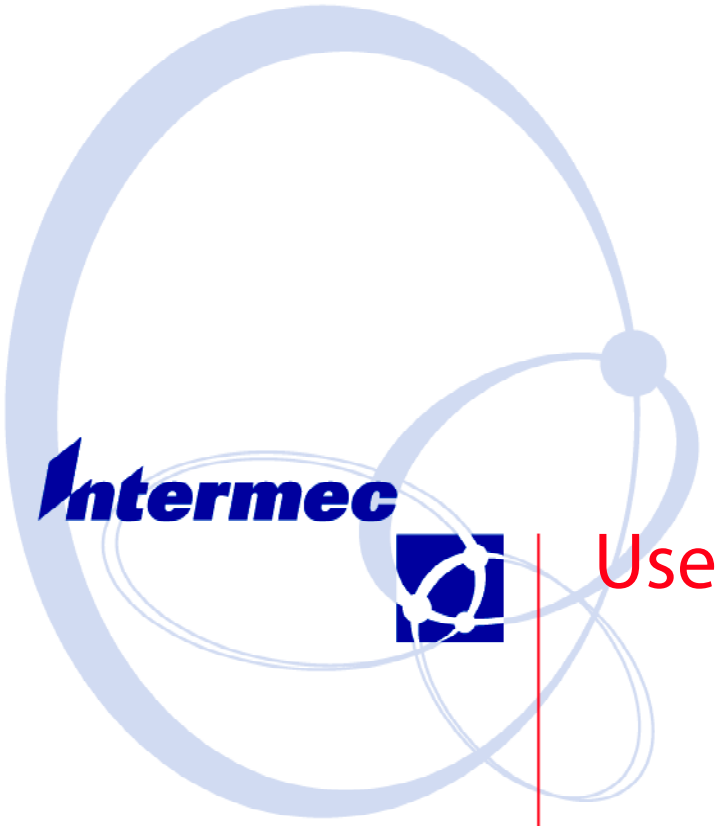


User's Manual

700 Series Color Mobile Computer

Intermec
Technologies Corporation

700 Series Color Mobile Computer

USER'S MANUAL

P/N 961-054-031
Revision A
September 2002

▶ NOTICE

The information contained herein is proprietary and is provided solely for the purpose of allowing customers to operate and service Intermec manufactured equipment and is not to be released, reproduced, or used for any other purpose without written permission of Intermec.

Disclaimer of Warranties. The sample source code included in this document is presented for reference only. The code does not necessarily represent complete, tested programs. The code is provided **"AS IS WITH ALL FAULTS." ALL WARRANTIES ARE EXPRESSLY DISCLAIMED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.**

We welcome your comments concerning this publication. Although every effort has been made to keep it free of errors, some may occur. When reporting a specific problem, please describe it briefly and include the book title and part number, as well as the paragraph or figure number and the page number.

Send your comments to:
Intermec Technologies Corporation
Publications Department
550 Second Street SE
Cedar Rapids, IA 52401

Acknowledgments

ActiveSync, ActiveX, Microsoft, Outlook, Pocket Outlook, Pocket PC, Windows, Windows NT, and the *Windows logo* are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Microsoft products are licensed to OEMs by Microsoft Licensing, Inc., a wholly owned subsidiary of Microsoft Corporation.

Bluetooth is a trademark of Bluetooth SIG, Inc., U.S.A.

CDMA2000 is a trademark of the Telecommunications Industry Association (TIA).

GSM is a registered trademark of the GSM Association.

Microclean II is a registered trademark of Foresight International.

MultiMediaCard is a trademark of Infineon Technologies AG, Germany, and is licensed to MMCA (MultiMediaCard Association).

SanDisk is a trademark of SanDisk Corporation.

Siemens is a registered trademark of Siemens AG. Siemens product names are either trademarks or registered trademarks of Siemens or Siemens AG.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

CONTENTS

SECTION 3

Installing Applications and Updating System Software

SECTION 4

Network Support

SECTION 5

Printer Support

SECTION 6

Scanner Support

SECTION 7

Programming

APPENDIX A

Control Panel Applets

APPENDIX B

Unit Manager

APPENDIX C

Bar Codes

TABLES

INDEX

Section 1
Introduction

This section introduces the 700 Series Color (700C) Mobile Computer, developed by Intermec Technologies Corporation to enhance wireless connectivity needs, and provides information about documentation and customer support.

About this Publication

This publication consists of the following information which makes up the User's Manual:

Section 2 — Pocket PC 2002
Introduces the Pocket PC 2002 operating system from Microsoft Corporation, and explains how to use its Outlook, ActiveSync, Internet Explorer, and other companion programs.

Section 3 — Installing Applications and Updating System Software
Provides methods to install applications and CAB files and to update the system software; also covers application migration.

Section 4 — Network Support
Introduces the CORE application, network adapters such as Ethernet, 802.11b radios, GSM/GPRS or CDMA/1xRTT embedded radio modules, and wireless printing equipped with a Bluetooth module, SNMP configuration, and Network Selection APIs.

Section 5 — Printer Support
Provides information on printing ASCII to either a port or to a generic serial port, and on working with IrDA, NPCP, and O'Neil printer drivers.

Section 6 — Scanner Support
Provides Automatic Data Collection COM and IImage interfaces and lists settings via Data Collection parameters.

Section 7 — Programming
Programming information that includes creating CAB files, the FTP Server, Full Screen, Kernel I/O Control Functions, Reboot Functions, and remapping the keypad.

Appendix A — Control Panel Applets
Contains detailed information about the Data Collection, SNMP, and Unit Information control panel applets.

Appendix B — Unit Manager
Describes how to configure some parameters via the Unit Manager application and includes reader commands.

Appendix C — Bar Codes
Describes some of the more common bar code symbologies and includes bar code labels that can be scanned to configure your 700 Series Computer.

About the 700 Series Color Mobile Computer

Audio System

Speaker

A speaker capable of variable volume levels is located on the back of the computer. This speaker has a transducer volume of 85 dB min at 10 CM and a frequency range of 1–8 KHz.

Internal Microphone

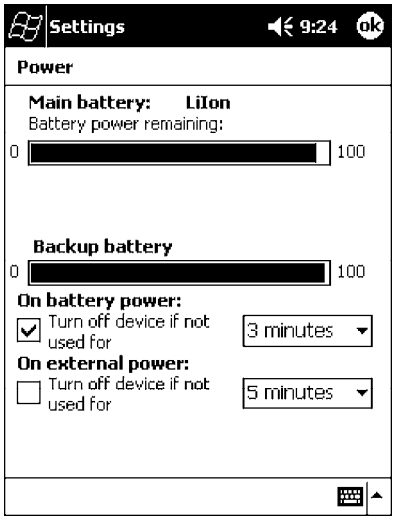
The internal microphone is located on the bottom of the unit to the left of the hirose connector.

External Headset Jack

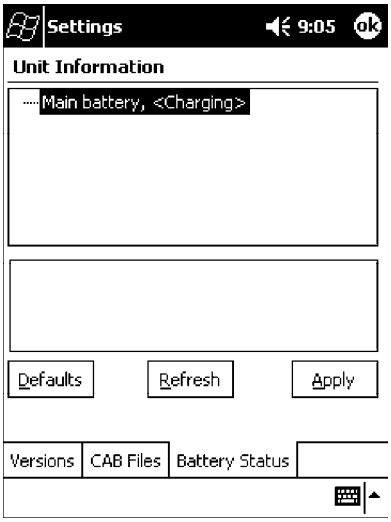
The external headset jack connects a mon headset to your mobile computer for use in noisy environments. The jack is a 2.5 mm, three-conductor jack, with autosensing of the headset jack insertion which disables the internal speaker and microphone. The external headset jack is located on the bottom of the mobile computer to the right of the hirose connector.

Battery

The 700 Series Computer comes equipped with a nominal 14.4 Watt-hour, 7.2V (2 x 2000 mAh cells), replaceable Lithium-Ion (LiIon) battery. To view the status of this battery from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Power** icon to view the current status of both the main battery and the backup battery. Tap **ok** to exit this information.



You can also view the battery status for the 700 Series Computer by accessing the **Unit Information** control panel applet. Tap the **Unit Information** icon, then tap the **Battery Status** tab to view the current status. Tap **ok** to exit this information.



Low Battery Shutdown

If your computer shuts down because of low battery conditions, your computer will not operate. This is done to ensure that data is protected. Although the battery will protect the data against loss for several hours, you should connect your computer to a power source when you first detect a low battery condition.

Your computer contains an internal super capacitor, a temporary power storage device, that protects data for up to ten minutes. This is to give you time to replace the main battery pack before that data is lost. *Be sure to put the computer in a suspend mode before doing so.*

The battery power fail level is set so that after the system shuts down in a low battery condition, there is still sufficient charge to allow the unit to remain configured, keep proper time, and maintain DRAM (Dynamic Random Access Memory) for at least 72 hours at room temperature *if* the main battery remains in the mobile computer. The configuration and time are lost if:

- The battery discharges beyond this level.
- The battery is removed when the computer is *not in suspend mode.*
- A cold reset is performed on the computer.

System Status Maintained

System status is maintained in “suspend” when the main battery is removed:

- 10 minutes for 64 MB low-power chips
- 5 minutes for 128 MB low-power chips

CAB Files Within 700C Software Tools CD

If you leave the default destination while you install the "\700 Color Mgmt Tools" directory onto your desktop PC, then "C:\Intermec\Intermec 700 Color Mgmt Tools\Cab Files" will be the default directory. There are folders within the "\Cab Files" directory that contain demos and program files. See the *Software Tools Help* for more information about these files.

Modem Support

Modem PC Cards are not supported by the 700 Series Computer. However, modem options do include the following:

- Switchable dock that includes a built-in modem and a serial port between which an application can switch.
- Mini-Landline Modem that can be tethered to the port on the bottom of the 700 Series Computer.
- Other external modems that may be connected to the bottom of the 700 Series Computer or to the dock.

Network Support

Radio CompactFlash Cards cannot be installed by a user. The 700 Series Computer must be serviced to install or replace radios. See Section 4, "*Network Support*" for more information.

- 802.11b radio
- Integrated GSM/GPRS radio
- CDPM/1xRTT radio
- Wireless printing equipped with a Bluetooth qualified module by Socket Communications

Removeable Card Support

To access either the CompactFlash (CF) or SecureDigital (SD) card slot, locate the access door at the top of the 700 Series Computer, remove its two screws, then remove the door.

CompactFlash Cards

Support is limited to one CompactFlash (CF) Storage Card in the 700 Series Computer, either for storage or for the 802.11b radio.

SecureDigital Cards

Support is limited to one SecureDigital (SD) Storage Card in the 700 Series Computer, either for storage or for the CDMA/1xRTT or GSM/GPRS radio.

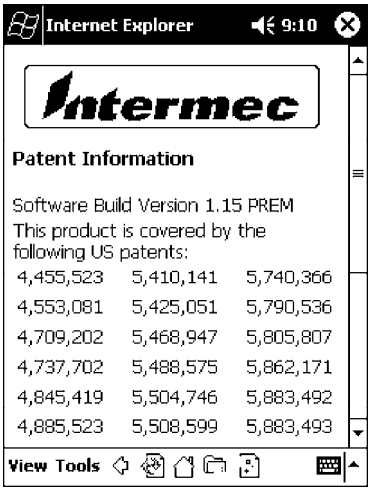
MultiMediaCards

MultiMediaCards (MMCs) are not supported in the 700 Series Computers because current technology shows that SD cards quickly surpass MMC cards in storage capacity.

Software Build Version

To check to see if your 700 Series Computer has the latest build, select **Start** → **Internet Explorer** → the **Intermec** logo.

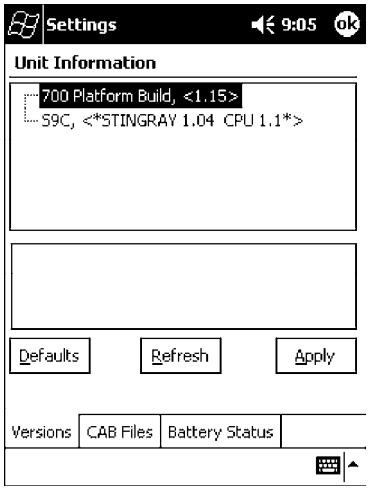
The latest software build version is displayed beneath the **Patent Information** title. This information will be useful should you need customer assistance.



► NOTE: *The **Unit Information** control panel applet is only available in the 700 Series Computer if Intermec Content is enabled, the Plus region is enabled and installed, and a laser scanner is installed.*

Unit Information

You can also view the latest software build version on your 700 Series Computer by accessing the **Unit Information** control panel applet. Select **Start** → **Settings** → the **System** tab → the **Unit Information** icon → the **Versions** tab to view the current build version on your 700 Series Computer.



