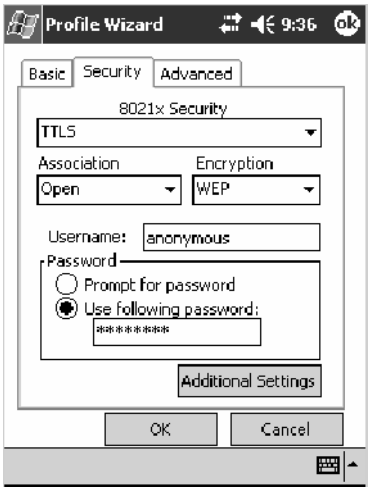
This protocol provides authentication like EAP-TLS (see page 383) but does not require certificates for every user. Instead, authentication servers are issued certificates. User authentication is done using a password or other credentials that are transported in a securely encrypted "tunnel" established using server certificates.

EAP-TTLS works by creating a secure, encrypted tunnel through which you present your credentials to the authentication server. Thus, inside EAP-TTLS there is another *inner authentication protocol* that you must configure via Additional Settings.

Use "TTLS" to configure the use of EAP-TTLS as an authentication protocol, and select either "Open" or "WPA" as an association mode.

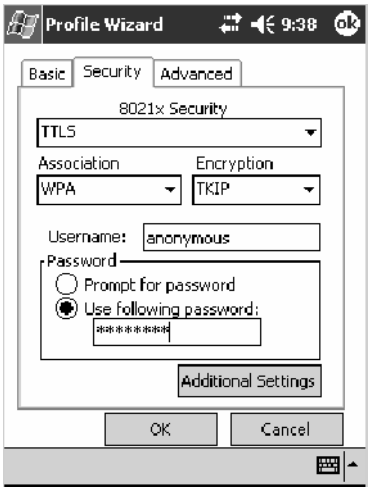
To Enable TTLS with an Open Association

1. Set **8021x Security** as "TTLS."
2. Set **Association** to "Open."
3. Skip **Encryption** as it is automatically set to "WEP." See page 372 for information about WEP encryption.
4. Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
5. Tap **Additional Settings** to assign an inner TTLS authentication and an inner EAP, and set options for server certificate validation and trust. See page 389 for more information.



To Enable TTLS with WPA Encryption

1. Set **8021x Security** as "TTLS."
2. Set **Association** to "WPA." See page 372 for information about WPA encryption.
3. Skip **Encryption** as it is automatically set to "TKIP." See page 372 for more information about TKIP.
4. Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
5. Tap **Additional Settings** to assign an inner TTLS authentication and an inner EAP, and set options for server certificate validation and trust. See page 389 for more information.



Additional Settings

1 Select an authentication protocol from the **Inner TTLS Authentication** drop-down list:

PAP	Password Authentication Protocol. A simple authentication protocol that sends security information in the clear.
CHAP	Challenge Handshake Authentication Protocol. Use of Radius to authenticate a terminal without sending security data in the clear. Authenticates against non-Windows user databases. *You cannot use this if authenticating against a Windows NT Domain or Active Directory.*
MS-CHAP; MS-CHAP-V2	Authenticates against a Windows Domain Controller and other non-Windows user databases.
PAP/Token Card	Use with token cards. The password value entered is never cached.
EAP	Extensible Authentication Protocol. See page 371 for information about EAP.

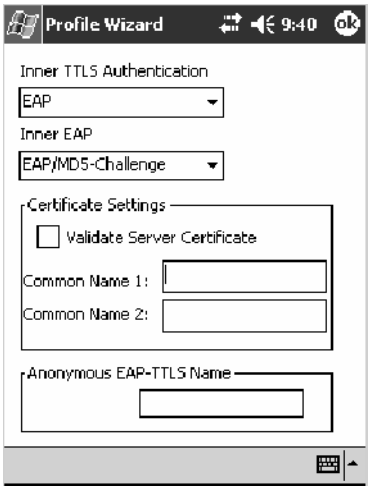
2 If you select "EAP" for the inner authentication protocol, then an inner EAP protocol from the **Inner EAP** drop-down list.

3 Enter the **Common Names** of trusted servers. *Note that if these fields are left blank, the server certificate trust validation is not performed or required.*

4 Check **Validate Server Certificate** to verify the identity of the authentication server based on its certificate when using TTLS, PEAP, and TLS.

5 Enter the **Anonymous EAP-TTLS Name** as assigned for public usage. Use of this outer identity protects your login name or identity.

6 Click **ok** to return to the Security page.



LEAP (Cisco Lightweight EAP)

LEAP is the Cisco Lightweight version of EAP. See page 371 for information about EAP.

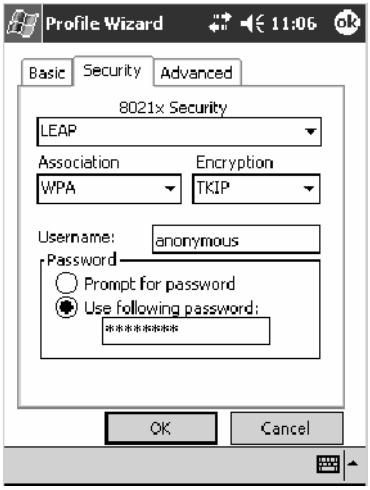
Use "LEAP" to configure the use of LEAP as an authentication protocol, select "Open," "WPA," or "Network EAP" as an association mode, or assign Network EAP. *Note that this defaults to the Network EAP.*

To Enable LEAP with an Open Association

1. Set **8021x Security** as "LEAP."
2. Set **Association** to "Open."
3. Skip **Encryption** as it is automatically set to "WEP." See page 372 for information about WEP encryption.
4. Enter your unique **User Name** to use this protocol.
5. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.

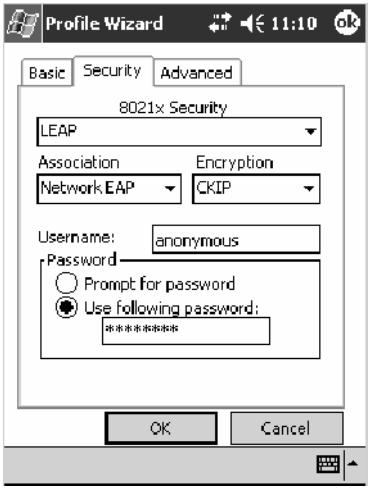
To Enable LEAP with WPA Encryption

1. Set **8021x Security** as "LEAP."
2. Set **Association** to "WPA." See page 372 for information about WPA encryption.
3. Skip **Encryption** as it is automatically set to "TKIP." See page 372 for more information about TKIP.
4. Enter your unique **User Name** to use this protocol.
5. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.



To Enable LEAP with Network EAP

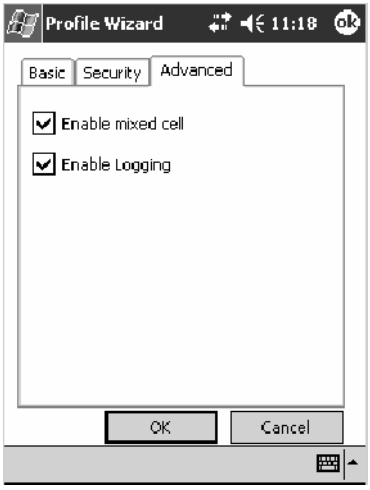
1. Set **8021x Security** as "LEAP."
2. Set **Association** to "Network EAP," an EAP protocol for the network. See page 371 for information about EAP.
3. Set **Encryption** to either "WEP" or "CKIP." See page 371 for information about CKIP and page 372 for information about WEP encryption.
4. Enter your unique **User Name** to use this protocol.
5. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.



Advanced

Use this page to configure additional settings for this profile. Tap **ok** or **OK** to return to the Profiles page.

- **Enable mixed cell**:
  Mixed cell is a profile-dependent setting. If enabled, you can connect to mixed cell without using WEP, then you can query the cell to determine whether you can use encryption.
- **Enable Logging**:
  Check this to log what activity incurs for this profile.



Other Configurable Parameters

The following parameters can be configured by sending reader commands through the network or from an application. See "*Using Reader Commands*" on page 396 for more information.

Audio Volume

Changes the volume of all audio signals.

SNMP OID
1.3.6.1.4.1.1963.15.3.1.3.0

Options (Syntax Data for Reader Commands)
0 Off
1 Very quiet
2 Quiet
3 Normal (default)
4 Loud
5 Very loud

Automatic Shutoff

Sets the length of time the 700 Series Computer remains on when there is no activity. When you turn on the 700 Computer, it either resumes exactly where it was when you turned it off or boots and restarts your application.

SNMP OID
1.3.6.1.4.1.1963.15.11.3.0

Options (Syntax Data for Reader Commands)
1 1 minute
2 2 minutes
3 3 minutes (default)
4 4 minutes
5 5 minutes

Backlight Timeout

Sets the length of time that the display backlight remains on. If you set a longer timeout value, you use the battery power at a faster rate.

SNMP OID

1.3.6.1.4.1.1963.15.13.1.0

Options (Syntax Data for Reader Commands)

10	10 seconds
30	30 seconds
60	1 minute (default)
120	2 minutes
180	3 minutes
240	4 minutes
300	5 minutes

Date/Time

Sets the current date and time.

SNMP OID

Date: 1.3.6.1.4.1.1963.15.501.2.1.0
Time: 1.3.6.1.4.1.1963.15.501.2.2.0

Options (Syntax Data for Reader Commands)

Date	Year	0000–9999 *(1999)*
	Month	1–12 *(6)*
	Day	1–31 *(1)*
Time	Hour	0–23 *(0)*
	Minute	0–59 *(00)*
	Second	0–59 *(00)*

Key Clicks

Enables or disables the keypad clicks. The 700 Series Computer emits a click each time you press a key or decode a row of a two-dimensional symbology.

SNMP OID

1.3.6.1.4.1.1963.15.12.1.0

Options (Syntax Data for Reader Commands)

0	Disable clicks
1	Enable soft key clicks
2	Enable loud key clicks (default)

Using Reader Commands

After the 700 Series Computer is connected to your network, you can send the 700 Series Computer a reader command from an application to perform a task, such as changing the time and date. Some reader commands temporarily override the configuration settings and some change the configuration settings.

Change Configuration

The Change Configuration command must precede any configuration command. If you enter a valid string, the 700 Series Computer configuration is modified and the computer emits a high beep. To send the Change Configuration command through the network, use the `$+ [command]` syntax where *command* is the two-letter command syntax for the configuration command followed by the value to be set for that command.

You can also make changes to several different commands by using the `$+ [command]...[command n]` syntax. There are seven configuration command settings that can be changed in this way. *See each command for information on respective acceptable "data" values.*

Command	Syntax
Audio Volume	BV*data*
Automatic Shutoff	EZ*data*
Backlight Timeout	DF*data*
Key Clicks	KC*data*
Virtual Wedge Grid	AF*data*
Virtual Wedge Postamble	AE*data*
Virtual Wedge Preamble	AD*data*

Note: See pages 342 and 344 for more information about the **Virtual Wedge Postamble** and **Virtual Wedge Preamble** commands.

Example 1

To change the Beep Volume to Off, you can send this string to the 700 Series Computer through the network: `$+BV0`

where:

`$+`	Indicates Change Configuration.
`BV`	Specifies the Audio Volume parameter.
`0`	Specifies a value of Off.

Example 2

To change the Beep Volume to Very Quiet and the Virtual Wedge Grid to 123: `$+BV1AF123`

where:

`$+`	Indicates Change Configuration
`BV1`	Specifies Audio Volume, set to Very Quiet (1)
`AF123`	Specifies Virtual Wedge Grid, set to a value of 123.

Set Time and Date

This command sets the date and time on the 700 Series Computer. The default date and time is *June 1, 1999 at 12:00 AM*.

From the network, send the following:

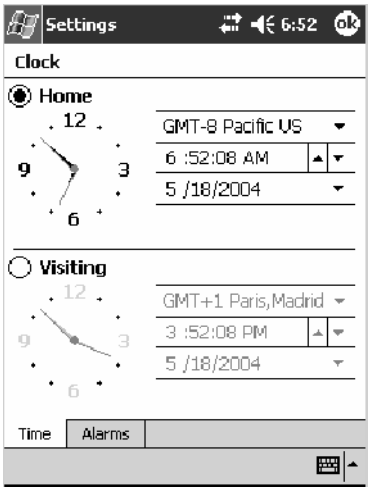
```
/+ yyyymmddhhmmss
```

where acceptable values for the date are:

`yyyy`	0000–9999	Year
`mm`	01–12	Month of the year
`dd`	01–31	Day of the month
`hh`	00–23	Hour
`mm`	00–59	Minutes
`ss`	00–59	Seconds

Clock

You can also set the time and date by using Configuration Management in Unit Manager, or by using the **Clock** control panel applet in the Settings menu. To access this control panel applet, tap **Start** > **Settings** > the **System** tab > the **Clock** icon to access its control panel applet.