Related Publications

To order printed versions of the Intermec manuals, contact your local Intermec representative or distributor. Following are related INTERMEC manuals, CD-ROMs, and part numbers (P/N):

- *700 Series Color Mobile Computer Quick Start Guide* (P/N: 962-054-053)
- *700 Series Color Tools CD* (P/N: 235-045-001)
- *Windows 95 and Windows CE Configuration Utilities Reference Manual* (P/N: 978-054-010)

Global Services and Support Center

Select any of the following services available from Intermec Technologies Corporation:

Factory Repair and On-site Repair
To request a return authorization number for one of our authorized service centers, or to request an on-site repair technician,
call 1-800-755-5505, then select option 1.

Technical Support
For technical support on your Intermec product,
call 1-800-755-5505, then select option 2.

Service Contract Status
To inquire about an existing contract, or to renew a contract,
call 1-800-755-5505, then select option 3.

Schedule Site Surveys or Installations
To schedule a site survey, or to request a product or system installation,
call 1-800-755-5505, then select option 4.

Web Support

Visit our Web site at *http://www.intermec.com* to download many of our current manuals in PDF format.

Visit our technical knowledge base (Knowledge Central) at *http://intermec.custhelp.com* to review technical information or to request technical help for all Intermec products.

Section 2

Pocket PC 2002

Congratulations on purchasing a Pocket PC. Due to the size and capabilities of this 700 Series Color Mobile Computer, you can keep your most important business and personal information up-to-date and close at hand. Microsoft ActiveSync increases the power of your 700 Series Computer by allowing you to synchronize the information on your desktop or laptop computer with your 700 Series Computer. Picture yourself in the following situations:

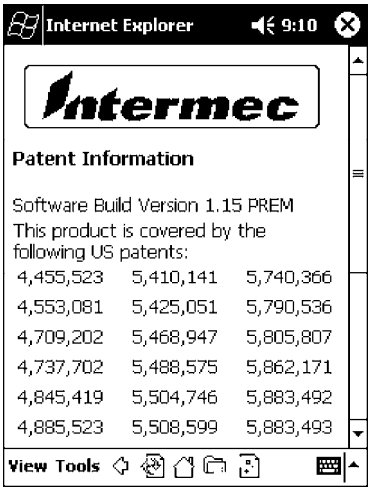
- A Calendar reminder alerts you that it is time to catch the bus. You grab your 700 Color Pocket PC Mobile Computer and catch the bus just in time. Because ActiveSync keeps the information on your 700 Series Computer up-to-date, you leisurely review your task list, make notes about the new books and CDs you want to buy, and read and respond to e-mail messages. When you get back to the office, ActiveSync transfers any task changes you made, your notes, and your e-mail message responses to your desktop computer. For more information on ActiveSync, see "*Microsoft ActiveSync*" on page 2-19.
- While walking with a colleague, your 700 Color Pocket PC Mobile Computer rings. You look at the caller ID and see it is your manager who is calling. She asks if you two are free this afternoon for an emergency meeting. While your colleague fumbles through his paper organizer, you press a button on your 700 Series Computer and instantly see a list of today's appointments and meetings. You are quickly able to tell your manager your available times, and make a note of the new meeting while on the call. You hang up, send an e-mail with a schedule request for the three of you at the desired location. For more information on scheduling appointments and meetings, see "*Microsoft Pocket Outlook"* on page 2-21.
- You are meeting your friends tonight for dinner and a movie. You download the latest movie information from the Internet to your desktop computer and then synchronize it with your 700 Series Computer. At dinner, you pull out your 700 Color Pocket PC Mobile Computer and review your movie options with your friends. For more information on downloading Web pages to your 700 Series Computer, see "*Pocket Internet Explorer*" on page 2-46.

Premium versus Professional Editions

Your 700 Series Computer will have either the Premium Edition or the Professional Edition of Pocket PC 2002. Do the following to determine which edition of Pocket PC 2002 is on your unit.

1. Select **Start** → **Internet Explorer** → the **Intermec** logo.

2. Note the "Software Build Version X.XX" information displayed beneath the **Patent Information** title. "PREM" indicates the Premium Edition and "PRO" indicates the Professional Edition.



3. Tap the **Close** icon in the top right corner to exit the Internet Explorer.

Below is a list of components provided for each edition of Pocket PC 2002.

▶ **NOTE:** *Components marked with "RAM" are provided on a Companion CD for download into RAM rather than burned into Flash ROM.*

Component	Premium Edition	Professional Edition
Microsoft ActiveSync Client *(page 2-19)*	X	X
Microsoft Pocket Outlook *(page 2-21)*	X	X
Pocket Word *(page 2-35)*	X	X
Pocket Excel *(page 2-39)*	X	X
MSN Messenger *(page 2-40)*	X	
Windows Media Player for Pocket PC *(page 2-43)*	X	RAM
Microsoft Reader *(page 2-44)*	X	RAM
Pocket Internet Explorer *(page 2-46)*	X	X

Where to Find Information

This section describes your 700 Series Computer hardware, provides an overview of the programs on your 700 Series Computer, and explains how to connect your 700 Series Computer to a desktop computer, a network, or the Internet. For instructions on setting up your 700 Series Computer and installing ActiveSync, see the Quick Start Card. The following is a guide to additional information to help you use your 700 Series Computer.

For information on:	See this source:
Programs on your mobile computer.	This section and mobile computer Help. To view Help, tap **Start → Help**.
Additional programs that can be installed on the mobile computer.	The Pocket PC Companion CD.
Connecting to and synchronizing with a desktop computer.	The Quick Start Card or ActiveSync Help on your desktop computer. To view Help, click **Help → Microsoft ActiveSync Help**.
Last-minute updates and detailed technical information.	The Read Me files, located in the Microsoft ActiveSync folder on the desktop computer and on the Pocket PC Companion CD.
Up-to-date information on your Pocket PC.	*http://www.microsoft.com/mobile/pocketpc*

Pocket PC and many of the technologies supported by the 700 Series Computer are not from Intermec Technologies. Many of the utilities and features on a Pocket PC device come directly from Microsoft without any modification from Intermec Technologies. There may be certain Microsoft-specific issues that Intermec Technologies would not be able to support, so you will have to contact Microsoft Corporation. Use the following URLs to determine your Microsoft support options:

- *http://msdn.microsoft.com/support/*
- *http://support.microsoft.com/*
- *news://news.microsoft.com* (a free support option)

Basic Skills

Learning to use your 700 Series Computer is easy. This describes the basic concepts of using and customizing your 700 Series Computer.

Buttons and Stylus

Your 700 Series Computer has hardware buttons that control actions and scroll functions, and a stylus for selecting items and entering information. On the 700 Series Computer, the stylus replaces the mouse.

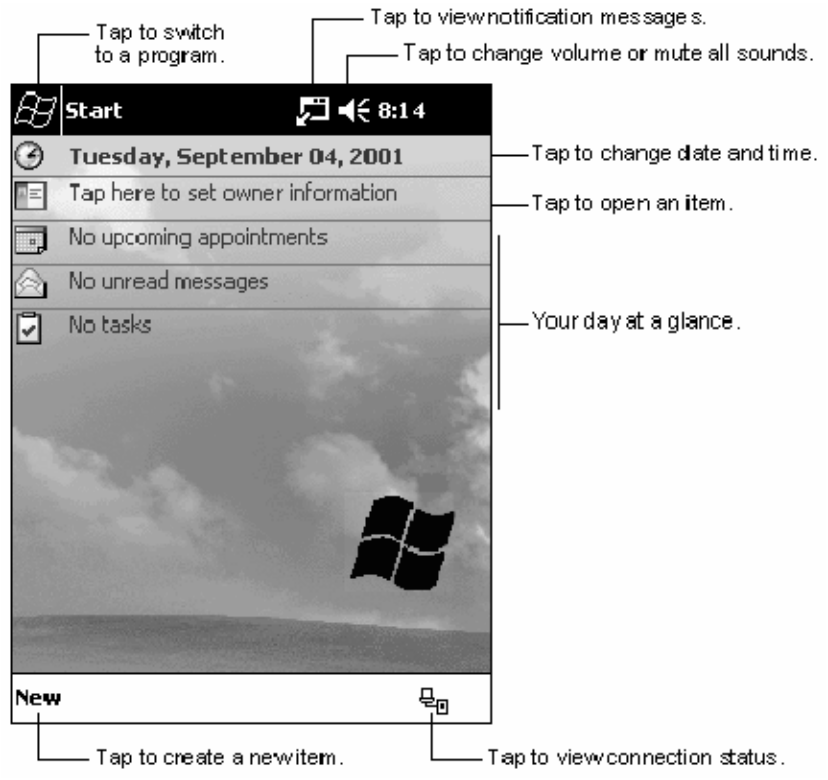
Tap:
Touch the screen once with the stylus to open items and select options.

Drag:
Hold the stylus on the screen and drag across the screen to select text and images. Drag in a list to select multiple items.

Tap and hold:
Tap and hold the stylus on an item to see a list of actions available for that item. On the pop-up menu that appears, tap the action to be performed.

Today Screen

When you turn on your 700 Series Computer for the first time each day *(or after four hours of inactivity)*, you will see the **Today** screen. You can also display it by tapping the **Start** flag *(shown left)* and then **Today**. On the **Today** screen, you can see at a glance important information for the day.



Following are some of the status icons you may see:

Status Icon	Meaning:
	Turns all sounds on and off.
	Backup battery is low.
	Main batteries are charging.
	Main batteries are low.
	Main batteries are very low.
	Main batteries are full.
	Connection is active.
	Synchronization is beginning or ending.
	Synchronization is occurring.
	Notification that one or more instant messages were received.
	Notification that one or more e-mail messages were received.

If more notification icons need to be displayed than there is room to display them, the **Notification** icon *(shown left)* will display. Tap the icon to view all notification icons.

Programs

You can switch from one program to another by selecting it from the **Start** menu. (You can customize which programs you see on this menu. For information, see "*Adjusting Settings*" on page 2-16.) To access some programs, tap **Start** → **Programs**, and then the program name.

You can also switch to some programs by pressing a program button. Your 700 Series Computer has one or more program buttons located on the front or side of the computer. The icons on the buttons identify the programs they switch to.

▶ **NOTE:** *Some programs have abbreviated labels for check boxes and drop-down menus. To see the full spelling of an abbreviated label, tap and hold the stylus on the label. Drag the stylus off the label so that the command is not carried out.*

The following is a partial list of programs that are on your 700 Series Computer. Look on the Pocket PC Companion CD for additional programs that you can install onto your 700 Series Computer.

ActiveSync
Synchronize information between your 700 Series Computer and desktop computer.

Calendar
Keep track of your appointments and create meeting requests.

Contacts
Keep track of your friends and colleagues.

Inbox
Send and receive e-mail messages.

Pocket Internet Explorer
Browse Web and WAP (Wireless Application Protocol) sites, and download new programs and files from the Internet.

Notes
Create handwritten or typed notes, drawings, and recordings.

Tasks
Keep track of your tasks.

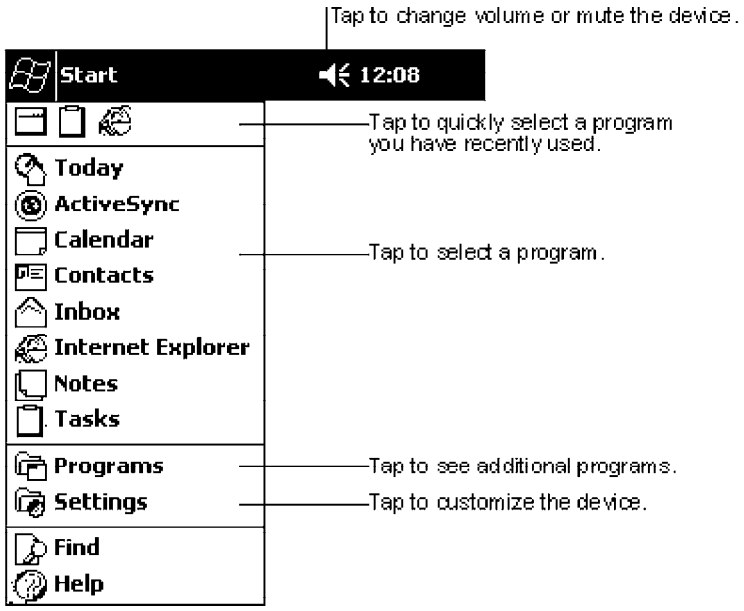
Pocket Excel
Create new workbooks or view and edit Excel workbooks created on your desktop computer.

MSN Messenger
Send and receive instant messages with your MSN Messenger contacts.

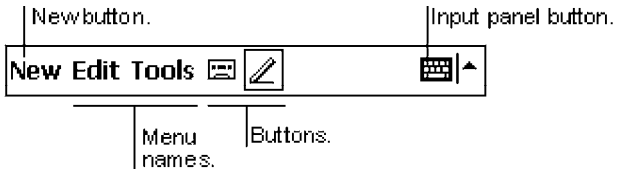
Pocket Word
Create new documents or view and edit Word documents created on your desktop computer.

Navigation Bar and Command Bar

The navigation bar is located at the top of the screen. It displays the active program and current time, and allows you to switch to programs and close screens.

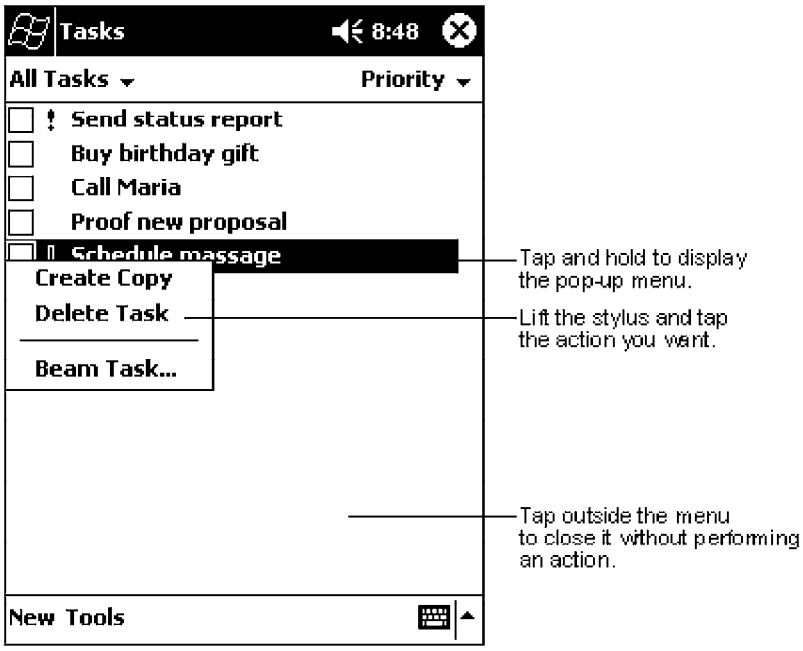


Use the command bar at the bottom of the screen to perform tasks in programs. The command bar includes menu names, buttons, and the **Input Panel** button. To create a new item in the current program, tap **New**. To see the name of a button, tap and hold the stylus on the button. Drag the stylus off the button so that the command is not carried out.



Pop-up Menus

With pop-up menus, you can quickly choose an action for an item. For example, you can use the pop-up menu in the contact list to quickly delete a contact, make a copy of a contact, or send an e-mail message to a contact. The actions in the pop-up menus vary from program to program. To access a pop-up menu, tap and hold the stylus on the item name that you want to perform the action on. When the menu appears, lift the stylus, and tap the action you want to perform. Or tap anywhere outside the menu to close the menu without performing an action.



Notifications

Your 700 Series Computer reminds you in a variety of ways when you have something to do. For example, if you have set up an appointment in Calendar, a task with a due date in Tasks, or an alarm in Clock, you will be notified in any of the following ways:

- A message box appears on the screen.
- A sound, which you can specify, is played.
- A light flashes on your 700 Series Computer.

To choose reminder types and sounds for your 700 Series Computer, tap **Start** → **Settings** → the **Personal** tab → **Sounds & Notifications**. The options you choose here apply throughout the 700 Series Computer.

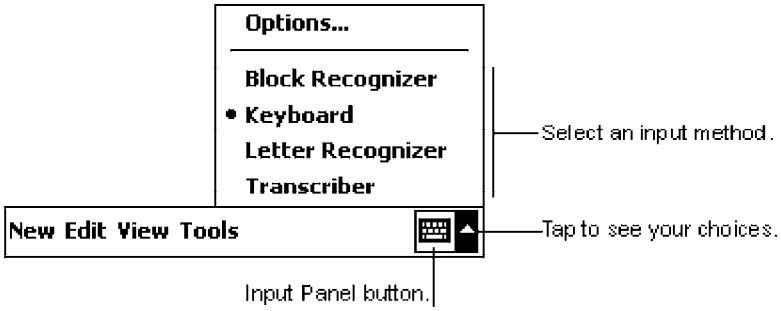
Enter Information on Your 700 Series Computer

You have several options for entering new information:

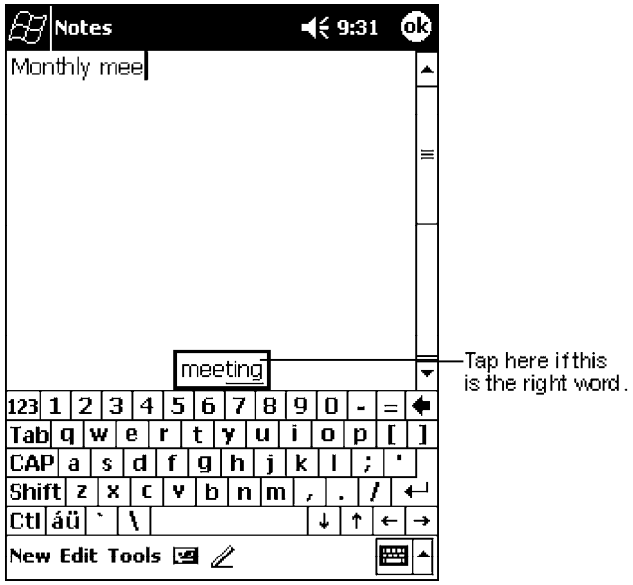
- Use the input panel to enter typed text, either by using the soft keyboard or other input method.
- Write directly on the screen.
- Draw pictures on the screen.
- Speak into your 700 Series Computer microphone to record a message.
- Use Microsoft ActiveSync to synchronize or copy information from your desktop computer to your 700 Series Computer. For more information on ActiveSync, see ActiveSync Help on your desktop computer.

Use the input panel to enter information in any program on your 700 Series Computer. You can either type using the soft keyboard or write using **Block Recognizer**, **Letter Recognizer**, or **Transcriber**. In either case, the characters appear as typed text on the screen.

To show or hide the input panel, tap the **Input Panel** button. Tap the arrow next to the **Input Panel** button to see your choices.



When you use the input panel, your 700 Series Computer anticipates the word you are typing or writing and displays it above the input panel. When you tap the displayed word, it is inserted into your text at the insertion point. The more you use your 700 Series Computer, the more words it learns to anticipate.



▶ **NOTE:** *To change word suggestion options, such as the number of words suggested at one time, tap* ***Start*** *→* ***Settings*** *→ the* ***Personal*** *tab →* ***Input*** *→ the* ***Word Completion*** *tab.*

Typing With the Soft Keyboard

1. Tap the arrow next to the **Input Panel** button, and then **Keyboard**.
2. On the soft keyboard that is displayed, tap the keys with your stylus.

Using Block Recognizer

With Block Recognizer, you can input character strokes using the stylus that are similar to those used on other 700 Series Computers.

1. Tap the arrow next to the **Input Panel** button, and then **Block Recognizer**.
2. Write a letter in the box.

When you write a letter, it is converted to typed text that appears on the screen. For specific instructions on using Block Recognizer, with Block Recognizer open, tap the question mark next to the writing area.

Using Letter Recognizer

With Letter Recognizer, you can write letters using the stylus just as you would on paper.

1. Tap the arrow next to the **Input Panel** button, and then **Letter Recognizer**.
2. Write a letter in the box.

When you write a letter, it is converted to typed text that appears on the screen. For specific instructions on using Letter Recognizer, with Letter Recognizer open, tap the question mark next to the writing area.

Using Transcriber

With Transcriber, you can write anywhere on the screen using the stylus just as you would on paper. Unlike Letter Recognizer and Block Recognizer, you can write a sentence or more of information. Then, pause and let Transcriber change the written characters to typed characters.

1. Tap the arrow next to the **Input Panel** button, and then **Transcriber**.
2. Write anywhere on the screen.

For specific instructions on using Transcriber, with Transcriber open, tap the question mark in the lower right hand corner of the screen.

Selecting Typed Text

If you want to edit or format typed text, you must select it first.

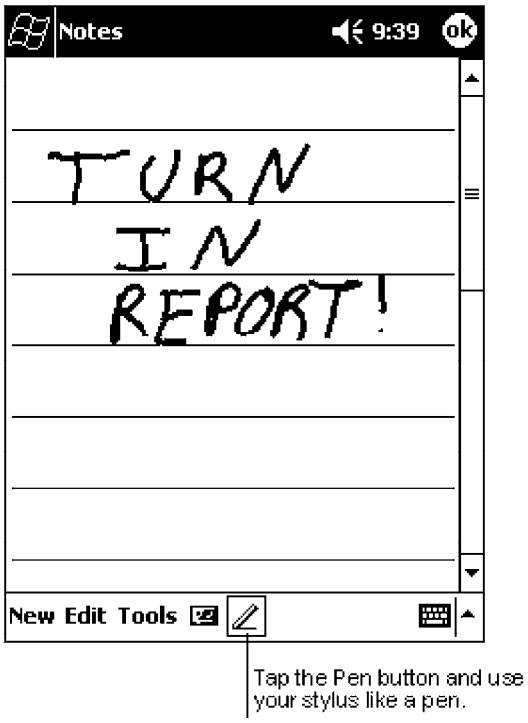
- Drag the stylus across the text you want to select.

You can cut, copy, and paste text by tapping and holding the selected words and then tapping an editing command on the pop-up menu, or by tapping the command on the **Edit** menu.

Writing on the Screen

In any program that accepts writing, such as the Notes program, and in the **Notes** tab in Calendar, Contacts, and Tasks, you can use your stylus to write directly on the screen. Write the way you do on paper. You can edit and format what you have written and convert the information to text at a later time.

- Tap the **Pen** button to switch to writing mode. This action displays lines on the screen to help you write.



▶ NOTE: *Some programs that accept writing may not have the **Pen** button. See the documentation for that program to find out how to switch to writing mode.*

Selecting the Writing

If you want to edit or format writing, you must select it first.

1. Tap and hold the stylus next to the text you want to select until the insertion point appears.
2. Without lifting, drag the stylus across the text you want to select.

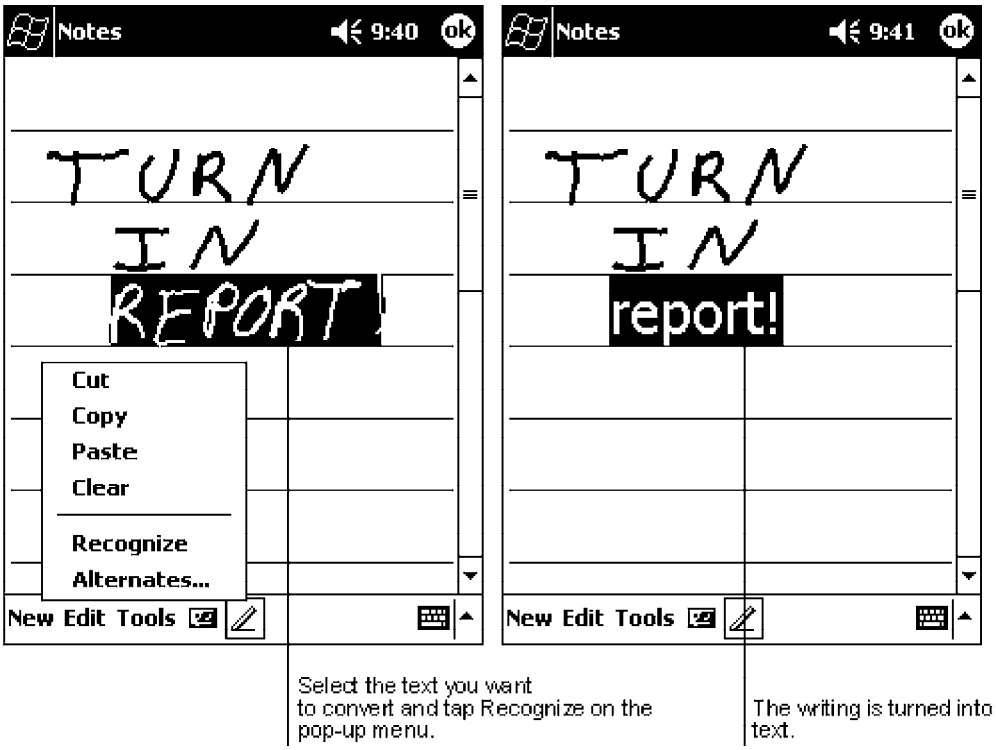
If you accidentally write on the screen, tap **Tools** → **Undo** and try again. You can also select text by tapping the **Pen** button to deselect it and then dragging the stylus across the screen.