Configuration Bar Codes

You can change some settings on your 700 Series Computer by scanning the following Code 39 bar code labels.

- You can use the Data Collection control panel to set the three Virtual Wedge parameters *(starting on page 341)*.

Note: When you use a bar code creation utility to make a scannable bar code label, the utility probably adds opening and closing asterisks automatically. Asterisks are included here for translation purposes.

Audio Volume

Note: The Audio Volume parameter information is on page 394.

Turn Audio Off

$+BV0

Set Audio Volume to very quiet

$+VB1

Set Audio Volume to quiet

$+BV2

Set Audio Volume to normal *(default)*

$+BV3

Set Audio Volume to loud

$+BV4

Set Audio Volume to very loud

$+BV5

Automatic Shutoff

Note: The Automatic Shutoff parameter information is on page 394.

Set Automatic Shutoff to 1 minute

$+EZ1

Set Automatic Shutoff to 2 minutes

$+EZ2

Set Automatic Shutoff to 3 minutes *(default)*

$+EZ3

Set Automatic Shutoff to 4 minutes

$+EZ4

Set Automatic Shutoff to 5 minutes

$+EZ5

Backlight Timeout

Note: The Backlight Timeout parameter information is on page 395.

Backlight Timeout 10 seconds

$+DF10

Backlight Timeout 30 seconds

$+DF30

Backlight Timeout 1 minute *(default)*

$+DF60

Backlight Timeout 2 minutes

$+DF120

Backlight Timeout 3 minutes

$+DF180

Backlight Timeout 4 minutes

$+DF240

Backlight Timeout 5 minutes

$+DF300

Key Clicks

Note: The Key Clicks parameter information is on page 395.

Disable key clicks

$+KC0

Enable soft key clicks

$+KC1

Enable loud key clicks *(default)*

$+KC2

Virtual Wedge Grid, Preamble, Postamble

The following parameters are user-configurable strings. Refer to a full ASCII chart for more information.

Grid

For Virtual Wedge Grid, the first part of the bar code would be the following, which can include a string of up to 240 characters. *Parameter information starts on page 346.*

*$+AF

Preamble

For Virtual Wedge Preamble, the first part of the bar code would be below, followed by a string of up to 31 characters *(no <NUL>)* and an asterisk. *Default is no characters. Parameter information is on page 342.*

*$+AD

Postamble

For Virtual Wedge Postamble, the first part of the bar code would be below, followed by a string of up to 31 characters *(no <NUL>)* and an asterisk. *Default is no characters. Parameter information is on page 344.*

*$+AE

B Bar Code Symbologies

This appendix contains a brief explanation of some of the bar code symbologies that the 700 Series Color Mobile Computer decodes and explains some of the general characteristics and uses of these bar code types.

The 700 Series Computer recognizes eleven of the most widely used bar code symbologies. With bar code symbologies, like languages, there are many different types. A bar code symbology provides the required flexibility for a particular inventory tracking system.

A symbology may be for particular industries, such as food and beverage, automotive, railroad, or aircraft. Some of these industries have established their own bar code symbology because other symbologies did not meet their needs.

Without going into great detail on the bar code structure, note that no two products use the same bar code. Each product gets a unique bar code.

Industries that use a particular type of bar code symbology have formed regulating committees or are members of national institutes that issue and keep track of bar codes. This ensures that each organization that contributes to a particular industry conforms to its standard. Without some form of governing body, bar coding would not work.

Codabar

Codabar was for retail price-labeling systems. Today it is widely accepted by libraries, medical industries, and photo finishing services.

Codabar is a discrete, self-checking code with each character represented by a stand-alone group of four bars and three intervening spaces.

Four different start or stop characters get defined and designated “a”, “b”, “c”, and “d”. These start and stop characters are constructed using one wide bar and two wide spaces. A complete Codabar symbol begins with one of the start or stop characters followed by some number of data characters and ending in one of the start or stop characters.

Any of the start or stop characters may be used on either end of the symbol. It is possible to use the 16 unique start or stop combinations to identify label type or other information.

Since Codabar is variable-length, discrete, and self-checking, it is a versatile symbology. The width of space between characters is not critical and may vary significantly within the same symbol. The character set consists of “0” through “9”, “-”, “$”, “:”, “/”, “.”, and “+”.

The specific dimensions for bars and spaces in Codabar optimize performance of certain early printing and reading equipment. Codabar has 18 different dimensions for bar and space widths. So many different dimensions often result in labels printed out of specification and cause Codabar printing equipment to be more expensive.

Code 11

Code 11 satisfies the requirements for a very high density, discrete numeric bar code. The name Code 11 derives from 11 different data characters that can be represented, in addition to a start or stop character.

The character set includes the 10 digits and the dash symbol. Each character is represented by a stand-alone group of three bars and two intervening spaces. Although Code 11 is discrete, it is not self-checking. A single printing defect can transpose one character into another valid character. One or two check digits obtain data security.

The specifications for Code 11 suggest that this code should have a narrow element width of 7.5 mils. This results in an information density of 15 characters per inch.

Code 39

Code 39 (C39) is the most widely used symbology among the industrial bar codes. Most major companies, trade associations, and the federal government find this code to fit their needs. The main feature of this symbology is the ability to encode messages using the full alphanumeric character set, seven special characters, and ASCII characters.

Programming for this symbology can be for any length that the application requires. The application program for the 751G Computer handles symbology at least one character but no more than 32 characters in length.

When programming the computer for Code 39, it is important to set the symbology limit as close as possible (minimum and maximum bar code lengths being scanned). Doing so keeps the computer bar code processing time to a minimum and conserves battery power.

Bar code readers can respond to Uniform Symbology Specification symbols in non-standard ways for particular applications. These methods are not for general applications, because of the extra programming required. Code 39 Full ASCII is one example of non-standard code.

Note: See page 398 to scan several Code 39 bar code labels available to change settings on your 751G Computer.

Encoded Code 39 (Concatenation)

If the first data character of a symbol is a space, the reader may be programmed to append the information contained in the remainder of the symbol to a storage buffer. This operation continues for all successive symbols that contain a leading space, with messages being added to the end of previously stored ones. When a message is read which does not contain a leading space, the contents are appended to the buffer, the entire buffer is transmitted, and the buffer is cleared.

Encoded Code 39 (Full ASCII)

If the bar code reader is programmed for the task, the entire ASCII character set (128 characters) could be coded using two character sequences: a symbol (“$”,“.”,“%”,“/”) followed by a letter of the alphabet.

Code 93

The introduction of Code 93 provided a higher density alphanumeric symbology designed to supplement Code 39. The set of data characters in Code 93 is identical with that offered with Code 39. Each character consists of nine modules arranged into three bars and three spaces.

Code 93 uses 48 of the 56 possible combinations. One of these characters, represented by a square, is reserved for a start or stop character, four are used for control characters, and the remaining 43 data characters coincide with the Code 39 character set. An additional single module termination bar after the stop character concludes the final space.

Code 93 is a variable length, continuous code that is not self-checking. Bar and spaces widths may be one, two, three, or four modules wide. Its structure uses edge-to-similar-edge decoding. This makes the bar code immune to uniform ink spread, which allows liberal bar width tolerances.

Code 93 uses two check characters. Its supporters believes this makes it the highest density alphanumeric bar code. The dual check digit scheme provides for high data integrity. All substitution errors in a single character are detected for any message length.

Code 128

Code 128 (C128) is one of the newest symbologies used by the retail and manufacturing industries. It responds to the need for a compact alphanumeric bar code symbol that could encode complex product identification.

The fundamental requirement called for a symbology capable of being printed by existing data processing printers (primarily dot-matrix printers) that produce daily, work-in-progress, job, and product traceability documents. The ability to print identification messages between 10 and 32 characters long, on existing forms and labels deemed an important requirement.

Code 128 uniquely addresses this need as the most compact, complete, alphanumeric symbology available.

Additionally, the Code 128 design with geometric features, improves scanner read performance, does self-checking, and provides data message management function codes.

Code 128 encodes the complete set of 128 ASCII characters without adding extra symbol elements. Code 128 contains a variable-length symbology and the ability to link one message to another for composite message transmission. Code 128, being a double-density field, provides two numeric values in a single character.

Code 128 follows the general bar code format of start zone, data, check digit, stop code, and quiet zone. An absolute minimum bar or space dimension of nine mils (0.010 inch minimum nominal ± 0.001 inch tolerance) must be maintained.

Characters in Code 128 consist of three bars and three spaces so that the total character set includes three different start characters and a stop character.

UCC/EAN-128 Shipping Container Labeling is a versatile tool that can ease movement of products and information. The Shipping Container Labeling bar code can take any form and usually has meaning only within the company or facility where applied.

Because this *random* data can get mistaken later for an industry standard code format, the UCC and EAN chose a symbology uniquely identified from these other bar codes. This standard is for maximum flexibility, to handle the diversity of distribution in global markets by cost efficiency.

The UCC/EAN-128 Container Labeling specification calls for a FUNC1 to immediately follow the bar code's start character. FUNC1 also follows any variable-length application field. The specification also calls for the computer to send "]C1" for the first FUNC1. The specification requires that the computer send a "<GS>" (hex 1D) for subsequent FUNC1 codes in the bar code.

Because "<GS>" is not compatible with computer emulation data streams, the Uniform Code Council has been asked to change the specification. This change is made to send the same three character sequence "]C1" to identify the embedded FUNC1 codes.

This implementation should provide for clean application coding by identifying the same sequences for the same scanned codes. If the communication of Norand bar code types is enabled, the Shipping Container Label codes precede with a "J". These strings will appear on the computer display. The application may have to allow for strings longer than 48 characters (maximum length indicated in the specification). Actual length variance depends on the number of variable-length data fields. Allowing for 60 characters should be sufficient. Within the Code 128 specification, the computer can link bar codes together. If this is to happen, allow for more characters (computer limit is 100 characters).

The Application Identifier Standard, that is part of the UCC/EAN Shipping Label concept, complements, rather than replaces, other UCC/EAN standards. Most UCC/EAN standards primarily identify products.

Several industries expressed the need to standardize more than product identification. The UCC/EAN Code 128 Application Identifier Standard supplies this tool. The standard adds versatility for inter-enterprise exchanges of perishability dating, lot and batch identification, units of use measure, location codes, and several other information attributes.

For more detailed information on Code 128 UCC/EAN Shipping Label bar code and Application Identifier Standard, refer to the UCC/EAN-128 Application Identifier Standard specification.

Data Matrix

Data Matrix is a high density 2D matrix code that can store a large amount of information. It has excellent error correction abilities and is mostly used for marking and tracking parts.

Data Matrix can store from 1 to about 2000 characters. The symbol is square and can range from 0.001 inch per side up to 14 inches per side. As an example of density, 500 numeric-only characters can encode in a 1-inch square using a 24-pin dot matrix printer.

Data Matrix is used to encode product and serial number information on electrical rating plates; to mark of surgical instruments in Japan; to identify lenses, circuit boards, and other items during manufacturing.

EAN (European Article Numbering)

EAN symbology is similar to UPC symbology, except that it contains 13 characters and uses the first two to identify countries.

The EAN symbology is used in the retail environment throughout most of Europe. Though similar to UPC symbology, these are not interchangeable.

I 2 of 5 (Interleaved)

I 2 of 5 (Interleaved 2 of 5 Code) is an all-numeric symbology, widely used for warehouse and heavy industrial applications. Its use has been particularly prevalent in the automobile industry. The I 2 of 5 symbology can be placed on smaller labels than what the standard UPC symbology requires.

I 2 of 5 also provides a little more flexibility on the type of material it can print on. Interleaved 2 of 5 Code has its name because of the way the bar code is configured.

I 2 of 5 bars and spaces both carry information. The bars represent the odd number position digits, while spaces represent the even number position digits. The two characters are interleaved as one. Messages encoded with this symbology have to use an even number of characters since two numeric characters always get interleaved together.