You can cut, copy, and paste written text in the same way you work with typed text: tap and hold the selected words and then tap an editing command on the pop-up menu, or tap the command on the **Edit** menu.

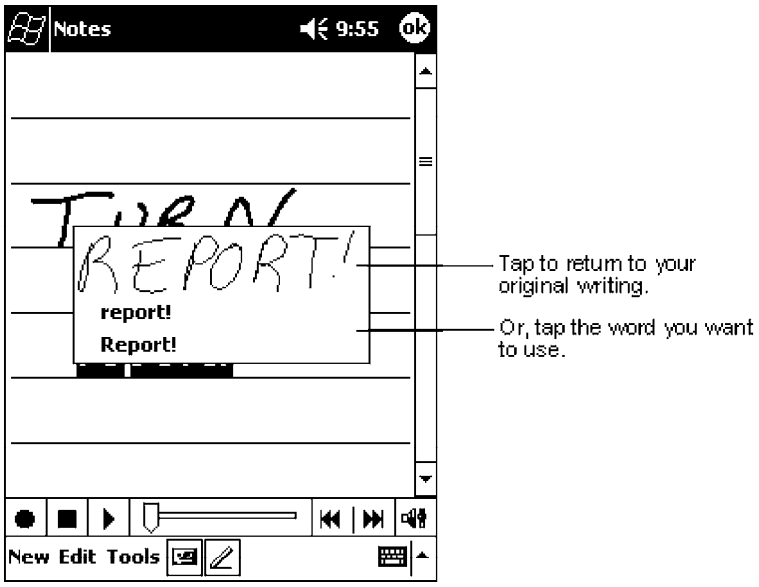
Converting Writing to Text

▶ Tap **Tools** → **Recognize**.

If you want to convert only certain words, select them before tapping **Recognize** on the **Tools** menu *(or tap and hold the selected words and then tap **Recognize** on the pop-up menu)*. If a word is not recognized, it is left as writing.



If the conversion is incorrect, you can select different words from a list of alternates or return to the original writing. To do so, tap and hold the incorrect word (tap one word at a time). On the pop-up menu, tap **Alternates**. A menu with a list of alternate words appears. Tap the word you want to use, or tap the writing at the top of the menu to return to the original writing.



Tips for getting good recognition:

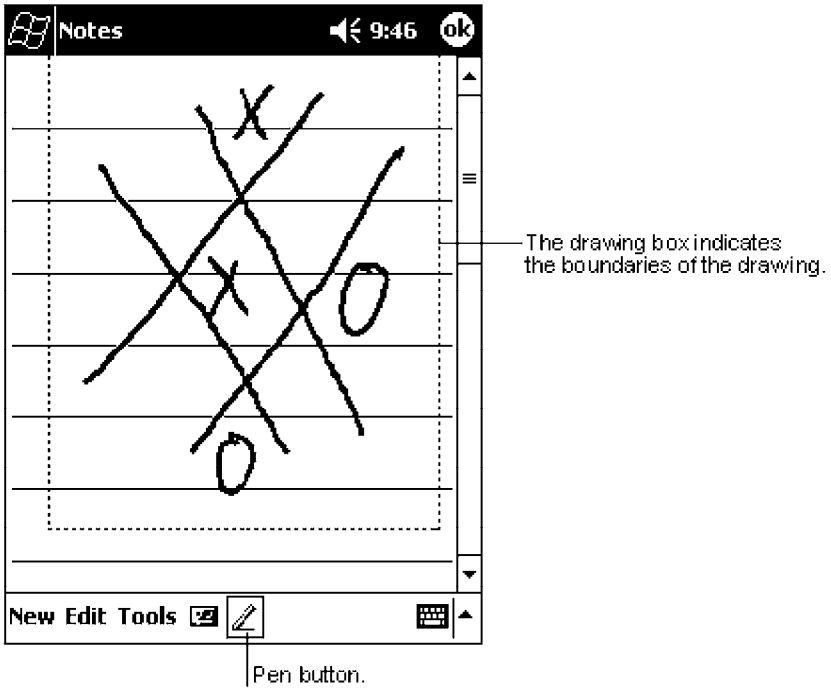
- Write neatly.
- Write on the lines and draw descenders below the line. Write the cross of the "t" and apostrophes below the top line so that they are not confused with the word above. Write periods and commas above the line.
- For better recognition, try increasing the zoom level to 300% using the **Tools** menu.
- Write the letters of a word closely and leave big gaps between words so that the 700 Series Computer can easily tell where words begin and end.
- Hyphenated words, foreign words that use special characters such as accents, and some punctuation cannot be converted.
- If you add writing to a word to change it (such as changing a "3" to an "8") after you attempt to recognize the word, the writing you add will not be included if you attempt to recognize the writing again.

Drawing on the Screen

You can draw on the screen in the same way that you write on the screen. The difference between writing and drawing on the screen is how you select items and how they can be edited. For example, selected drawings can be resized, while writing cannot.

Creating a Drawing

- Cross three ruled lines on your first stroke. A drawing box appears. Subsequent strokes in or touching the drawing box become part of the drawing. Drawings that do not cross three ruled lines will be treated as writing.



▶ **NOTE:** *You may want to change the zoom level so that you can more easily work on or view your drawing. Tap* ***Tools*** *and then a zoom level.*

Selecting a Drawing

If you want to edit or format a drawing, you must select it first.

- Tap and hold the stylus on the drawing until the selection handle appears. To select multiple drawings, deselect the **Pen** button and then drag to select the drawings you want.

You can cut, copy, and paste selected drawings by tapping and holding the selected drawing and then tapping an editing command on the pop-up menu, or by tapping the command on the **Edit** menu. To resize a drawing, make sure the **Pen** button is not selected, and drag a selection handle.

Recording a Message

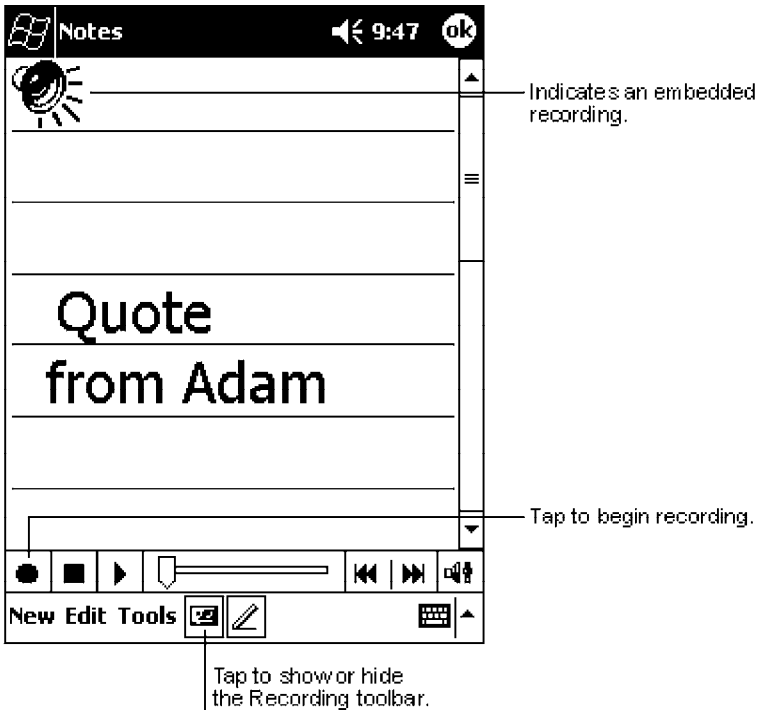
In any program where you can write or draw on the screen, you can also quickly capture thoughts, reminders, and phone numbers by recording a message. In Calendar, Tasks, and Contacts, you can include a recording in the **Notes** tab. In the Notes program, you can create a stand-alone recording or include a recording in a written note. If you want to include the recording in a note, open the note first. In the Inbox program, you can add a recording to an e-mail message.

Creating a Recording

1. Hold your computer's microphone near your mouth or source of sound.
2. Press and hold the **Record** hardware button on your 700 Series Computer until you hear a beep.
3. While holding down the **Record** button, make your recording.
4. To stop recording, release the **Record** button. Two beeps will sound. The new recording appears in the note list or as an embedded icon.

▶ **NOTE:** *You can also make a recording by tapping the **Record** button on the Recording toolbar.*

To play a recording, tap it in the list or tap its icon in the note.



Using My Text

When using Inbox or MSN Messenger, use **My Text** to quickly insert preset or frequently used messages into the text entry area. To insert a message, tap **My Text** and tap a message.

▶ **NOTE:** *You can add text after inserting a **My Text** message before sending it.*

To edit a My Text message, in the **Tools** menu, tap **Edit → My Text Messages**. Select the message you wish to edit and make desired changes.

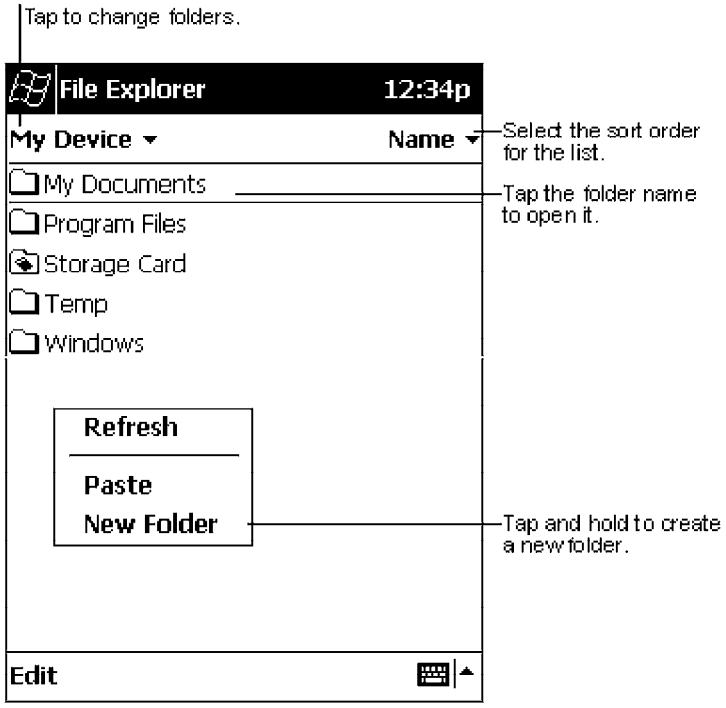
Finding and Organizing Information

The **Find** feature on your 700 Series Computer helps you quickly locate information.

Tap **Start → Find**. Enter the text you want to find, select a data type, and then tap **Go** to initiate the search.

▶ **NOTE:** *To quickly find information that is taking up storage space on your 700 Series Computer, select **Larger than 64 KB** in **Type**.*

You can also use the File Explorer to find files on your 700 Series Computer and to organize these files into folders. Tap **Start** → **Programs** → **File Explorer**.



▶ **NOTE:** *You can move files in File Explorer by tapping and holding the item you want to move, and then tapping **Cut** or **Copy** and **Paste** on the pop-up menu.*

Customizing Your 700 Series Computer

You can customize your 700 Series Computer by adjusting settings and installing additional software.

Adjusting Settings

You can adjust settings to suit the way you work. To see available options, tap **Start** → **Settings** → either the **Personal** tab or the **System** tab located at the bottom of the screen. You might want to adjust the following:

Clock:
To change the time or to set alarms.

Menus:
To customize what appears on the **Start** menu, and to enable a pop-up menu from the **New** button.

Owner Information:
To enter your contact information.

Password:
To limit access to your 700 Series Computer.

Power:
To maximize battery life.

Today:
To customize the look and the information that is displayed on the **Today** screen.

Adding or Removing Programs

Programs added to your 700 Series Computer at the factory are stored in ROM (Read Only Memory). You cannot remove this software, and you will never accidentally lose ROM contents. ROM programs can be updated using special installation programs with a *.XIP extension. All other programs and data files added to your 700 Series Computer after factory installation are stored in RAM (Random Access Memory).

You can install any program created for your 700 Series Computer, as long as your 700 Series Computer has enough memory. The most popular place to find software for your 700 Series Computer is on the Pocket PC Web site (*http://www.microsoft.com/mobile/pocketpc*).

Adding Programs Using ActiveSync

You will need to install the appropriate software for your 700 Series Computer on your desktop computer before installing it on your 700 Series Computer.