Matrix 2 of 5

Matrix 2 of 5 is a derivitive of Code 11 and is a linear bar code that is only read by linear imagers. It is limited to the ten digits and start/stop character. Discrete but not self-checking. Matrix 2 of 5 is used with a single Mmodulo 10 check digit. Compared with Code 11 (using two check digits) and the other industrial symbologies, Matrix 2 of 5 is somewhat more subject to substitution errors and offers no particular advantage.

MaxiCode

MaxiCode is a fixed-size code which holds up to 93 data characters. The symbol is composed of a central bulls-eye locator and offset rows of hexagonal elements; the overall dimensions of the symbol are approximately 1.11 x 1.054 inches. Each element measures 0.035 x 0.041 inches.

Created by United Parcel Service, the MaxiCode symbol was designed for quick automated scanning of packages on high-speed conveyor lines (special cameras can read a MaxiCode on a carton travelling at up to 500 feet per minute).

PDF417

The PDF417 symbology is a stacked 2D symbology that allows you to scan across rows of code. Each row consists of start/stop characters, row identifiers, and symbol characters, which consist of four bars and four spaces each and contain the actual data. This symbology uses error correction symbol characters appended at the end to recover loss of data.

PDF417 can store up to about 1800 printable ASCII characters or 1100 binary characters per symbol. The symbol is rectangular; the shape of the symbol can be adjusted to some extent by setting the width and allowing the height to grow with the data. It is also possible to break large amounts

of data into several PDF417 symbols which are logically linked. There is no theoretical limit on the amount of data that can store in a group of PDF417 symbols.

The capacity of PDF417 can be helpful in applications where the data must travel with the labeled item, where a host database is not always available for quick look-up. PDF417 is used for hazardous materials labeling; storing technical specifications, and calibration data on electronic instruments; encoding fingerprints and photographs on the backs of drivers' licenses.

The maximum data density is determined by the smallest elements which can be reliably printed and scanned. Using the smallest recommended element size of 0.0075 inch wide and 0.010 inch high, the maximum data density in the binary mode is 686 bytes per square inch (106.2 bytes per square centimeter). In the printable ASCII mode the density is 1144 characters per square inch (177.2 characters per square centimeter).

Micro PDF417

Micro PDF417 is derived from PDF417. The code has a limited set of symbol sizes and a fixed level of error correction for each symbol size. Module dimensions are user-specified so that the symbol may be printed with a variety of printers. The symbology allows up to 150 bytes, 250 alphanumeric characters, or 366 numeric digits to be stored. This is done by specifying one of three compaction modes: data, text or numeric. Text Compaction mode permits all printable ASCII characters to be encoded (values 32 to 126 inclusive) as well as selected control characters. Byte Compaction mode permits all 256 possible 8-bit byte values to be encoded. This includes all ASCII characters value 0 to 127 inclusive and provides for international character set support

Micro PDF417 is designed for applications where the symbol must be smaller than PDF417 will allow.

Plessey

Plessey finds its origin in the pulse width modulated (PWM) code developed in England. It is widely used for shelf markings in grocery stores. Pulse width modulated codes represent each bit of information by a bar and space pair. A zero bit consists of a narrow bar followed by a wide space, while a one bit consists of a wide bar followed by a narrow space. It is mainly a numeric symbology (0-9) with six extra characters available for assigning any symbol or letter desired.

Plessey codes employ a variety of check characters and a polynomial-based Cyclic Redundancy Check (CRC). For start and stop characters, Plessey employs a 1101 and previously used a 0101.

This symbology is very limited about what information can be encoded. It is not considered for new applications.

MSI Code (Variant of Plessey)

The MSI Plessey bar code is a variant of the Plessey bar code. It is a pulse-width modulated non-self checking code, and is used primarily in store shelf labeling. Each character consists of eight elements, four bars and four spaces. The character set includes the digits 0 through 9. A Modulo 10 checksum is appended to the end of the code. For start and stop checks, MSI employs a single bit pair of 1 as a start symbol and a single bit pair of 0 as a stop symbol. MSI reverses the 1-2-4-8 BCD pattern for bit pair weighting to 8-6-2-1.

QR Code (Quick Response Code)

QR Code is a 2D matrix symbology containing dark and light square data modules. It has position detection patterns on three of its for corners and features direct encodation of the Japanese Kana-Kanji character set. A 2D imaging device such as a CCD camera is necessary to scan the symbology. QR Code is designed with selectable levels of error correction. It supports industry standard escape sequences to define international code pages and special encodation schemes. QR Code is used for small item marking applications using a wide variety of printing and marking technologies. This document includes descriptions of the character encodation, symbol structure, reference decode algorithm, and symbol quality measurements for QR Code.

S 2 of 5 (Standard 2 of 5)

The code S 2 of 5 (Standard 2 of 5 Code) is designed primarily for:

- Warehouse inventory handling
- Identification of photo finishing envelopes
- Airline tickets
- Baggage and cargo handling

The code S 2 of 5 is simple and straightforward. All information is contained in the widths of the bars, with the spaces serving only to separate the individual bars.

Bars can either be wide or narrow, and the wide bars are usually three times the widths of the narrow bars. Spaces may be any reasonable width but are typically equal to the narrow bars. Narrow bars are identified as zero bits and wide bars as one bits.

Remember the code structure by associating the bar positions from left to right with weighting factors 1, 2, 4, 7, and parity. Exceptions to this rule are zero, start, and stop. This code is a discrete code, since the white spaces between the characters are not part of the code. Because the white spaces carry no information, their dimensions are not critical.

The S 2 of 5 code is self-checking, meaning a scanner passing through a printing void would detect the proper ratio of wide bars to total bars. When the scanner spots an error, a non-read will occur.

Telepen

Telepen was devised by George Sims, Managing Director of SB Electronic Systems Limited, in early 1972, this is the only symbology to directly represent the full ASCII character set without shift characters.Telepen carries the double-density numeric-only mode and is very compact. The Telepen symbol is up to 8 ASCII characters or16 digit per inch, and is easy to print. It has a fixed 3:1 ratio, with a tolerance at least 0.4x. This symbology is extremely secure, with negligible risk of misreading. It is supported by most leading manufacturers.

UPC (Universal Product Code)

The UPC (Universal Product Code) is the symbology used throughout the grocery and retail industries. This bar code symbology contains two pieces of numerical information encoded on the bar code, producer identification, and product identification information.

The UPC symbol is 12 characters long. The first character of the UPC symbol is a number system character, such as “0” for grocery items and “3” for drug- and health-related items.

The UPC symbology is for retail environments such as grocery stores, convenience stores, and general merchandise stores.

Some retail items are so small that a standard UPC bar code cannot fit on the packaging. When this occurs there is a permitted shorter version of the UPC symbology, referred to as UPC-E. UPC-E is six characters long (eight including number system and check digit), approximately half the size of a standard UPC bar code.

Index

The Classes and Functions Index covers classes and functions for the 700 Series Color Mobile Computer.

The General Index covers all topics. Those in italics are figures, those in bold are tables.

The Files Index is to assist you in locating descriptions for device drivers, applications, utilities, batch files, or other files within this publication.

Classes and Functions

A

add_registry_section, [AddReg]
  flags, 216
  registry_root_string, 216
  value_name, 216
AddReg, [DefaultInstall], 212
[AddReg], add_registry_section
  flags, 216
  registry_root_string, 216
  value_name, 216
AddWep(), 263
AppName, [CEStrings], 209

B

BuildMax, [CEDevice], 210
BuildMin, [CEDevice], 210

C

[CEDevice]
  BuildMax, 210
  BuildMin, 210
  ProcessorType, 210
  UnsupportedPlatforms, 210
  VersionMax, 210
  VersionMin, 210
CESelfRegister, [DefaultInstall], 212
CESetupDLL, [DefaultInstall], 212
CEShortcuts, [DefaultInstall], 212
[CEShortcuts], shortcut_list_section
  shortcut_filename, 217
  shortcut_type_flag, 217
  target_file/path, 217
  target_file_path, 217
CESignature
  [SourceDiskNames], 212
  [Version], 208
[CEStrings]
  AppName, 209
  InstallDir, 209
CloseHandle()
  DTR printing, 188, 189
  IrDA printing, 182
  NPCP printing, 183, 184
ConfigureProfile(), 268
Copyfiles, [DefaultInstall], 212
[CopyFiles], file_list_section
  destination_filename, 215
  flags, 215
  source_filename, 215
CreateEvent(), 279
CreateFile()
  DTR printing, 188, 189
  IrDA printing, 182
  NPCP printing, 183, 184

D

[DefaultInstall]
  AddReg, 212
  CESelfRegister, 212
  CESetupDLL, 212
  CEShortcuts, 212
  Copyfiles, 212
DeregisterDevice(), 183
  DTR printing, 188
[DestinationDirs], file_list_section, 214
DeviceIOControl(), 251
  DTR printing, 188
  NPCP printing, 183
DeviceIoControl(), NPCP printing, 184, 185
disk_ordinal, [SourceDiskNames], 212
DllRegisterServer, 212
DllUnregisterServer, 212

E

EnableSuppLogging(), 272
EnableWep(), 263
EnableZeroConfig(), 269
EncryptionStatus(), 264
EncryptWepKeyForRegistry(Deprecated), 273

F

file_list_section
  [CopyFiles]
    destination_filename, 215
    flags, 215
    source_filename, 215
  [DestinationDirs], 214
filename, [SourceDiskFiles], 213

G

GetAssociationStatus(), 254
GetAuthenticationMode(), 255
GetBSSID(), 255
GetCCXStatus(), 262
GetCurrentDriverName(), 272
GetDiversity(), 256
GetLinkSpeed(), 256
GetMac(), 257
GetMedia(Deprecated), 273
GetMedium(Deprecated), 273
GetNetworkMode(), 257
GetNetworkType(), 258
GetNicStats(Deprecated), 273
GetPowerMode(), 259
GetRadioIpAddress(), 262
GetRSSI(), 259
GetRTSThreshold(Deprecated), 273
GetSSID(), 258
GetTXPower(), 260
GetWepStatus(), 261

I

InstallDir, [CEStrings], 209
isDHCPEnabled(), 271
isOrinoco(), 269
isSupplicantRunning(), 270
isZeroConfigEnabled(), 269

K

KernelIoControl(), 235

N

NLEDGetDeviceInfo, 275
NLEDSetDevice, 275

O

OSVERSIONINFO.dwBuildNumber, 210
OSVERSIONINFO.dwVersionMajor, 210
OSVERSIONINFO.dwVersionMinor, 210

P

ProcessorType, [CEDevice], 210
Provider, [Version], 208

R

RadioConnect(), 253
RadioDisassociate(), 254
RadioDisconnect(), 253
ReadFile(), NPCP printing, 183
RegFlushKey(), 123, 225
RegisterDevice(), 183
  DTR printing, 188
RegOpenKeyEx(), 278
RegQueryValueEx(), 278
RegSetValueEx(), 278
RemoveWep(), 268
RenewDHCP(), 271
ResetRadioToSystemSave(), 272

S

SetAuthenticationMode(), 265
SetCCXStatus(), 267
SetChannel(), 265
SetDiversity(Deprecated), 273
SetMixedCellMode(), 267
SetNetworkMode(), 266
SetPowerMode(), 266
SetRTSThreshold(Deprecated), 273
SetSSID(), 267
SetTXRate(Deprecated), 273
SHFullScreen(), 225, 226
shortcut_list_section, [CEShortcuts]
  shortcut_filename, 217
  shortcut_type_flag, 217
  target_file/path, 217
  target_file_path, 217
Signature, [Version], 208
[SourceDiskFiles], filename, 213
[SourceDiskNames]
  CESignature, 212
  disk_ordinal, 212
SourceDisksNames.MIPS, 213
SourceDisksNames.SH3, 213
StartScanList(), 270
StartSupplicant(), 270
StopSupplicant(), 271
string_key, [Strings], 209
[Strings], string_key, 209
SwitchPacketDriver(), 273
SYSTEMINFO.dwProcessorType, 210

U

UnsupportedPlatforms, [CEDevice], 210

V

[Version]
  CESignature, 208
  Provider, 208
  Signature, 208
VersionMax, [CEDevice], 210
VersionMin, [CEDevice], 210

W

WriteFile()
  DTR printing, 188, 189
  IrDA printing, 182
  NPCP printing, 183, 184

General Index

Numbers

1470 Imager. *See* Imager
1551/1553 Tethered Scanner
  See also Tethered scanner
  configuring, 201
  reset to factory defaults, 204
  troubleshooting, 204
1D laser scanner, about, 191
1D OmniDir Decode Enable, configuration parameter, 328
2D Imager, about, 191
4820 printer, NPCP driver, 183
6804DM printer
  DTR driver, 188
  IrDA driver, 182
6804T printer
  DTR driver, 188
  IrDA driver, 182
6805A printer
  DTR driver, 188
  IrDA driver, 182
6806 printer
  DTR driver, 188
  IrDA driver, 182
6808 printer
  DTR driver, 188
  IrDA driver, 182
  printer support, 181
681T printer, DTR driver, 188
6820 printer
  IrDA driver, 182
  NPCP driver, 183
  printer support, 181
6920 Communications Server, ManifestName parameter, 229
700 Platform Build, version number, 349
740 Color Computer, 277
781 printers, DTR driver, 188
782T printer, printer support, 181
802.11 CR radio CORE module, 129
  installing available modules, 126
  loading a module, 126
802.11b
  antenna color code, 127
  API, 252
  channel, 362
  communications setup, 129, 361
  configuration profiles, 252
  CORE module, 129
  LEAP
    network EAP, 380
    WPA encryption, 379
  network type, 362
  PEAP
    network EAP, 369
    WPA encryption, 368
  profile label, 362
  profile security information, WEP encryption, 365
  profiles, 361
    advanced settings, 381
    basic information, 362
    security information, 363
  SSID (network name), 362
  TTLS, WPA encryption, 376
  WPA encryption, 366

A

Abstract Syntax Notation.1. *See* ASN.1
Accessory list, **22**
Accounts, via Inbox, 75
ActiveSync
  ActiveSync Help, 45
  adding programs, 42
  adding programs to Start menu, 43
  Folder behavior connected to e-mail server, 74
  installing applications, 119
  Microsoft Reader, 90
  Pocket Internet Explorer
    favorite links, 94
    mobile favorites, 95
    Mobile Favorites folder, 94
  Start menu icon, 27
  URL, 44
  Windows Mobile, 44
ActiveX control tools, unit information control panel, CAB files, 352
AD command, with/without data, 331
Adding bookmarks, Microsoft Reader, 93
Adding drawings to text, Microsoft Reader, 93
Adding programs
  ActiveSync, 42
  Pocket Internet Explorer, 42
  to the Start menu, 43
    via ActiveSync, 43
    via File Explorer, 43
  Windows Mobile, 41
Adjusting settings, Windows Mobile, 41
Adobe Acrobat Reader, URL, 169
AE command, with/without data, 333
Aimer LED Duration, configuration parameter, 324
All-Day events, Calendar, 48
  creating, 52
Alpha plane on keypad, 278
Alphanumeric keypad
  alpha (blue) key sequences, 15
  [gold/white] key sequences, 13
  registry settings
    alpha plane, 278
    gold plane, 278
    unshifted plane, 278
  scan codes, 281
Ambient lighting, 2
Annotations index, Microsoft Reader, 93
ANT_DIVERSITY, GetDiversity(), 256
ANT_PRIMARY, GetDiversity(), 256
ANT_SECONDARY, GetDiversity(), 256

Antenna, radio type, 127
APIs
  802.11b, 252
  AT command interface, 169
  IrSock, 182
App launch, control panel applet, 357
Application keys
  app launch control panel applet, 357
  wakeup mask control panel applet, 356
Appointments
  Calendar
    adding a note, 54
    assigning to a category, 56
    changing, 51
    creating, 51
    deleting, 58
    finding, 58
    making recurring, 55
    setting a reminder, 53
    viewing, 49
  via Calendar, 46
APS linear imager, about, 191
ASCII
  printing, 182
  printing to a port, port print method, 182
  raw text to printer, 182
ASN.1, 179
Asset management, DeviceURL parameter, 228
AT command interface, 169
  testing, 170
Attaching notes to text, Microsoft Reader, 93
Audio, phone application, 167
Audio control panel applet, input mixing, 8
Audio files, Windows Media Player, 89
Audio system
  external headset jack, 4
  microphone, 4
  speaker, 3
AutoCab, command line syntax, 124
AutoFTP, 233
AutoIP, 178
Automatic Private IP. *See* AutoIP
Automatic shutoff
  bar code configuration, 382, 387
  configuration parameter, 382
Autostart FTP, 233
AvantGo channels, Pocket Internet Explorer, 96
AXCommunication, 352
AXFileTransfer, 352
AXReaderCommand, 352
AXVWedge, 352

B

Backlight control panel applet
  ambient light sensor, 2
  keypad, 11
Backlight timeout
  bar code configuration, 383, 387
  configuration parameter, 383
Bar code configuration
  audio volume, 382
  automatic shutoff, 382
  backlight timeout, 383
  key clicks, 383
Bar codes
  configuration
    audio volume, 386
    automatic shutoff, 387
    backlight timeout, 387
    Code 39, 386
    key clicks, 388
  internal scanner supported symbologies, **195**
  scanning labels, 386
  supported symbologies, 289
  symbologies
    Codabar, 392
    Code 11, 392
    Code 128, 394
    Code 39, 392
    Code 39 concatenation, 393
    Code 39 full ASCII, 393
    Code 93, 393
    Data Matrix, 395
    EAN, 395
    I 2 of 5, 396
    Matrix 2 of 5, 396
    MaxiCode, 396
    Micro PDF417, 397
    MSI code, 398
    PDF417, 396
    Plessey, 397
    QR Code, 398
    S 2 of 5, 398
    Telepen, 399
    UPC, 399
  tethered scanner supported symbologies, **205**
Basic connect/disconnect functions, 253
Battery
  ambient lighting, 2
  low battery conditions, 6
  RAM maintenance, 6
  status, 5
Battery status, unit information control panel applet, 350
Beeper
  configuration parameter
    frequency, 321
    volume, 319, 320
  disabling the volume, 320
  selecting a volume, 9
  silencing the volume, 10
  supported functions, 318
  volume, turning it on, 7
  when not available
    beeper frequency, 321
    good read beep duration, 323
    good read beeps, 322
Bell Mobility activation process, 157
Block recognizer, Windows Mobile input panel, 32

BlockSize, FTP Server, 228
Bluealps CORE module
  installing available modules, 126
  loading a module, 126
Bluetooth
  accessing, 174
  activating, 174
  unit information control panel, main stack CAB file, 351
Bluetooth compatibility, network support, 174
Books, Microsoft Reader
  adding bookmarks, 93
  adding drawings, 93
  annotations index, 93
  attaching notes, 93
  copying, 93
  downloading, 90
  highlighting, 93
  reading, 92
  removing, 93
  searching, 93
Browsing the Internet, Pocket Internet Explorer, 97
Build information, software, 18

C

CAB files
  after the extraction, 220
  creating, 208
    INF files, 208
    with CAB Wizard, 223
  installation functions, SETUP.DLL, 220
  placing files onto storage card, 122
  unit information control panel applet, 351
Cabinet Wizard
  creating CAB files, 223
  troubleshooting, 224
  using the application, 208
Cabling, scanner, 199
Calendar
  all day events, 48
    creating, 52
  appointments
    adding a note, 54
    assigning to a category, 56
    changing, 51
    creating, 51
    deleting, 58
    finding, 58
    making recurring, 55
    setting a reminder, 53
    viewing, 49
  categories, 47
  meetings, sending a request, 57
  options, changing, 59
  Pocket Outlook, 46
  recurrence pattern, 49
  Start menu icon, 27
  synchronizing, 47
Capacitor, internal super, 6
Capturing thoughts and ideas, via Notes, 71
Card support
  CompactFlash cards, 19
  MultiMediaCards, 19
  radios, 21
  Secure Digital cards, 19
Carrier, location of ESN, 140
Categories
  calendar, 47
  contacts, assigning to, 65
CDMA/1xRTT, 134
  activation with SB555 Watcher, 140
    Bell Mobility, 157
    Sprint, 147
    Telus, 157
    Verizon, 143
    WWANInit demo program, 157
  antenna color code, 127
  AT command set, 169
  copying files from web site, 137
    via Microsoft ActiveSync, 138
    via storage cards, 138
  CORE module, 134
  location of ESC, 140
  setting up, 137
    terminology, 137
CEImager
  location of the executable file, 122
  migrating AUTORUN.DAT files, 122
Channel, 802.11 radio module, 362
ClassID field values
  VN_CLASS_ASIC, 237
  VN_CLASS_BOOTSTRAP, 237
  VN_CLASS_KBD, 237
Clock
  restore real-time after cold-boot, 355
  setting date and time, 385
  Windows Mobile settings, 41
Closing drivers, NPCP, 184
Codabar, 392
  configuration parameter, 292
    user ID, 311
Code 11, 392
  configuration parameter, 306
    user ID, 315
Code 128, 394
  configuration parameter, 295
    FNC1 character, 297
    user ID, 311
Code 39, 392
  configuration parameter, 290
    user ID, 311
Code 93, 393
  configuration parameter, 294
    length, 294
    user ID, 312
Code Division Multiple Access. *See* CDMA/1xRTT

Codes
11, 392
128, 394
39, 392
39 concatenation, 393
39 full ASCII, 393
93, 393
Cold boot, IOCTL_HAL_COLDBOOT, 245
Cold boot, performing, 17
COM A, modem position, 354
COM B, serial position, 354
COM port
configuration, 197
wedge settings, 197
COM1, NPCP parameter, 183
COM1 port, 182
Comm port wedge
disabling, 197
enabling, 196
error messages, 197
limitations, 199
settings, 197
unit information control panel, 351
Command line syntax, AutoCab, 124
Common Object Resource Environment. *See* CORE
Communications
DTR, 189
NPCP, 186
Communications options, 125
CompactFlash cards
card support, 19
installing applications, 120
migrating applications, 122
packaging an application, 118
Computer shutdown, 6
Concatenation, 393
Configuration parameters
1D OmniDir decode enable, 328
aimer LED duration, 324
automatic shutoff, 382
backlight timeout, 383
beeper, 319
frequency, 321
volume, 320
codabar, 292
user ID, 311
code 11, 306
user ID, 315
code 128, 295
FNC1 character, 297
user ID, 311
code 39, 290
user ID, 311
code 93, 294
length, 294
user ID, 312
datamatrix, 308
date/time, 383
EAN
13 user ID, 314
8 user ID, 314
good read
beep duration, 323
beeps, 322
identification
contact, 345
location, 347
name, 346
image dimension, 326
interleaved 2 of 5, 303
user ID, 312
key clicks, 383
lighting mode, 327
macro PDF, 300
matrix 2 of 5, 304
user ID, 314
maxicode, 309
micro PDF417, 302
MSI, 299
user ID, 312
PDF417, 300
user ID, 312
plessey, 298
user ID, 313
prefix, 316
QR code, 307
security
encryption key, 342
read encryption, 340
read-only community string, 338
read/write community string, 339
write encryption, 341
SNMP, security subnet mask, 310
standard 2 of 5, 291
user ID, 313
sticky aimer duration, 325
suffix, 317
telepen, 305
user ID, 314
trap
authentication, 343
threshold, 344
UPC
A user ID, 313
E user ID, 313
UPC/EAN, 293
virtual wedge, 329
code page, 335
grid, 334
postamble, 332
preamble, 330
volume, 382
Connecting to
an ISP, 98
e-mail server, 114
work, 104
Connecting to a mail server, via Inbox, 75

Connections
See also Getting connected
directly to e-mail server, 114
ending, 114
setting up an e-mail account, 114
to an ISP, 98
via Ethernet, 103
via modem, 98
to work, 104
via Ethernet, 113
via modem, 106
via VPN server, 111
via wireless network, 108
via Ethernet
to an ISP, 103
to work, 113
via modem
to an ISP, 98
to work, 106
via VPN server, to work, 111
via wireless network, to work, 108
Conserving battery power, 2
Contacts
adding a note, 64
assigning to a category, 65
changing, 63
changing options, 68
copying, 65
creating, 61, 63
deleting, 67
finding, 67
MSN Messenger
managing, 87
sending messages, 88
working with, 86
Pocket Outlook, 60
sending a message, 66
Start menu icon, 27
synchronizing, 61
viewing, 62
Control panel applets
Audio, 8
backlight, 2, 11
clock, 385
data collection, 288
beeper volume, 9
beeper/LED, 318
imager, 324
symbologies, 289
symbology options, 310
vibrator, 20
virtual wedge, 329
intemec settings
beeper volume, 10
vibrator, 21
intermec settings, 288, 336
power
battery status, 5
RAM maintenance, 6
SNMP, 337
identification, 345
security, 338
traps, 343
system, wireless network, 129, 361
unit information, 348
battery status, 5, 350
CAB files, 351
versions, 18, 349
utilities, 353
app launch, 357
dock switch, 354
registry save, 123, 355
wakeup mask, 356
wireless network, 359
Converting writing to text, 35
Copying, contacts, 65
Copying text, Microsoft Reader, 93
CORE, 126
802.11b radio module, 129
details, 131
general, 130
accessing from
Programs panel, 126
Today screen, 127
activating, 126
installing available modules, 126
loading a module, 126
module for 802.11b NIC, 252
WAN monitor, GSM/GPRS, 164
WAN radio module
CDMA/1xRTT, 134
general, 164
Creating
a modem connection
to an ISP, 98
to work, 106
a VPN server connection, to work, 111
a wireless network connection, to work, 108
an Ethernet connection
to an ISP, 103
to work, 113
CAB files, 208
with CAB Wizard, 223
contacts via Contacts, 61
document via Pocket Word, 78
drawing via Notes, 37
INF files, 208
note via Notes, 71
task via Tasks, 70
workbook via Pocket Excel, 82