1. Determine your 700 Series Computer and processor type so that you know which version of the software to install. Tap **Start** → **Settings** → the **System** tab → **About** → the **Version** tab, then make a note of the information in **Processor**.
2. Download the program to your desktop computer (or insert the CD or disk that contains the program into your desktop computer). You may see a single *.XIP, *.EXE, or *.ZIP file, a SETUP.EXE file, or several versions of files for different 700 Series Computer types and processors. Be sure to select the program designed for the Pocket PC and your 700 Series Computer processor type.
3. Read any installation instructions, Read Me files, or documentation that comes with the program. Many programs provide special installation instructions.
4. Connect your 700 Series Computer and desktop computer.
5. Double-click the *.EXE file.
   - If the file is an installer, the installation wizard will begin. Follow the directions on the screen. Once the software has been installed on your desktop computer, the installer will automatically transfer the software to your 700 Series Computer.
   - If the file is not an installer, you will see an error message stating that the program is valid but it is designed for a different type of computer. You will need to move this file to your 700 Series Computer. If you cannot find any installation instructions for the program in the Read Me file or documentation, use ActiveSync Explore to copy the program file to the Program Files folder on your 700 Series Computer. For more information on copying files using ActiveSync, see ActiveSync Help.

Once installation is complete, tap **Start** → **Programs**, and then the program icon to switch to it.

Adding a Program Directly from the Internet

1. Determine your 700 Series Computer and processor type so that you know which version of the software to install. Tap **Start** → **Settings** → the **System** tab → **About** → the **Version** tab, then make a note of the information in **Processor**.
2. Download the program to your 700 Series Computer straight from the Internet using Pocket Internet Explorer. You may see a single *.XIP, *.EXE, or *.ZIP file, a SETUP.EXE file, or several versions of files for different 700 Series Computer types and processors. Be sure to select the program designed for the Pocket PC and your 700 Series Computer processor type.
3. Read any installation instructions, Read Me files, or documentation that comes with the program. Many programs provide installation instructions.
4. Tap the file, such as a *.XIP or *.EXE file. The installation wizard will begin. Follow the directions on the screen.

Adding a Program to the Start Menu

Tap **Start** → **Settings** → **Menus** → the **Start Menu** tab, and then the check box for the program. If you do not see the program listed, you can either use File Explorer on the 700 Series Computer to move the program to the **Start Menu** folder, or use ActiveSync on the desktop computer to create a shortcut to the program and place the shortcut in the **Start Menu** folder.

Using File Explorer on the 700 Series Computer:
Tap **Start** → **Programs** → **File Explorer**, and locate the program (tap the folder list, labeled **My Documents** by default, and then **My Device** to see a list of all folders on the 700 Series Computer). Tap and hold the program and tap **Cut** on the pop-up menu. Open the **Start Menu** folder located in the Windows folder, tap and hold a blank area of the window, and tap **Paste** on the pop-up menu. The program will now appear on the **Start** menu. For more information on using File Explorer, see "*Finding and Organizing Information*" on page 2-15.

Using ActiveSync on the desktop computer:
Use the Explorer in ActiveSync to explore your 700 Series Computer files and locate the program. Right-click the program, and then click **Create Shortcut**. Move the shortcut to the **Start Menu** folder in the Windows folder. The shortcut now appears on the **Start** menu. For more information, see ActiveSync Help.

Removing Programs

- Tap **Start** → **Settings** → the **System** tab → **Remove Programs**.

If the program does not appear in the list of installed programs, use File Explorer on your 700 Series Computer to locate the program, tap and hold the program, and then tap **Delete** on the pop-up menu.

Microsoft ActiveSync

Visit the following Microsoft Web site for the latest in updates, technical information, and samples:

http://www.microsoft.com/mobile/pocketpc/downloads/activesync.asp

Using Microsoft ActiveSync, you can synchronize the information on your desktop computer with the information on your 700 Series Computer. Synchronization compares the data on your 700 Series Computer with your desktop computer and updates both computers with the most recent information. For example:

- Keep Pocket Outlook data up-to-date by synchronizing your 700 Series Computer with Microsoft Outlook data on your desktop computer.
- Synchronize Microsoft Word and Microsoft Excel files between your 700 Series Computer and desktop computer. Your files are automatically converted to the correct format

▶ **NOTE:** *By default, ActiveSync does not automatically synchronize all types of information. Use ActiveSync options to turn synchronization on and off for specific information types.*

With ActiveSync, you can also:

- Back up and restore your 700 Series Computer data.
- Copy (rather than synchronize) files between your 700 Series Computer and desktop computer.
- Control when synchronization occurs by selecting a synchronization mode. For example, you can synchronize continually while connected to your desktop computer or only when you choose the synchronize command.
- Select which information types are synchronized and control how much data is synchronized. For example, you can choose how many weeks of past appointments you want synchronized.

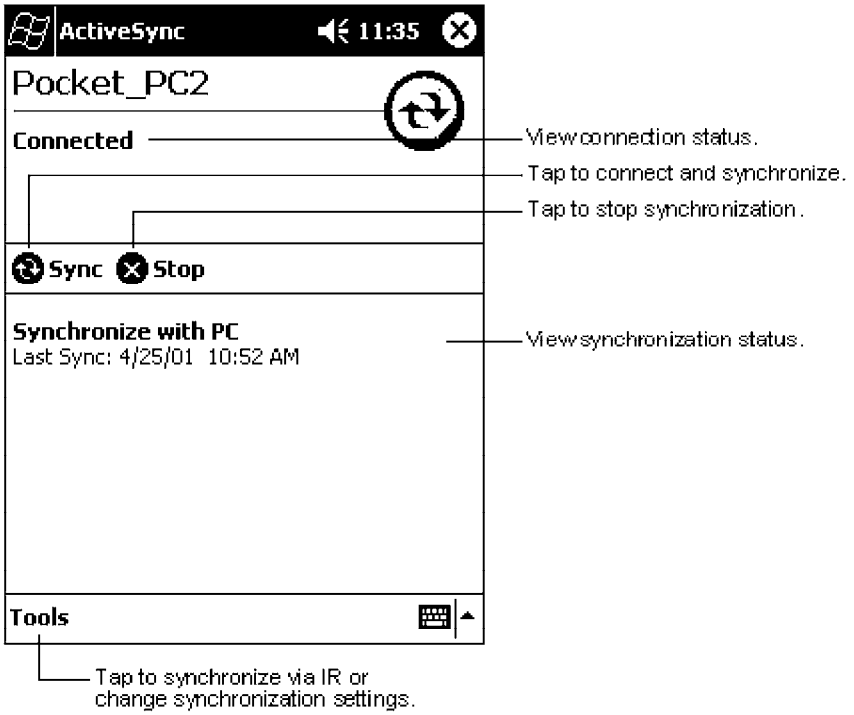
Before you begin synchronization, install ActiveSync on your desktop computer from the Pocket PC Companion CD. For more information on installing ActiveSync, see your Quick Start card. ActiveSync is already installed on your 700 Series Computer.

After installation is complete, the ActiveSync Setup Wizard helps you connect your 700 Series Computer to your desktop computer, set up a partnership so you can synchronize information between your 700 Series Computer and your desktop computer, and customize your synchronization settings. Your first synchronization process will automatically begin when you finish using the wizard.

After your first synchronization, take a look at Calendar, Contacts, and Tasks on your 700 Series Computer. You will notice that information you have stored in Microsoft Outlook on your desktop computer has been copied to your 700 Series Computer, and you did not have to type a word. Disconnect your 700 Series Computer from your computer and you are ready to go!

Once you have set up ActiveSync and completed the first synchronization process, you can initiate synchronization from your 700 Series Computer. To switch to ActiveSync on your 700 Series Computer, tap **Start → ActiveSync**. Note that if you have a wireless LAN card, you can synchronize remotely from your 700 Series Computer.

For information about using ActiveSync on your desktop computer, start ActiveSync on your desktop computer, and then see ActiveSync Help.



For more information about ActiveSync on your 700 Series Computer, switch to ActiveSync, then tap **Start** → **Help**.

Microsoft Pocket Outlook

► **NOTE:** *The Professional Edition of Microsoft Pocket Outlook does not include a spell checker.*

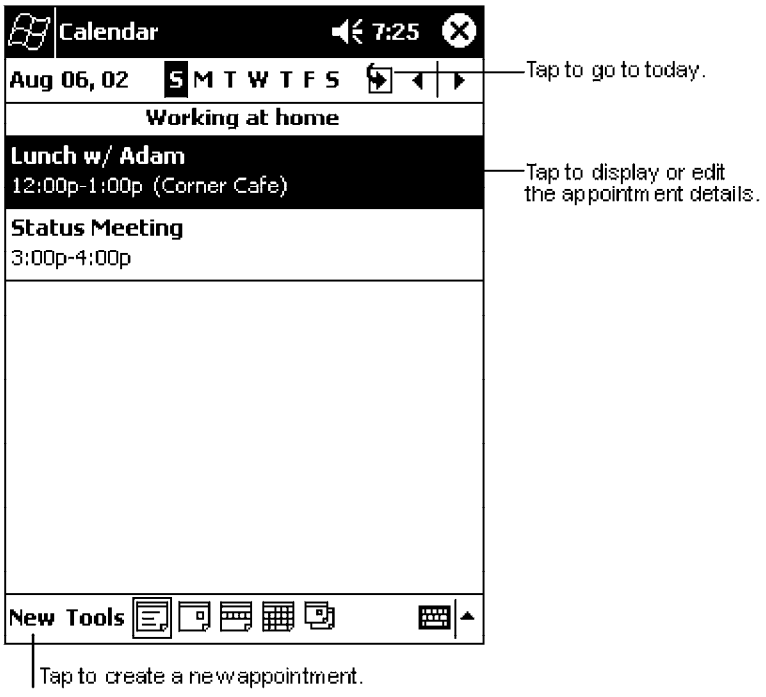
Microsoft Pocket Outlook includes Calendar, Contacts, Tasks, Inbox, and Notes. You can use these programs individually or together. For example, e-mail addresses stored in Contacts can be used to address e-mail messages in Inbox.

Using ActiveSync, you can synchronize information in Microsoft Outlook or Microsoft Exchange on your desktop computer with your 700 Series Computer. You can also synchronize this information directly with a Microsoft Exchange server. Each time you synchronize, ActiveSync compares the changes you made on your 700 Series Computer and desktop computer or server and updates both computers with the latest information. For information on using ActiveSync, see ActiveSync Help on the desktop computer.

You can switch to any of these programs by tapping them on the **Start** menu.

Calendar: Scheduling Appointments and Meetings

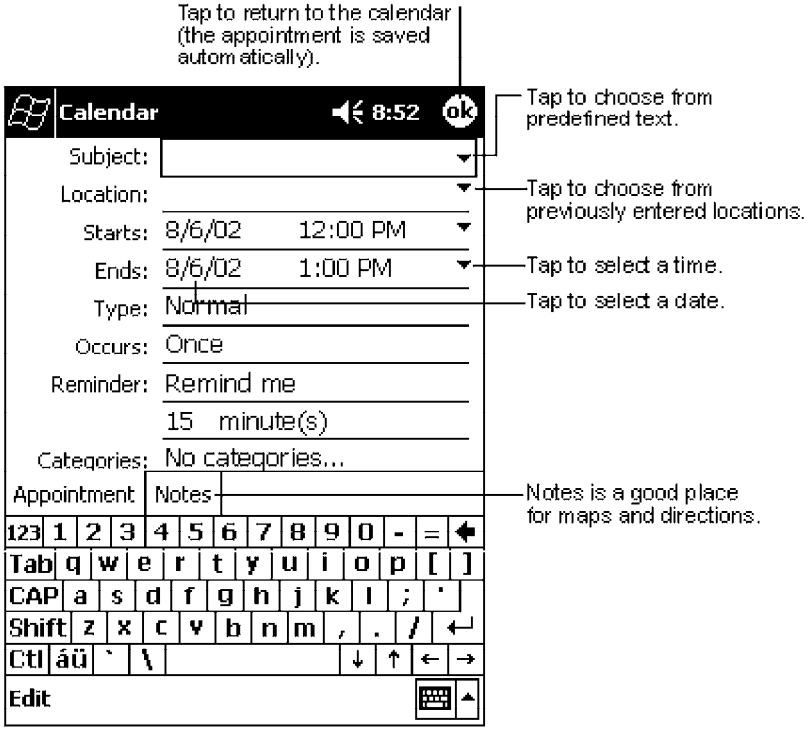
Use Calendar to schedule appointments, including meetings and other events. You can check your appointments in one of several views (Agenda, Day, Week, Month, and Year) and easily switch views by using the **View** menu.



► **NOTE:** *You can customize the Calendar display, such as changing the first day of the week, by tapping **Options** on the **Tools** menu.*

Creating an Appointment

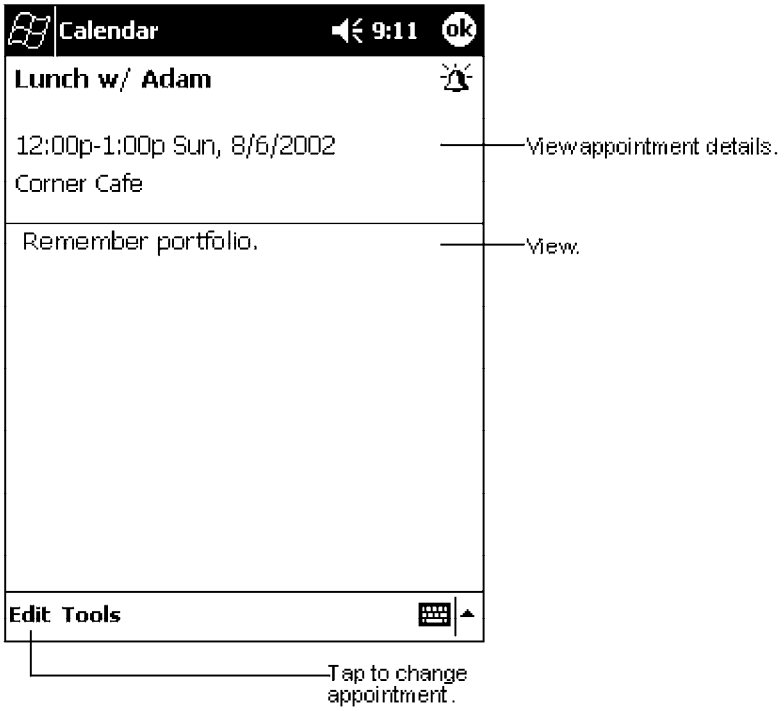
1. If you are in **Day** or **Week** view, tap the desired date and time for the appointment.
2. Tap **New**.
3. Using the input panel, enter a description and a location. Tap first to select the field.
4. If needed, tap the date and time to change them.
5. Enter other desired information. You will need to hide the input panel to see all available fields.
6. To add notes, tap the **Notes** tab. You can enter text, draw, or create a recording. For more information on creating notes, see "*Notes: Capturing Thoughts and Ideas*" on page 2-29.
7. When finished, tap **OK** to return to the calendar.



▶ **NOTE:** *If you select **Remind me** in an appointment, your 700 Series Computer will remind you according to the options set in **Start** → **Settings** → the **Personal** tab → **Sounds & Reminders**.*

Using the Summary Screen

When you tap an appointment in Calendar, a summary screen is displayed. To change the appointment, tap **Edit**.



Creating Meeting Requests

You can use Calendar to set up meetings with users of Outlook or Pocket Outlook. The meeting request will be created automatically and sent either when you synchronize Inbox or when you connect to your e-mail server. Indicate how you want meeting requests sent by tapping **Tools → Options**. If you send and receive e-mail messages through ActiveSync, select **ActiveSync**.

Scheduling a Meeting

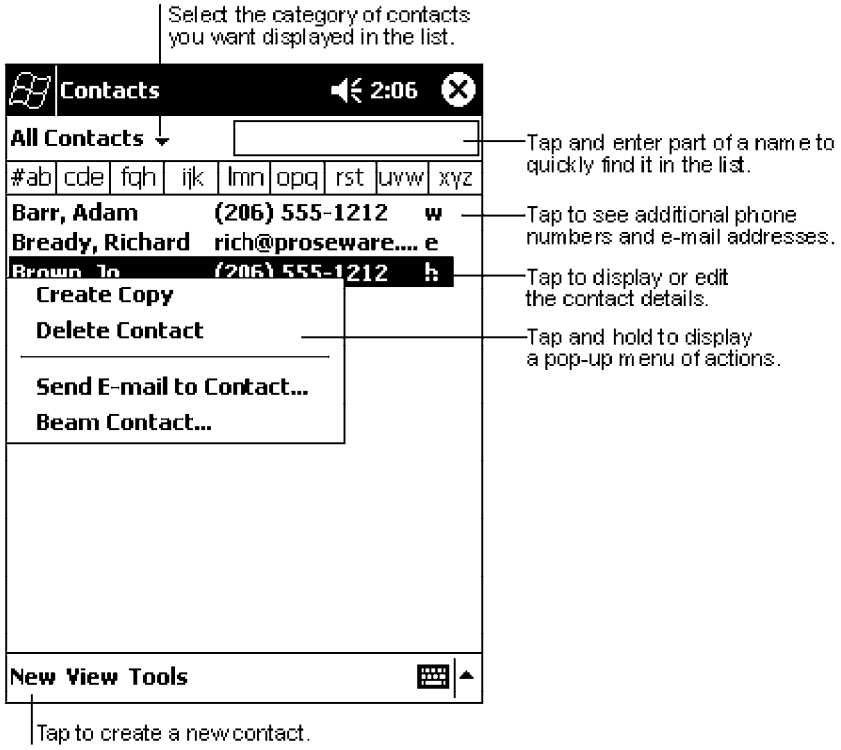
1. Create an appointment.
2. In the appointment details, hide the input panel, and then tap **Attendees**.
3. From the list of e-mail addresses you have entered in Contacts, select the meeting attendees.

The meeting notice is created automatically and placed in the Outbox folder.

For more information on sending and receiving meeting requests, see Calendar Help and Inbox Help on the 700 Series Computer.

Contacts: Tracking Friends and Colleagues

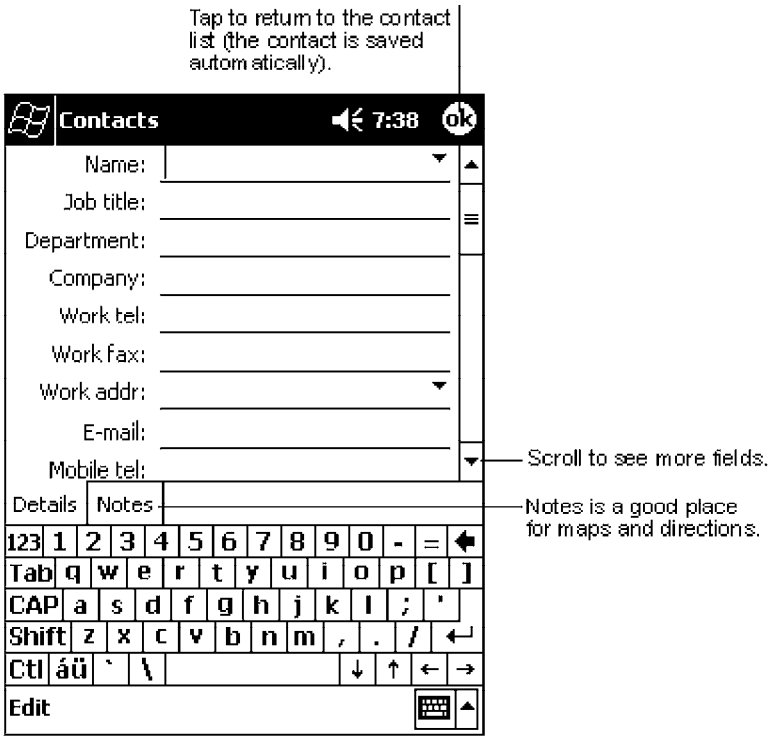
Contacts maintains a list of your friends and colleagues so that you can easily find the information you are looking for, whether you are at home or on the road. Using the 700 Series Computer infrared (IR) port, you can quickly share Contacts information with other 700 Series Computer users.



▶ **NOTE:** *To change the way information is displayed in the list, tap **Tools** → **Options**.*

Creating a Contact

1. Tap **New**.
2. Using the input panel, enter a name and other contact information. You will need to scroll down to see all available fields.
3. To assign the contact to a category, scroll to and tap **Categories** and select a category from the list. In the contact list, you can display contacts by category.
4. To add notes, tap the **Notes** tab. You can enter text, draw, or create a recording. For more information on creating notes, see "*Notes: Capturing Thoughts and Ideas*" on page 2-29.
5. When finished, tap **OK** to return to the contact list.



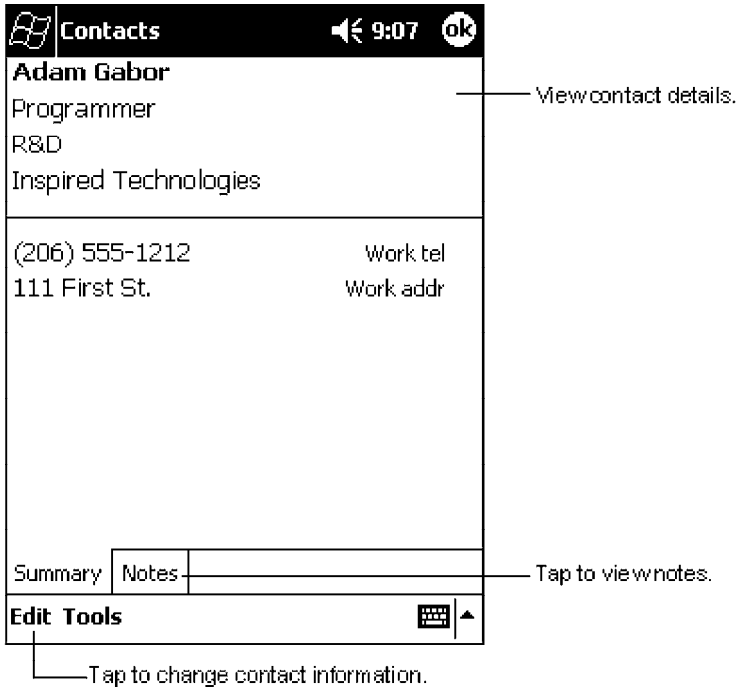
Finding a Contact

There are four ways to find a contact:

- In the contact list, enter a contact name in the box under the navigation bar. To show all contacts again, clear text from the box or tap the button to the right of the box.
- In the contact list, tap the category list (labeled **All Contacts** by default) and select the type of contact that you want displayed. To show all contacts again, select **All Contacts**. To view a contact not assigned to a category, select **None**.
- To view the names of companies your contacts work for, in the contact list, tap **View** → **By Company**. The number of contacts that work for that company will be displayed to the right of the company name.
- Tap **Start** → **Find**, enter the contact name, select **Contacts** for the type, and then tap **Go**.

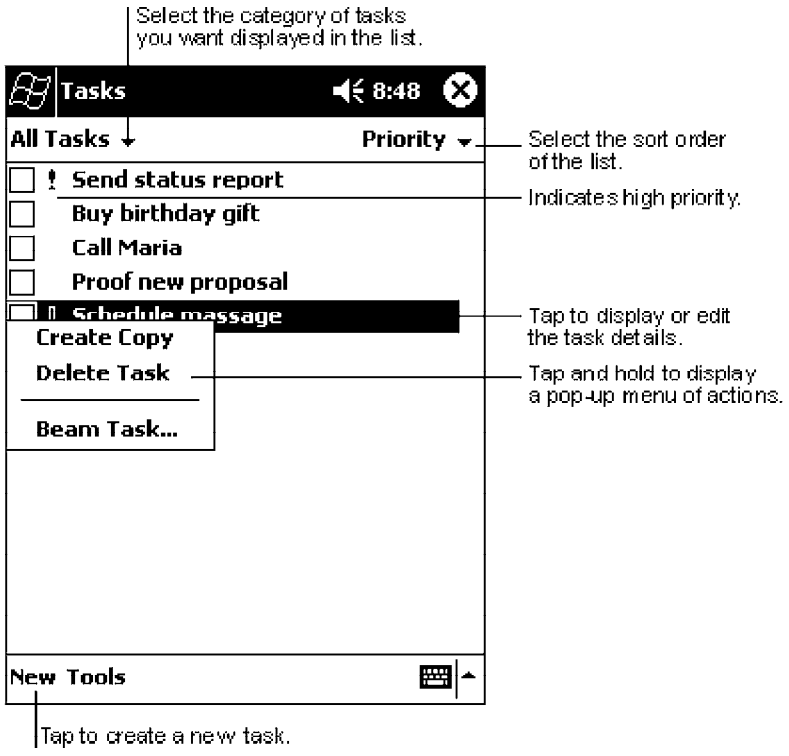
Using the Summary Screen

When you tap a contact in the contact list, a summary screen is displayed. To change the contact information, tap **Edit**.