D

Data collection
- build version number, 349
- configuration parameters
  - 1D OmniDir decode enable, 328
  - aimer LED duration, 324
  - beeper, 319
  - beeper frequency, 321
  - beeper volume, 320
  - codabar, 292
  - codabar user ID, 311
  - code 11, 306
  - code 11 user ID, 315
  - code 128, 295
  - code 128 FNC1 character, 297
  - code 128 user ID, 311
  - code 39, 290
  - code 39 user ID, 311
  - code 93, 294
  - code 93 length, 294
  - code 93 user ID, 312
  - datamatrix, 308
  - EAN-13 user ID, 314
  - EAN-8 user ID, 314
  - good read beep duration, 323
  - good read beeps, 322
  - image dimension, 326
  - interleaved 2 of 5, 303
  - interleaved 2 of 5 user ID, 312
  - lighting mode, 327
  - macro PDF, 300
  - matrix 2 of 5, 304
  - matrix 2 of 5 user ID, 314
  - maxicode, 309
  - micro PDF417, 302
  - MSI, 299
  - MSI user ID, 312
  - PDF417, 300
  - PDF417 user ID, 312
  - plessey, 298
  - plessey user ID, 313
  - prefix, 316
  - QR code, 307
  - standard 2 of 5, 291
  - standard 2 of 5 user ID, 313
  - sticky aimer duration, 325
  - suffix, 317
  - telepen, 305
  - telepen user ID, 314
  - UPC-E user ID, 313
  - UPC-A user ID, 313
  - UPC/EAN, 293
  - virtual wedge, 329
  - virtual wedge code page, 335
  - virtual wedge grid, 334
  - virtual wedge postamble, 332
  - virtual wedge preamble, 330
- vibrator, 20

Data Matrix, 395
- configuration parameter, 308

Date, setting, 385
Date/Time, configuration parameter, 383
Deprecated functions, 273
DeviceName, FTP Server, 228
DeviceURL, FTP Server, 228
DHCP, 178
Display full screen, 226
Dock switch, control panel applet, 354
Docks, modem support, 16
DRAM
- low battery shutdown, 6
- maintenance, 6

Drawing mode, Pocket Word, 81
Drawing on the screen
- *See also* Notes
- Pocket Word, 81

Drivers
- DTR
  - communications, 189
  - installing, 188
  - opening, 189
  - removing, 188
  - writing to, 189
- NPCP
  - closing, 184
  - communications, 186
  - I/O controls, 185
  - installing, 183
  - opening, 184
  - reading from, 184
  - removing, 183
  - writing to, 184
- O'Neil. *See* DTR printing

DTR printing, 188
- closing driver, 189
- communications, 189
- opening driver, 189
- removing driver, 188
- writing to driver, 189

E

E-mail account, setting up an account, 114
E-mail server, getting connected, 114
EAN, configuration parameter, 293
- 13 user ID, 314
- 8 user ID, 314

Editing a profile, 361
Edition information, 24
Encoded Code 39
- concatenation, 393
- full ASCII, 393

Ending a connection, 114
Epson Escape Sequences, 182
Error messages
- comm port wedge, 197
- tethered scanner, 197

ERROR_INSUFFICIENT_BUFFER
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
ERROR_INVALID_PARAMETER
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
ESN, location on computer, 140
Ethernet
communications setup, 128
creating a connection
to an ISP, 103
to work, 113
ETSI GSM 07.05 interface specifications, 169
ETSI GSM 07.07 interface specifications, 169
European Article Numbering. *See* EAN
European Article Numbering code. *See* EAN

F

Favorite links, Pocket Internet Explorer, 94
File Explorer
adding programs to Start menu, 43
removing programs, 43
Windows Mobile, 40
File Transfer Protocol. *See* FTP
Find feature, Windows Mobile, 40
Flash File Store
migrating applications, 122
packaging an application, 118
Flash file system, control panel applet, 355
Folder behavior connected to e-mail server
ActiveSync, 74
IMAP4, 74
POP3, 74
SMS, 74
FRAME_NOT_ACKED, 185
FTP
client, 230
configurable parameters, 228
BlockSize, 228
DeviceName, 228
DeviceURL, 228
IDNATarget, 229
ManifestName, 229
PauseAtStartup, 229
Root, 229
FTPDCMDS subdirectory, 232
heartbeat, 230
RTC 959, 232
server, 230
installing applications, 120
server requests
CDUP, 230
CWD, 230
DELE, 230
HELP, 230
LIST, 230
MKD, 230
MODE, 230
NLST, 230
NOOP, 230
PASS, 230
PWD, 230
QUIT, 230
RETR, 230
RMD, 230
RNFR, 230
RNTO, 230
SITE, 231
SITE ATTRIB, 231
SITE BOOT, 231
SITE COPY, 231
SITE EKEY, 232
SITE EVAL, 232
SITE EXIT, 231
SITE GVAL, 232
SITE HELP, 231
SITE KILL, 231
SITE LOG, 231
SITE PLIST, 231
SITE PVAL, 232
SITE RUN, 231
SITE STATUS, 232
SITE TIMEOUT, 232
STOR, 230
SYST, 230
TYPE, 230
USER, 230
XCUP, 230
XCWD, 230
XMKD, 230
XPWD, 230
XRMD, 230
stopping server from application, 233
support, 230
web browsers, 232
FTPDCMDS subdirectory, FTP support, 232
Full screen display, 226

G

GDI approach, 182
General Packet Radio Service. *See* GSM/GPRS
Getting connected
directly to an e-mail server, 114
ISP, 98
setting up an e-mail account, 114
to an ISP, 98
creating a modem connection, 98
creating an Ethernet connection, 103
to work, 104
creating a modem connection, 106
creating a VPN server connection, 111
creating a wireless network connection, 108
creating an Ethernet connection, 113
Windows Mobile, 98
Gold plane on keypad, 277
Good read, configuration parameter
beep duration, 323
beeps, 322
Grid data, configuration parameter, 334

GSM/GPRS, 164
antenna color code, 127
AT command set, MC45, 169
CORE module, 164
phone application, 166

H

HAL, verion of Pocket PC
IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
IOCTL_HAL_GET_OAL_VERINFO, 243
Handset
phone application, 166
volume, 168
Headset jack, external, 4
Helper functions, 268
Highlighting text, Microsoft Reader, 93

I

I 2 of 5. *See* Interleaved 2 of 5
I/O controls, NPCP driver, 185
ID field values
IOCTL_HAL_ITC_READ_PARM
ITC_NVPARM_80211_INSTALLED, 238
ITC_NVPARM_80211_RADIOTYPE, 238
ITC_NVPARM_ANTENNA_DIVERSITY, 237
ITC_NVPARM_BLUETOOTH_INSTALLED, 239
ITC_NVPARM_CONTRAST, 237
ITC_NVPARM_DISPLAY_TYPE, 237
ITC_NVPARM_ECN, 237
ITC_NVPARM_EDBG_SUBNET, 237
ITC_NVPARM_EDG_IP, 237
ITC_NVPARM_ETHERNET_ID, 236
ITC_NVPARM_INTERMEC_DATACOLLECTION_HW, 238
ITC_NVPARM_INTERMEC_DATACOLLECTION_SW, 238
ITC_NVPARM_INTERMEC_SOFTWARE_CONTENT, 237
ITC_NVPARM_LAN9000_INSTALLED, 239
ITC_NVPARM_MANF_DATE, 236
ITC_NVPARM_MCODE, 237
ITC_NVPARM_RTC_RESTORE, 238
ITC_NVPARM_SERIAL_NUM, 236
ITC_NVPARM_SERIAL2_INSTALLED, 239
ITC_NVPARM_SERVICE_DATE, 236
ITC_NVPARM_SIM_PROTECT_HW_INSTALLED, 239
ITC_NVPARM_SIM_PROTECT_SW_INSTALLED, 239
ITC_NVPARM_VERSION_NUMBER, 237
ITC_NVPARM_VIBRATE_INSTALLED, 239
ITC_NVPARM_WAN_FREQUENCY, 238
ITC_NVPARM_WAN_INSTALLED, 238
ITC_NVPARM_WAN_RADIOTYPE, 238
ITC_NVPARM_WAN_RI, 237
IOCTL_HAL_ITC_WRITE_SYSPARM
ITC_ DOCK_SWITCH, 241
ITC_ WAKEUP_MASK, 241
ITC_AMBIENT_FRONTLIGHT, 241
ITC_AMBIENT_KEYBOARD, 241
ITC_REGISTRY_SAVE_ENABLE, 241
Identification, configuration parameter
contact, 345
location, 347
name, 346
IDNA
DeviceName, 228
DeviceURL, 228
IDNATarget, 229
ManifestName, 229
IDNATarget, FTP Server, 229
Image dimension, configuration parameter, 326
Imager
beeper functions not available
beeper frequency, 321
good read beep duration, 323
good read beeps, 322
beeper/LED parameters, beeper, 319
control panel appet, data collection, 324
data collection parameters
1D OmniDir decode enable, 328
aimer LED duration, 324
datamatrix, 308
image dimension, 326
lighting mode, 327
maxicode, 309
QR code, 307
sticky aimer duration, 325
settings, 198
supported
beeper functions, 318
functions, 324
symbologies, 289
symbologies not available
CIP 128 French Pharmaceutical, 296
Code 11, 306
Code 128 FNC1 character, 297
EAN 128]C1, 296
Macro PDF, 300
Matrix 2 of 5, 304
micro PDF417, 302
Telepen, 305