Tasks: Keeping a To Do List

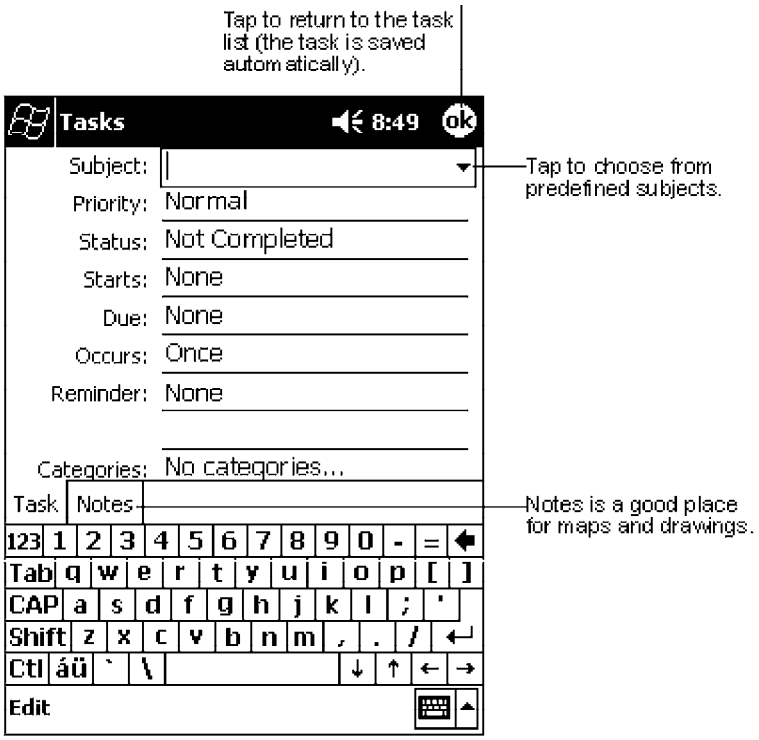
Use Tasks to keep track of what you have to do.



▶ **NOTE:** *To change the way information is displayed in the list, tap **Tools** → **Options**.*

Creating a Task

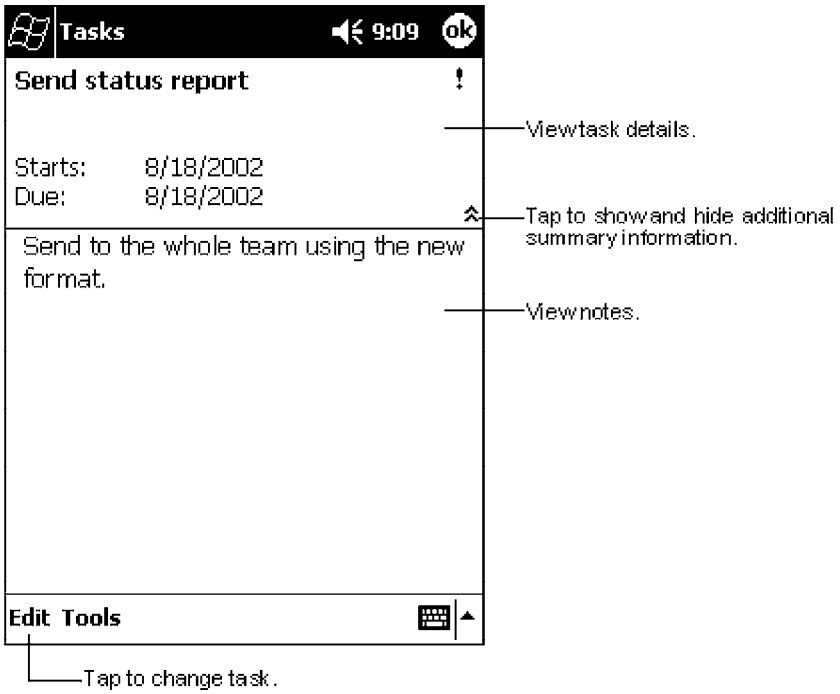
1. Tap **New**.
2. Using the input panel, enter a description.
3. You can enter a start date and due date or enter other information by first tapping the field. If the input panel is open, you will need to hide it to see all available fields.
4. To assign the task to a category, tap **Categories** and select a category from the list. In the task list, you can display tasks by category.
5. To add notes, tap the **Notes** tab. You can enter text, draw, or create a recording. For more information on creating notes, see "*Notes: Capturing Thoughts and Ideas*" on page 2-29.
6. When finished, tap **OK** to return to the task list.



▶ **NOTE:** *To quickly create a task with only a subject, tap **Entry Bar** on the **Tools** menu. Then, tap **Tap here to add a new task** and enter your task information.*

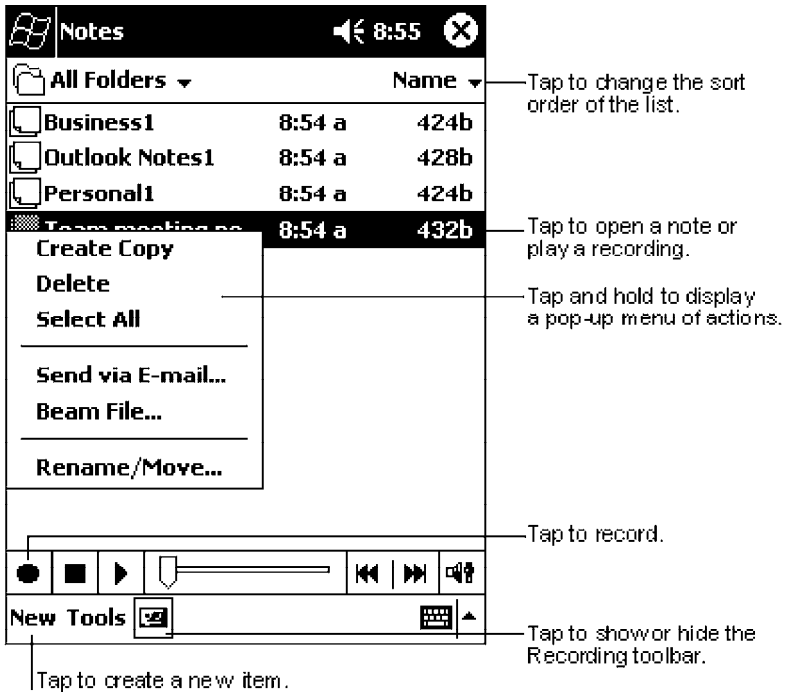
Using the Summary Screen

When you tap a task in the task list, a summary screen is displayed. To change the task, tap **Edit**.



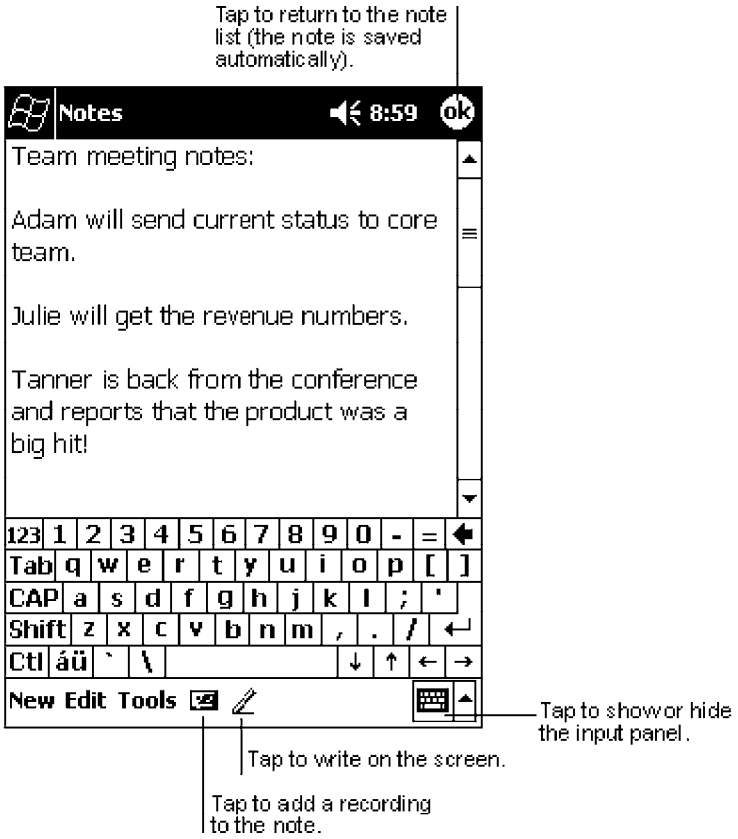
Notes: Capturing Thoughts and Ideas

Quickly capture thoughts, reminders, ideas, drawings, and phone numbers with Notes. You can create a written note or a recording. You can also include a recording in a note. If a note is open when you create the recording, it will be included in the note as an icon. If the note list is displayed, it will be created as a stand-alone recording.



Creating a Note

1. Tap **New**.
2. Create your note by writing, drawing, typing, and recording. For more information about using the input panel, writing and drawing on the screen, and creating recordings, see "*Basic Skills*" on page 2-4.



Inbox: Sending and Receiving E-mail Messages

Use Inbox to send and receive e-mail messages in either of these ways:

- Synchronize e-mail messages with Microsoft Exchange or Microsoft Outlook on your desktop computer.
- Send and receive e-mail messages by connecting directly to an e-mail server through an Internet Service Provider (ISP) or a network.

Synchronizing E-mail Messages

E-mail messages can be synchronized as part of the general synchronization process. You will need to enable Inbox synchronization in ActiveSync. For information on enabling Inbox synchronization, see ActiveSync Help on the desktop computer. During synchronization:

- Messages are copied from the mail folders of Exchange or Outlook on your desktop computer to the ActiveSync folder in Inbox on your 700 Series Computer. By default, you will receive messages from the past three days only, the first 100 lines of each message, and file attachments of less than 100 KB in size.
- E-mail messages in the Outbox folder on your 700 Series Computer are transferred to Exchange or Outlook, and then sent from those programs.
- E-mail messages in subfolders must be selected in ActiveSync on your desktop computer in order to be transferred.

Connecting Directly to an E-mail Server

In addition to synchronizing e-mail messages with your desktop computer, you can send and receive e-mail messages by connecting to an e-mail server using a modem or network card connected to your 700 Series Computer. You will need to set up a remote connection to a network or an ISP, and a connection to your e-mail server. For more information, see "*Getting Connected*" on page 2-50.

When you connect to the e-mail server, new messages are downloaded to the 700 Series Computer Inbox folder, messages in the 700 Series Computer Outbox folder are sent, and messages that have been deleted on the e-mail server are removed from the 700 Series Computer Inbox folder.

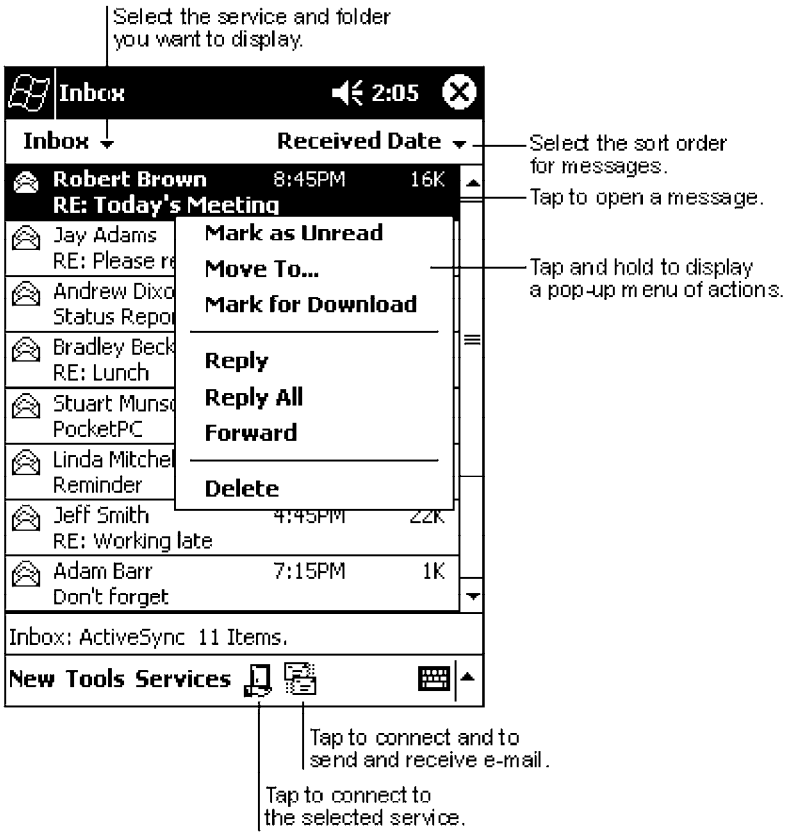
Messages that you receive directly from an e-mail server are linked to your e-mail server rather than your desktop computer. When you delete a message on your 700 Series Computer, it is also deleted from the e-mail server the next time you connect based on the settings you selected in ActiveSync.

You can work online or offline. When working online, you read and respond to messages while connected to the e-mail server. Messages are sent as soon as you tap **Send**, which saves space on your 700 Series Computer.