symbology user IDs not available
Codabar, 311
Code 11, 315
Code 128, 311
Code 39, 311
Code 93, 312
EAN 13, 314
EAN 8, 314
Interleaved 2 of 5, 312
Matrix 2 of 5, 314
MSI, 312
PDF417, 312
Plessey, 313
Standard 2 of 5, 313
Telepen, 314
UPC A, 313
UPC E, 313
vibrator, enabling, 20
IMAP4, Folder behavior connected to e-mail server, 74
Inbox
accounts, 75
composing/sending messages, 77
connecting to a mail server, 75
downloading messages from server, 76
getting connected, 98
managing e-mail messages and folders, 74
Pocket Outlook, 73
Start menu icon, 27
synchronizing e-mail messages, 73
using My Text, 39
INF files, creating, 208
Input Mixing, Audio control panel applet, 8
Input panel
block recognizer, 32
keyboard, 31
letter recognizer, 33
Pocket Word, 79
selecting typed text, 33
transcriber, 33
Windows Mobile, 28
word suggestions, 31
Installation functions, SETUP.DLL, 220
Installing applications
using a storage card, 120
using CompactFlash cards, 120
using Secure Digital cards, 121
with ActiveSync, 119
with FTP Server, 120
Installing drivers
DTR, 188
NPCP, 183
Instant messaging, 84
Integrated scanners. *See* Internal scanners
Interface specifications, ETSI GSM 07.0x, 169
Interleaved 2 of 5, 396
configuration parameter, 303
user ID, 312
Intermec Device Network Announcement. *See* IDNA
Intermec part numbers, **22**
Intermec settings, 288, 336
beeper volume, 10
vibrator, 21
INTERMEC_PACKET_DRIVER, SwitchPacketDriver(), 273
Internal scanners
configuring, 194
supported symbologies, **195**
Internet explorer
software build version, 18
Windows Mobile 2003 edition, 24
Internet Service Provider. *See* ISP
IOCTL_GET_CPU_ID, 250
IOCTL_HAL_COLDBOOT, 245, 276
IOCTL_HAL_GET_BOOT_DEVICE, 247
IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
IOCTL_HAL_GET_DEVICE_INFO, 235
IOCTL_HAL_GET_DEVICEID, 242
IOCTL_HAL_GET_OAL_VERINFO, 243
IOCTL_HAL_GET_RESET_INFO, 246
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
IOCTL_HAL_REBOOT, 248, 276
IOCTL_HAL_WARMBOOT, 245, 276
IOCTL_LOAD_NDIS_MINIPORT, 251
IOCTL_NPCP_BIND, 185
IOCTL_NPCP_CANCEL, 185
IOCTL_NPCP_CLOSE, 185
IOCTL_NPCP_ERROR, 185
IOCTL_NPCP_FLUSH, 185
IOCTL_PROCESSOR_INFORMATION, 249
IOCTL_UNLOAD_NDIS_MINIPORT, 251
IrDA printing, 182
ISP
connecting to via Windows Mobile, 98
creating
a modem connection, 98
an Ethernet connection, 103
Pocket Internet Explorer, 94
Windows Mobile, 98
ITC_ DOCK_SWITCH, 241
ITC_ WAKEUP_MASK, 241
ITC_AMBIENT_FRONTLIGHT, 241
ITC_AMBIENT_KEYBOARD, 241
ITC_DEVID_80211RADIO_INTEL_2011B, 238
ITC_DEVID_80211RADIO_MAX values
ITC_DEVID_80211RADIO_INTEL_2011B, 238
ITC_DEVID_80211RADIO_NONE, 238
ITC_DEVID_80211RADIO_NONE, 238
ITC_DEVID_INTERMEC_EVIO, 238
ITC_DEVID_INTERMEC2D_IMAGER, 238
ITC_DEVID_OEM2D_IMAGER, 238
ITC_DEVID_SCANHW_MAX values
ITC_DEVID_INTERMEC_EVIO, 238
ITC_DEVID_INTERMEC2D_IMAGER, 238
ITC_DEVID_OEM2D_IMAGER, 238
ITC_DEVID_SCANHW_NONE, 238
ITC_DEVID_SE900_LASER, 238
ITC_DEVID_SE900HS_LASER, 238

ITC_DEVID_SCANHW_NONE, 238
ITC_DEVID_SE900_LASER, 238
ITC_DEVID_SE900HS_LASER, 238
ITC_DEVID_WANRADIO_NONE, 238
ITC_DEVID_WANRADIO_SIEMENS_MC45, 238
ITC_DEVID_WANRADIO_SIERRA_SB555, 238
ITC_DEVID_WANRADIO_XIRCOM_GEM3503, 238
ITC_IFTP_STOP, 233
ITC_KEYBOARD_CHANGE, CreateEvent(), 279
ITC_NVPARM_80211_INSTALLED, 238
ITC_NVPARM_80211_RADIOTYPE, 238
ITC_NVPARM_ANTENNA_DIVERSITY, 237
ITC_NVPARM_BLUETOOTH_INSTALLED, 239
ITC_NVPARM_CONTRAST, 237
ITC_NVPARM_DISPLAY_TYPE, 237
ITC_NVPARM_ECN, 237
ITC_NVPARM_EDBG_SUBNET, 237
ITC_NVPARM_EDG_IP, 237
ITC_NVPARM_ETHERNET_ID, 236
ITC_NVPARM_INTERMEC_DATACOLLEC-
TION_HW, 238
ITC_NVPARM_INTERMEC_DATACOLLEC-
TION_SW, 238
ITC_NVPARM_INTERMEC_SOFTWARE_CON-
TENT, 237
ITC_NVPARM_LAN9000_INSTALLED, 239
ITC_NVPARM_MANF_DATE, 236
ITC_NVPARM_MCODE, 237
ITC_NVPARM_RTC_RESTORE, 238
ITC_NVPARM_SERIAL_NUM, 236
ITC_NVPARM_SERIAL2_INSTALLED, 239
ITC_NVPARM_SERVICE_DATE, 236
ITC_NVPARM_SIM_PROTECT_HW_INSTALLED,
239
ITC_NVPARM_SIM_PROTECT_SW_INSTALLED,
239
ITC_NVPARM_VERSION_NUMBER, 237
ITC_NVPARM_VIBRATE_INSTALLED, 239
ITC_NVPARM_WAN_FREQUENCY, 238
ITC_NVPARM_WAN_INSTALLED, 238
ITC_NVPARM_WAN_RADIOTYPE, 238
ITC_NVPARM_WAN_RI, 237
ITC_REGISTRY_SAVE_ENABLE, 241
ITU-T interface specifications, 169

K

Keeping a to-do list, via Tasks, 69
KernelIoControl
IOCTL_GET_CPU_ID, 250
IOCTL_HAL_COLDBOOT, 245, 276
IOCTL_HAL_GET_BOOT_DEVICE, 247
IOCTL_HAL_GET_BOOTLOADER_VERINFO,
244
IOCTL_HAL_GET_DEVICE_INFO, 235
IOCTL_HAL_GET_DEVICEID, 242
IOCTL_HAL_GET_OAL_VERINFO, 243
IOCTL_HAL_GET_RESET_INFO, 246
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
IOCTL_HAL_REBOOT, 248, 276
IOCTL_HAL_WARMBOOT, 245, 276
IOCTL_PROCESSOR_INFORMATION, 249
Key clicks
bar code configuration, 383, 388
configuration parameter, 383
Key sequences
alpha (blue) keys
alphanumeric, 15
numeric, 14
[gold] keys, numeric, 12
[gold/white] keys, alphanumeric, 13
Keyboard
See also Keypad
Windows Mobile input panel, 31
Keypad
advanced remapping, 279
alphanumeric
alpha (blue) key sequences, 15
[gold/white] key sequences, 13
scan codes, 281
backlight control panel applet, 11
change notification, 279
driver registry settings, 279
numeric
alpha (blue) key sequences, 14
[gold] key sequences, 12
scan codes, 280
planes, 277
remapping, 277
sample registry keys, 283

L

Laser scanner
configuration parameters, 286
data collection parameters
beeper frequency, 321
beeper volume, 320
codabar, 292
codabar user ID, 311
code 11, 306
code 11 user ID, 315
code 128, 295
code 128 FNC1 character, 297
code 128 user ID, 311
code 39, 290
code 39 user ID, 311
code 93, 294
code 93 length, 294
code 93 user ID, 312
EAN-13 user ID, 314
EAN-8 user ID, 314
good read beep duration, 323
good read beeps, 322
interleaved 2 of 5, 303
interleaved 2 of 5 user ID, 312
macro PDF, 300
matrix 2 of 5, 304
matrix 2 of 5 user ID, 314
micro PDF417, 302
MSI, 299

MSI user ID, 312
PDF417, 300
PDF417 user ID, 312
plessey, 298
plessey user ID, 313
prefix, 316
standard 2 of 5, 291
standard 2 of 5 user ID, 313
suffix, 317
telepen, 305
telepen user ID, 314
UPC-E user ID, 313
UPC-A user ID, 313
UPC/EAN, 293
virtual wedge, 329
virtual wedge code page, 335
virtual wedge grid, 334
virtual wedge postamble, 332
virtual wedge preamble, 330
SNMP configuration parameters
identification contact, 345
identification location, 347
identification name, 346
security encryption key, 342
security read encryption, 340
security read-only community string, 338
security read/write community string, 339
security subnet mask, 310
security write encryption, 341
trap authentication, 343
trap threshold, 344
supported
beeper functions, 318
symbologies, 289
symbologies not available
Datamatrix, 307
datamatrix, 308
maxicode, 309
LEAP
802.11 radio module
network EAP, 380
WPA encryption, 379
profile security information, 378
WEP encryption, 378
Letter recognizer, Windows Mobile input panel, 33
Library, Microsoft Reader, 91
Lighting Mode, configuration parameter, 327
Line printing, 182
lpBytesReturned
IOCTL_GET_CPU_ID, 250
IOCTL_HAL_GET_BOOT_DEVICE, 247
IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
IOCTL_HAL_GET_DEVICE_INFO, 235
IOCTL_HAL_GET_DEVICEID, 242
IOCTL_HAL_GET_OAL_VERINFO, 243
IOCTL_HAL_GET_RESET_INFO, 246
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
IOCTL_PROCESSOR_INFORMATION, 249
lpInBuf
IOCTL_GET_CPU_ID, 250
IOCTL_HAL_COLDBOOT, 245
IOCTL_HAL_GET_BOOT_DEVICE, 247
IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
IOCTL_HAL_GET_DEVICE_INFO, 235
IOCTL_HAL_GET_DEVICEID, 242
IOCTL_HAL_GET_OAL_VERINFO, 243
IOCTL_HAL_GET_RESET_INFO, 246
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
IOCTL_HAL_REBOOT, 248
IOCTL_HAL_WARMBOOT, 245
IOCTL_PROCESSOR_INFORMATION, 249
lpInBufSize
IOCTL_GET_CPU_ID, 250
IOCTL_HAL_COLDBOOT, 245
IOCTL_HAL_GET_BOOT_DEVICE, 247
IOCTL_HAL_GET_DEVICE_INFO, 235
IOCTL_HAL_GET_DEVICEID, 242
IOCTL_HAL_GET_OAL_VERINFO, 243
IOCTL_HAL_GET_RESET_INFO, 246
IOCTL_HAL_REBOOT, 248
IOCTL_HAL_WARMBOOT, 245
lpOutBuf
IOCTL_GET_CPU_ID, 250
IOCTL_HAL_COLDBOOT, 245
IOCTL_HAL_GET_BOOT_DEVICE, 247
IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
IOCTL_HAL_GET_DEVICE_INFO, 235
IOCTL_HAL_GET_DEVICEID, 242
IOCTL_HAL_GET_OAL_VERINFO, 243
IOCTL_HAL_GET_RESET_INFO, 246
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
IOCTL_HAL_REBOOT, 248
IOCTL_HAL_WARMBOOT, 245
IOCTL_PROCESSOR_INFORMATION, 249
LPT9 printer device, 183

M

Macro PDF, configuration parameter, 300
Managing e-mail messages and folders, via Inbox, 74
ManifestName, FTP Server, 229
Matrix 2 of 5, 396
configuration parameter, 304
user ID, 314
MaxiCode, 396
configuration parameter, 309
Meetings
Calendar, sending a request, 57
via Calendar, 46
Menus, Windows Mobile settings, 41
Messages
sending to, contacts, 66
via Inbox
composing/sending, 77
downloading from server, 76

MIBs
ASN.1, 179
files, 179
object identifier, 180
OIDs, **180**
Micro PDF417, 397
configuration parameter, 302
Microphone, 4
phone application, 166
Microsoft Developer Network Library. *See* MSDN library
Microsoft Exchange e-mail account, 84
Microsoft Passport account, 84
Microsoft Reader
books
downloading, 90
reading, 92
removing, 93
features, 93
adding bookmarks, 93
adding drawings, 93
annotations index, 93
attaching notes, 93
copying text, 93
highlighting text, 93
searching for text, 93
using the library, 91
Windows Mobile, 90
Microsoft's Wireless Zero Config, 361
Migrating applications
Flash File Store, 122
CompactFlash storage cards, 122
Secure Digital storage cards, 122
Migrating to a 700 Color Computer, 124
Mobile Favorites, Pocket Internet Explorer, 95
Mobile Favorites folder, Pocket Internet Explorer, 94
Modem position, COM A, 354
Modems, creating a connection
to an ISP, 98
to work, 106
MP3 files, Windows Media Player, 89
MSDN library, 233
MSDN Windows CE documentation, 178
MSI, 398
configuration parameter, 299
user ID, 312
MSN Messenger
about, 84
accounts
Microsoft Exchange e-mail, 84
Microsoft Passport, 84
contacts
managing, 87
sending messages, 88
working with, 86
setting up an account, 85
using My Text, 39
MultiMediaCards, card support, 19

N

nDeviceId, NLEDGetDeviceInfo, 275
NDIS_ENCRYPTION_1_ENABLED
EncryptionStatus(), 264
GetWepStatus(), 261
NDIS_ENCRYPTION_1_KEY_ABSENT
EncryptionStatus(), 264
GetWepStatus(), 261
NDIS_ENCRYPTION_2_ENABLED
EncryptionStatus(), 264
GetWepStatus(), 261
NDIS_ENCRYPTION_2_KEY_ABSENT
EncryptionStatus(), 264
GetWepStatus(), 261
NDIS_ENCRYPTION_3_ENABLED
EncryptionStatus(), 264
GetWepStatus(), 261
NDIS_ENCRYPTION_3_KEY_ABSENT
EncryptionStatus(), 264
GetWepStatus(), 261
NDIS_ENCRYPTION_DISABLED
EncryptionStatus(), 264
GetWepStatus(), 261
NDIS_ENCRYPTION_NOT_SUPPORTED
EncryptionStatus(), 264
GetWepStatus(), 261
NDIS_MIXED_CELL_OFF, SetMixedCellMode(), 267
NDIS_MIXED_CELL_ON, SetMixedCellMode(), 267
NDIS_NET_AUTO_UNKNOWN
GetNetworkMode(), 257
SetNetworkMode(), 266
NDIS_NET_MODE_ESS
GetNetworkMode(), 257
SetNetworkMode(), 266
NDIS_NET_MODE_IBSS
GetNetworkMode(), 257
SetNetworkMode(), 266
NDIS_NET_MODE_UNKNOWN
GetNetworkMode(), 257
SetNetworkMode(), 266
NDIS_NET_TYPE_DS, GetNetworkType(), 258
NDIS_NET_TYPE_FH, GetNetworkType(), 258
NDIS_NET_TYPE_OFDM_2_4G
GetNetworkMode(), 257
SetNetworkMode(), 266
NDIS_NET_TYPE_OFDM_5G
GetNetworkMode(), 257
SetNetworkMode(), 266
NDIS_NET_TYPE_UNDEFINED, GetNetworkType(), 258
NDIS_NETWORK_EAP_MODE_OFF
GetCCXStatus(), 262
SetCCXStatus(), 267
NDIS_NETWORK_EAP_MODE_ON
GetCCXStatus(), 262
SetCCXStatus(), 267
NDIS_POWER_LEVEL_1, GetTXPower(), 260
NDIS_POWER_LEVEL_15, GetTXPower(), 260
NDIS_POWER_LEVEL_30, GetTXPower(), 260
NDIS_POWER_LEVEL_5, GetTXPower(), 260
NDIS_POWER_LEVEL_63, GetTXPower(), 260

NDIS_POWER_LEVEL_UNKNOWN, GetTXPower(), 260
NDIS_RADIO_ASSOCIATED, GetAssocationStatus(), 254
NDIS_RADIO_AUTH_MODE_AUTO
GetAuthenticationMode(), 255
SetAuthenticationMode(), 265
NDIS_RADIO_AUTH_MODE_ERROR
GetAuthenticationMode(), 255
SetAuthenticationMode(), 265
NDIS_RADIO_AUTH_MODE_OPEN
GetAuthenticationMode(), 255
SetAuthenticationMode(), 265
NDIS_RADIO_AUTH_MODE_SHARED
GetAuthenticationMode(), 255
SetAuthenticationMode(), 265
NDIS_RADIO_AUTH_MODE_WPA
GetAuthenticationMode(), 255
SetAuthenticationMode(), 265
NDIS_RADIO_AUTH_MODE_WPA_NONE
GetAuthenticationMode(), 255
SetAuthenticationMode(), 265
NDIS_RADIO_AUTH_MODE_WPA_PSK
GetAuthenticationMode(), 255
SetAuthenticationMode(), 265
NDIS_RADIO_POWER_AUTO
GetPowerMode(), 259
SetPowerMode(), 266
NDIS_RADIO_POWER_MODE_CAM
GetPowerMode(), 259
SetPowerMode(), 266
NDIS_RADIO_POWER_MODE_FAST_PSP
GetPowerMode(), 259
SetPowerMode(), 266
NDIS_RADIO_POWER_MODE_PSP
GetPowerMode(), 259
SetPowerMode(), 266
NDIS_RADIO_POWER_UNKNOWN
GetPowerMode(), 259
SetPowerMode(), 266
NDIS_RADIO_SCANNING, GetAssociationStatus(), 254
NDIS_SUPP_LOGGING_OFF, EnableSuppLogging(), 272
NDIS_SUPP_LOGGING_ON, EnableSuppLogging(), 272
NDISUIO_PACKET_DRIVER, SwitchPacketDriver(), 273
Network adapters
802.11b, 129
antenna color code, 127
Ethernet communications, 128
no networking, 132
wireless printing, 174
WWAN radio options, 134
Network EAP
LEAP security method, 380
PEAP security method, 369
Network type, 802.11 radio module, 362
nInBufSize
IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
IOCTL_PROCESSOR_INFORMATION, 249
nInfoId, NLEDGetDeviceInfo, 275
NLED driver, vibrator, 274
NLED_COUNT_INFO, NLEDGetDeviceInfo, 275
NLED_SETTINGS_INFO_ID
NLEDGetDeviceInfo, 275
NLEDSetDevice, 275
NLED_SUPPORTS_INFO_ID, NLEDGetDeviceInfo, 275
Notes
adding to
appointments, 54
contacts, 64
creating a note, 71
drawing on the screen, 37
creating a drawing, 37
selecting a drawing, 37
Pocket Outlook, 71
recording a message, 38
Start menu icon, 27
synchronizing notes, 72
writing on the screen, 34
alternate writing, 35
converting writing to text, 35
selecting the writing, 34
tips for good recognition, 36
nOutBufSize
IOCTL_GET_CPU_ID, 250
IOCTL_HAL_COLDBOOT, 245
IOCTL_HAL_GET_BOOT_DEVICE, 247
IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
IOCTL_HAL_GET_DEVICE_INFO, 235
IOCTL_HAL_GET_DEVICEID, 242
IOCTL_HAL_GET_OAL_VERINFO, 243
IOCTL_HAL_GET_RESET_INFO, 246
IOCTL_HAL_ITC_READ_PARM, 236
IOCTL_HAL_ITC_WRITE_SYSPARM, 240
IOCTL_HAL_REBOOT, 248
IOCTL_HAL_WARMBOOT, 245
IOCTL_PROCESSOR_INFORMATION, 249

NPCP printing, 183
about, 183
closing driver, 184
COM1 parameters, 183
communications, 186
driver I/O controls, 185
installation, 183
LPT9, 183
opening driver, 184
reading from driver, 184
removal, 183
sample code, 186
unit information control panel, NPCPTEST CAB file, 351
writing to driver, 184
Numeric keypad
alpha (blue) key sequences, 14
[gold] key sequences, 12
registry settings
alpha plane, 278
gold plane, 278
unshifted plane, 278
scan codes, 280

O

O'Neil printing
See also DTR printer
installing driver, 188
Object Store, packaging an application, 118
Object store
IOCTL_HAL_COLDBOOT, 245
IOCTL_HAL_REBOOT, 248
IOCTL_HAL_WARMBOOT, 245
Oldstyle device ID, 242
Opening drivers
DTR, 189
NPCP, 184
Owner information, Windows Mobile settings, 41

P

Packaging an application
CompactFlash storage cards, 118
Flash File Store, 118
Object Store, 118
Persistent Storage Manager, 118
Secure Digital storage cards, 118
Page format printing, 182
Password
Pocket Excel, 82
Windows Mobile settings, 41
PauseAtStartup, FTP Server, 229
PB20 printers, printer support, 181
PDF417, 396
about the laser scanner, 191
configuration parameter, 300
user ID, 312
PEAP
802.11 radio module
network EAP, 369
WPA encryption, 368
profile security information, 367
WEP encryption, 367
Performing a cold boot, 17
Persistent Storage Manager. *See* PSM
Phone jack position, control panel applet, 354
PhoneUtility, 166
ring, 167
vibrate, 167
pInput, NLEDSetDevice, 275
Planes, keypad, 277
Plessey, 397
configuration parameter, 298
user ID, 313
Pocket Excel
about, 82
creating a workbook, 82
Pocket Internet Explorer
about, 94
adding programs, 42
AvantGo channels, 96
browsing the Internet, 97
favorite links, 94
getting connected, 98
mobile favorites, 95
Mobile Favorites folder, 94
software build, 18
Start menu icon, 27
viewing mobile favorites and channels, 97
Pocket Outlook, 46
Calendar, 46
Pocket PC
IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
IOCTL_HAL_GET_OAL_VERINFO, 243
Pocket Word
about, 78
creating a document, 78
drawing mode, 81
recording mode, 80
synchronizing, 81
tips, 83
typing mode, 79
writing mode, 80
POP3, Folder behavior connected to e-mail server, 74
Postamble
configuration parameter, 332
with/without data, 333
pOutput, NLEDGetDeviceInfo, 275
Power
control panel
battery status, 5
RAM maintenance, 6
Windows Mobile settings, 41
Preamble
configuration parameter, 330
with/without data, 331
Prefix, configuration parameter, user ID, 316

Printer support, 182
IrDA printer driver, 182
NPCP printer driver, 183
O'Neil printer driver, 188
Processor information, IOCTL_PROCESSOR_INFORMATION, 249
Profile label, 802.11 radio module, 362
Profiles
802.11 radio module, 361
advanced settings, 381
basic information, 362
security information, 363
editing, 361
Programs, adding or removing, Windows Mobile, 41
PSM
determining build version, 16
packaging an application, 118
PSM build, 288

Q

QR Code, 398
QR code, configuration parameter, 307
Query Information functions, 254
Quick Response Code. *See* QR Code
Quick Response code. *See* QR code

R

Radios
See also Network adapters
card support, 21
Reader commands, 384
configuration change, 384
date and time settings, 385
Reading from drivers, NPCP, 184
Real-Time Clock, restore after cold-boot, 355
Reboot methods
IOCTL_HAL_COLDBOOT, 276
IOCTL_HAL_REBOOT, 276
IOCTL_HAL_WARMBOOT, 276
Record button, recording a message, 38
Recording, via Notes, 38
Recording a message, Pocket Word, 80
Recording mode, Pocket Word, 80
Recovery CD
AutoCab method, 124
AUTOUSER.DAT file, 123
RegFlushKey() API, 225
S9C upgrade, 352
updating the system software, 121
Recurrence pattern, Calendar, 49
RegFlush utility, 123
Registry
FTP Server parameters, 228
keypad remapping, 279
sample view of key mapping, 283
save location, IOCTL_HAL_ITC_WRITE_SYSPARM, 240
writing to a storage card, 123
Registry Save, control panel applet, 355
Registry settings
AutoCfg, 178
AutoFTP, 234
AutoInterval, 178
AutoIP/DHCP, 178
DhcpMaxRetry, 178
DhcpRetryDialogue, 178
EnableDHCP, 178
keypad driver, 279
keypad planes
alpha, 278
gold, 278
unshifted, 278
Removing drivers
DTR, 188
NPCP, 183
Removing programs, Windows Mobile, 41, 43
Reset button, 17
RFC 959, 232
Root, FTP Server, 229
RTC. *See* Real-Time Clock

S

S 2 of 5. *See* Standard 2 of 5
S9C
unit information control panel, upgrade files, 352
version number, 349
Sabre 1551E or 1553
See also Tethered scanner
cabling, 199
settings, 198
Sample code, NPCP printing, 186
SB555 Watcher
activation, 140
Bell Mobility, 157
Sprint, 147
Telus, 157
Verizon, 143
WWANInit demo program, 157
copying files to computer, 137
via Microsoft ActiveSync, 138
via storage cards, 138
location of ESN, 140
Scan codes
alphanumeric keypad, 281
numeric keypad, 280
SCAN Mute, Audio control panel applet, 8
Scanner
beeper volume
selecting, 9
turning it off, 10
turning it on, 7
mute feature, turning it off, 8
unit configuration parameters
automatic shutoff, 382
backlight timeout, 383
date/time, 383
key clicks, 383
volume, 382
utilities configuration, button wakeup mask, 356

Scanner cabling, 199
Scheduling appointments and meetings, via Calendar, 46
SDK, unit information control panel, 352
SDMMC Disk, 122
Searching for text, Microsoft Reader, 93
Secure Digital cards
  card support, 19
  installing applications, 120, 121
  migrating applications, 122
  packaging an application, 118
Security, configuration parameter
  encryption key, 342
  read encryption, 340
  read-only community string, 338
  read/write community string, 339
  subnet mask, 310
  write encryption, 341
Selecting, drawing via Notes, 37
Sending and receiving messages, via Inbox, 73
Serial port, modem support, 16
Serial position, COM B, 354
Set information functions, 263
Setting date and time, 385
Setting up an e-mail account, 114
SETUP.DLL, installation functions, 220
SIM cards
  IMSI assigned
    CDMA/1xRTT, 136
    GSM/GPRS, 165
  installation status, GSM/GPRS, 165
  phone number assigned, GSM/GPRS, 164
  protection hardware, 239
  protection software, 239
  software installed, 239
Simple Network Management Protocol. *See* SNMP
SMS, Folder behavior connected to e-mail server, 74
Snap-on modems, 16
SNMP, 179
  configuration parameters
    identification contact, 345
    identification location, 347
    identification name, 346
    security encryption key, 342
    security read encryption, 340
    security read-only community string, 338
    security read/write community string, 339
    security subnet mask, 310
    security write encryption, 341
    trap authentication, 343
    trap threshold, 344
SNMP OIDs
  1D OmniDir decode enable, 328
  aimer LED duration, 324
  automatic shutoff, 382
  backlight timeout, 383
  beeper, 319
    frequency, 321
    volume, 320
  codabar, 292
    user ID, 311
code 11, 306
  user ID, 315
code 128, 295
  FNC1 character, 297
  user ID, 311
code 39, 290
  user ID, 311
code 93, 294
  length, 294
  user ID, 312
datamatrix, 308
date/time, 383
EAN
  13 user ID, 314
  8 user ID, 314
good read
  beep duration, 323
  beeps, 322
identification
  contact, 345
  location, 347
  name, 346
image dimension, 326
interleaved 2 of 5, 303
  user ID, 312
key clicks, 383
lighting mode, 327
macro PDF, 300
matrix 2 of 5, 304
  user ID, 314
maxicode, 309
micro PDF417, 302
MSI, 299
  user ID, 312
PDF417, 300
  user ID, 312
plessey, 298
  user ID, 313
prefix, 316
QR code, 307
security
  encryption key, 342
  read encryption, 340
  read-only community string, 338
  read/write community string, 339
  write encryption, 341
security subnet mask, 310
standard 2 of 5, 291
  user ID, 313
sticky aimer duration, 325
suffix, 317
telepen, 305
  user ID, 314
trap
  authentication, 343
  threshold, 344
UPC
  A user ID, 313
  E user ID, 313
UPC/EAN, 293

virtual wedge, 329
code page, 335
grid, 334
postamble, 332
preamble, 330
volume, 382
Software Developer's Kit. *See* SDK
Software versions, 18, 349
700 Series Computer, 18
unit information control panel applet, 349, 351
Speaker, 3
Speakerphone
phone application, 166
volume, 168
Sprint activation process, 147
SSID (network name), 802.11 radio module, 362
Standard 2 of 5, 398
configuration parameter, 291
user ID, 313
Start Menu, adding programs, 43
via ActiveSync, 43
via File Explorer, 43
Status icons, Windows Mobile, 26
Sticky Aimer Duration, configuration parameter, 325
Storage media, 19
Stream device driver
NPCPPORT.DLL, 183
ONEIL.DLL, 188
Suffix, configuration parameter, 317
Symbologies
internal scanner supported symbologies, **195**
scanning labels, 386
tethered scanner supported symbologies, **205**
user IDs
Codabar, 311
Code 11, 315
Code 128, 311
Code 39, 311
Code 93, 312
EAN 13, 314
EAN 8, 314
Interleaved 2 of 5, 312
Matrix 2 of 5, 314
MSI, 312
PDF417, 312
Plessey, 313
Standard 2 of 5, 313
Telepen, 314
UPC A, 313
UPC E, 313
when not available
imager, 298, 299, 300, 302, 304, 305, 306
laser scanner, 307, 308, 309
Synchronizing
AvantGo channels, 96
Calendar, 47
contacts, 61
e-mail messages, 73
favorite links, 94
mobile favorites, 95
notes, 72
Pocket Word, 81
Tasks, 70

T

Tasks
creating a task, 70
Pocket Outlook, 69
Start menu icon, 27
synchronizing, 70
TCP/IP client, DHCP server, 178
Telepen, 399
configuration parameter, 305
user ID, 314
Telus activation process, 157
Testing AT commands, 170
Tethered scanner
capabilities, 199
disabling, 197
enabling, 197
error messages, 197
limitations, 199
settings, 197
supported symbologies, **205**
Text messages, Windows Mobile, 39
Time, setting, 385
Tips for working, Pocket Excel, 83
TLS
802.1x profile
certificates, 373
WPA encryption, 372
profile security information
WEP encryption, 371
WPA encryption, 372
Today, Windows Mobile settings, 41
Today screen, Windows Mobile, 26
Tools CD
Bluetooth documentation, 174
CAB files, 120, 351
CE Imager, 122
Comm Port Wedge CAB file, 351
management tools installed on desktop, 119
MIB files, 179
sample NPCP code, 186
Wireless Printing Development Guide, 174
wireless printing sample, 352
wireless printing SDK, 174
Tracking people, via Contacts, 60
Transcriber, Windows Mobile input panel, 33
Trap configuration parameters
authentication, 343
threshold, 344
Traps, control panel appet, SNMP, 343
Troubleshooting
1551/1553 Tethered Scanners, 204
CAB Wizard, 224
TTLS
802.11 radio module, WPA encryption, 376
profile security information, WEP encryption, 375
Typing mode, Pocket Word, 79

Typing on the screen, Pocket Word, 79

U

UDP, FTPDCE, 230
UDP broadcasts, IDNATarget parameter, 229
Unit, configuration parameters
  automatic shutoff, 382
  backlight timeout, 383
  date/time, 383
  key clicks, 383
  volume, 382
Unit information
  battery status, 350
  CAB files, 351
    ActiveX control tools, 352
    Bluetooth stack, 351
    Comm Port Wedge, 351
    NPCP printer, 351
    S9C Upgrade, 352
    SDK, 352
    Windows configuration, 352
    wireless printing sample, 352
  versions, 18, 349
    700 Platform Build, 349
    DataCollection Build, 349
    S9C, 349
Unit Manager, date/time, 383
Universal Product Code. *See* UPC
Unshifted plane on keypad, regular keypad, 277
UPC, 399
  configuration parameter, 293
    A user ID, 313
    E user ID, 313
Updating, bootloader, 119
URLs
  ActiveSync, 44
  Adobe Acrobat Reader, 169
  AT command interface
    CDMA/1xRTT SB555, 169
    GPRS/GSM MC45, 169
  full screen display, 226
  MIBs, 179
  Microsoft Exchange e-mail account, 84
  Microsoft Passport account, 84
  Microsoft support, 25
  MSDN library, 233
  MSDN Windows CE documentation, 178
  Windows Mobile, 25
  Windows Mobile support, 25
Utilities control panel applet
  app launch, 357
  dock switch, 354
  registry save, 355
  wakeup mask, 356
UUID, 242

V

Verizon activation process, 143
Vibrator
  enabling, 20
  phone application, 167
  programming, 274
Video files, Windows Media Player, 89
Viewing mobile favorites and channels, Pocket Internet Explorer, 97
Virtual wedge
  bar code configuration
    grid, 389
    postamble, 389
    preamble, 389
  configuration parameter, 329
    code page, 335
    grid, 334
    postamble, 332
    preamble, 330
VN_CLASS_ASIC, 237
VN_CLASS_BOOTSTRAP, 237
VN_CLASS_KBD, 237
Volume
  bar code configuration, 382, 386
  configuration parameter, 382
  phone application, 166, 168
VPN server, creating a connection, to work, 111

W

Wakeup mask, control panel applet, 356
WAN monitor CORE module
  CDMA/1xRTT, 134
  GSM/GPRS, 164
  installing available modules, 126
  loading a module, 126
WAN radio IDs
  ITC_DEVID_WANRADIO_NONE, 238
  ITC_DEVID_WANRADIO_SIEMENS_MC45, 238
  ITC_DEVID_WANRADIO_SIERRA_SB555, 238
WAN rado CORE module
  installing available modules, 126
  loading a module, 126
WAP pages, 94
  connecting to an ISP, 98
Warm boot
  IOCTL_HAL_REBOOT, 248
  IOCTL_HAL_WARMBOOT, 245
Watcher applications
  activating, Sprint, 147
  downloading, Sprint, 147
  using
    Sprint, 150
    Verizon, 143
Web browsers, FTP support, 232
Web pages, 94
  connecting to an ISP, 98
Welch Allyn 1470 Imager
  cabling, 199
  settings, 198

WEP encryption
LEAP security method, 378
PEAP security method, 367
profile security information, 364, 365
TLS security method, 371
TTLS security method, 375
Windows CE documentation (MSDN), 178
Windows configuration, unit information control panel, WinCfg CAB file, 352
Windows Media files, Windows Media Player, 89
Windows Media Player
Start menu icon, 27
Windows Mobile, 89
Windows Mobile
ActiveSync, 44
basic skills, 26
Calendar, 46
command bar, 28
Contacts, 60
edition information, 24
getting connected, 98
Inbox, 73
MSN Messenger, 84
navigation bar, 28
Notes, 71
notifications, 29
Pocket Excel, 82
Pocket Word, 78
pop-up menus, 29
programs, 27
status icons, 26
support URLs, 25
Tasks, 69
Today screen, 26
where to find information, 25
Windows Media Player, 89
writing on the screen, 34
Wireless Network, creating a connection, to work, 108
Wireless network, 129, 361
Wireless printing
Bluetooth compatible module, 174
unit information control panel, WP_SAMPLE.CAB file, 352
Wireless TCP/IP installations, BlockSize parameter, 228
Wireless WAN
AT command interface
CDMA/1xRTT SB555, 169
GPRS/GSM MC45, 169
CDMA/1xRTT, 134
GSM/GPRS, 164
testing AT commands, 170
Work
creating
a modem connection, 106
a VPN server connection, 111
a wireless network connection, 108
an Ethernet connection, 113
getting connected, 104
WPA encryption
802.11 radio module, 366
LEAP security method, 379
PEAP security method, 368
TLS security method, 372
TTLS security method, 376
Writing mode, Pocket Word, 80
Writing on the screen
See also Notes
Pocket Word, 80
Writing to drivers
DTR, 189
NPCP, 184
WWAN. *See* Wireless WAN
WWANInit demo program, 157
creating a new connection, 159
setting up, 158
using the program, 161

X

Xscale processor ID, IOCTL_GET_CPU_ID, 250

Files Index

Numbers

80211API.DLL, 252
80211CONF.EXE, 252
80211SCAN.EXE, 252
802PM.DLL, 252

A

AUTOUSER.DAT, 120, 121

C

CABWIZ.DDF, 223
CABWIZ.EXE, 208, 223
CEIMAGER.EXE, 122
COREDLL.DLL, 274
CPL802.CPL, 252

D

DEVICEID.H, 242

E

EXITME.BIN, 232

F

FTPDCE.EXE, 230, 232
  AutoFTP, 234
  FTP Server, 227
FTPDCE.TXT, 232

I

INTERMEC.MIB, 179
ITCADC.MIB, 179
ITCSNMP.MIB, 179
ITCTERMINAL.MIB, 179

M

MAKECAB.EXE, 223
MOD80211.DLL, 252

N

NETWLAN.DLL, 252
NLED.H, 275
  NLEDGetDeviceInfo, 275
  NLEDSetDevice, 275
NPCPPORT.DLL, 183
NRINET.INI, 352

O

OEMIOCTL.H
  IOCTL_GET_CPU_ID, 250
  IOCTL_HAL_COLDBOOT, 245
  IOCTL_HAL_GET_BOOT_DEVICE, 247
  IOCTL_HAL_GET_BOOTLOADER_VERINFO, 244
  IOCTL_HAL_GET_OAL_VERINFO, 243
  IOCTL_HAL_GET_RESET_INFO, 246
  IOCTL_HAL_ITC_READ_PARM, 236
  IOCTL_HAL_ITC_WRITE_SYSPARM, 240
  IOCTL_HAL_REBOOT, 248
  IOCTL_HAL_WARMBOOT, 245
ONEIL.DLL, 188

P

PKFUNCS.H
  IOCTL_HAL_GET_DEVICEID, 242
  IOCTL_PROCESSOR_INFORMATION, 249
PRISMNDS.DLL, 252

R

REBOOTME.BIN, 232
__RESETMEPLEASE__.TXT, 220
RPM.EXE, 213
RPMCE212.INI, 213

S

SETUP.DLL, 212, 220
  DllMain, 220

T

TAHOMA.TTF, 213

U

URODDSVC.EXE, 252

W

WCESTART.INI, 213

Intermec

Corporate Headquarters
6001 36th Avenue West
Everett, Washington 98203
U.S.A.

tel 425.348.2600

fax 425.355.9551

www.intermec.com

700 Series Color Mobile Computer User's Manual - October 2004

P/N 961-054-031 REV G