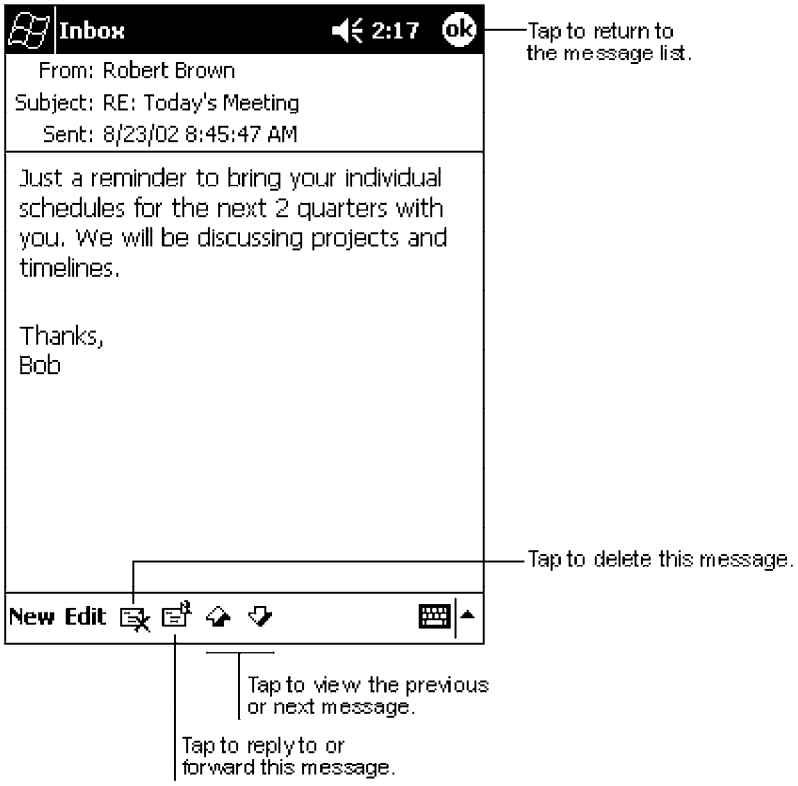
When working offline, once you have downloaded new message headers or partial messages, you can disconnect from the e-mail server and then decide which messages to download completely. The next time you connect, Inbox downloads the complete messages you have marked for retrieval and sends the messages you have composed.

Using the Message List

Messages you receive are displayed in the message list. By default, the most recently received messages are displayed first in the list.



When you receive a message, tap it to open it. Unread messages are displayed in bold.



When you connect to your e-mail server or synchronize with your desktop computer, by default, you will receive messages from the last five days only, the first 100 lines of each new message, and file attachments of less than 100 KB in size. The original messages remain on the e-mail server or your desktop computer.

You can mark the messages that you want to retrieve in full during your next synchronization or e-mail server connection. In the message list, tap and hold the message you want to retrieve. On the pop-up menu, tap **Mark for Download**. The icons in the Inbox message list give you visual indications of message status.

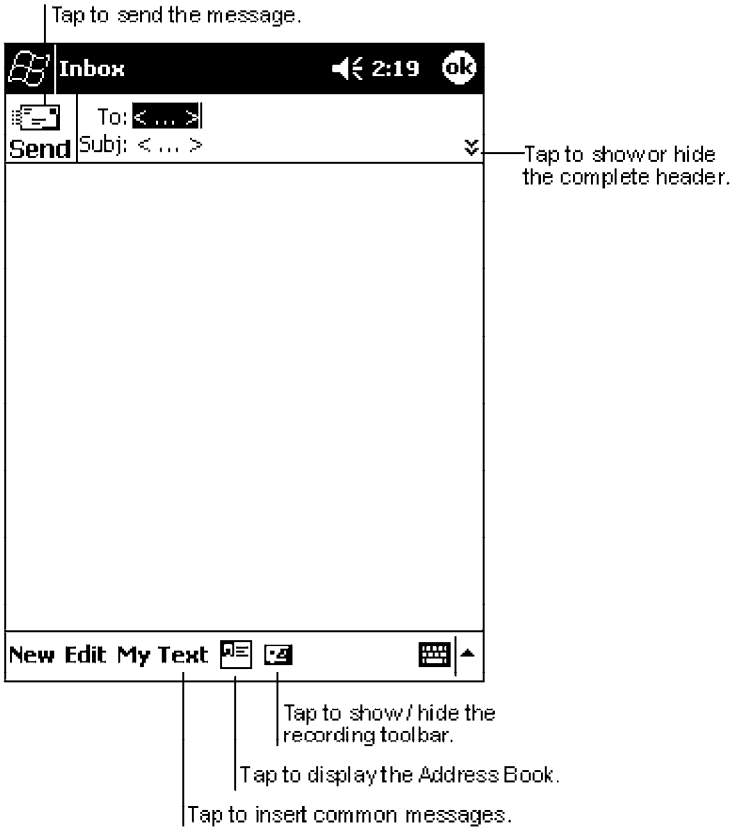
You specify your downloading preferences when you set up the service or select your synchronization options. You can change them at any time:

- Change options for Inbox synchronization using ActiveSync options. For more information, see ActiveSync Help.
- Change options for direct e-mail server connections in Inbox on your 700 Series Computer. Tap **Tools** → **Options** → the **Service** tab, then tap the service you want to change. Tap and hold the service and select **Delete** to remove a service.

Composing Messages

To compose a message:

1. Tap **New**.
2. In the **To** field, enter an e-mail or SMS address of one or more recipients, separating them with a semicolon, or select a name from the contact list by tapping the **Address Book** button. All e-mail addresses entered in the e-mail fields in Contacts appear in the Address Book.
3. Compose your message. To enter preset or frequently used messages, tap **My Text** and select a message.
4. Tap **Send** when you have finished the message. If you are working offline, the message is moved to the Outbox folder and will be sent the next time you connect.



If you are sending an SMS message and want to know if it was received, tap **Edit → Options → Request SMS text message delivery notification** before sending the message.

Managing E-mail Messages and Folders

By default, messages are displayed in one of five folders for each service you have created: Inbox, Deleted Items, Drafts, Outbox, and Sent Items. The Deleted Items folder contains messages that have been deleted on the 700 Series Computer. The behavior of the Deleted and Sent Items folders depends on the options you have chosen. In the message list, tap **Tools → Options →** the **Message** tab, then select your options.

If you want to organize messages into additional folders, tap **Tools → Manage Folders** to create new folders. To move a message to another folder, in the message list, tap and hold the message and then tap **Move to** on the pop-up menu.

Folder Behavior With a Direct Connection to an E-mail Server

The behavior of the folders you create depends on whether you are using ActiveSync, SMS, POP3, or IMAP4.

If you use ActiveSync:
E-mail messages in the Inbox folder in Outlook will automatically be synchronized with your 700 Series Computer. You can select to synchronize additional folders by designating them for ActiveSync. The folders you create and the messages you move will then be mirrored on the server. For example, if you move two messages from the Inbox folder to a folder named Family, and you have designated Family for synchronization, the server creates a copy of the Family folder and copies the messages into that folder. You can then read the messages while away from your desktop computer.

If you use SMS:
Messages are stored in the Inbox folder.

If you use POP3:
and you move e-mail messages to a folder you created, the link is broken between the messages on the 700 Series Computer and their copies on the mail server. The next time you connect, the mail server will see that the messages are missing from the 700 Series Computer Inbox and delete them from the server. This prevents you from having duplicate copies of a message, but it also means that you will no longer have access to messages that you move to folders created from anywhere except the 700 Series Computer.

If you use IMAP4:
The folders you create and the e-mail messages you move are mirrored on the server. Therefore, messages are available to you anytime you connect to your mail server, whether it is from your 700 Series Computer or desktop computer. This synchronization of folders occurs whenever you connect to your mail server, create new folders, or rename/delete folders when connected.

Companion Programs

The companion programs consist of Microsoft Pocket Word, Microsoft Pocket Excel, Windows Media Player for Pocket PC, and Microsoft Reader. To switch to a companion program on your 700 Series Computer, tap **Start → Programs**, then tap the program name.

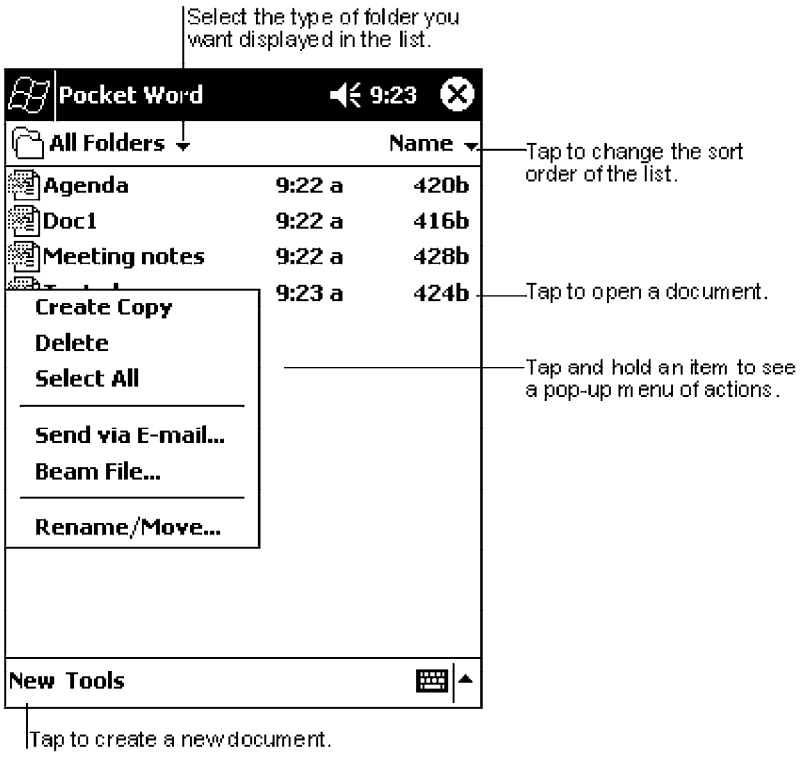
Pocket Word

Pocket Word works with Microsoft Word on your desktop computer to give you easy access to copies of your documents. You can create new documents on your 700 Series Computer, or you can copy documents from your desktop computer to your 700 Series Computer. Synchronize documents between your desktop computer and your 700 Series Computer so that you have the most up-to-date content in both locations.

Creating a Document

Use Pocket Word to create documents, such as letters, meeting minutes, and trip reports. To create a new file, tap **Start → Programs → Pocket Word → New**. A blank document appears. Or, if you have selected a template for new documents in the **Options** dialog box, that template appears with appropriate text and formatting already provided. You can open only one document at a time; when you open a second document, you will be asked to save the first. You can save a document you create or edit in a variety of formats, including Word (.DOC), Pocket Word (.PSW), Rich Text Format (.RTF), and Plain Text (.TXT).

Pocket Word contains a list of the files stored on your 700 Series Computer. Tap a file in the list to open it. To delete, make copies of, and send files, tap and hold a file in the list. Then, select the appropriate action on the pop-up menu.



You can enter information in Pocket Word in one of four modes (typing, writing, drawing, and recording), which are displayed on the **View** menu. Each mode has its own toolbar, which you can show and hide by tapping the **Show/Hide Toolbar** button on the command bar.

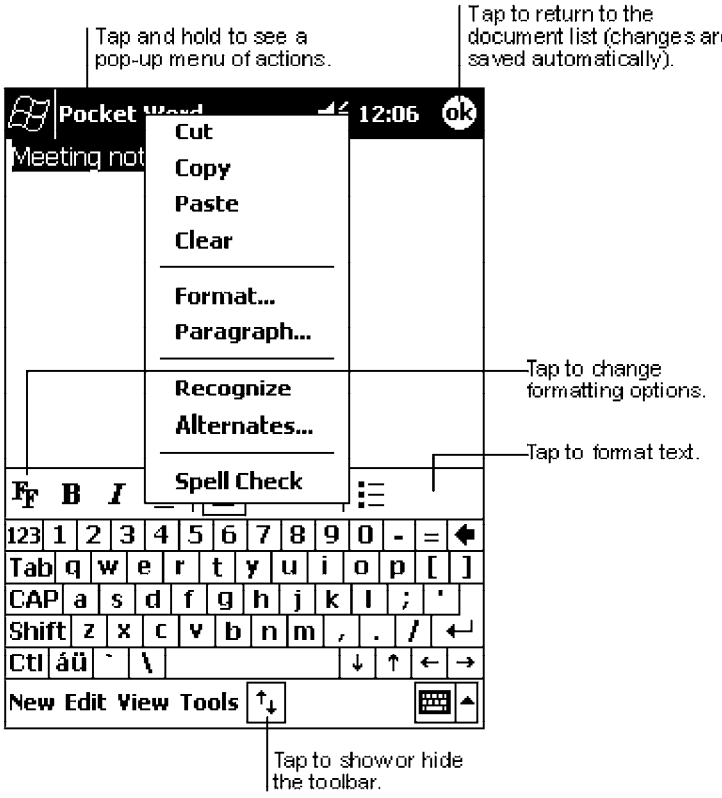
▶ **NOTE:** *You can change the zoom magnification by tapping **View** → **Zoom**, then select the percentage you want. Select a higher percentage to enter text and a lower one to see more of your document.*

*If you are opening a Word document created on a desktop computer, select **Wrap to Window** on the **View** menu so that you can see the entire document.*

Typing Mode

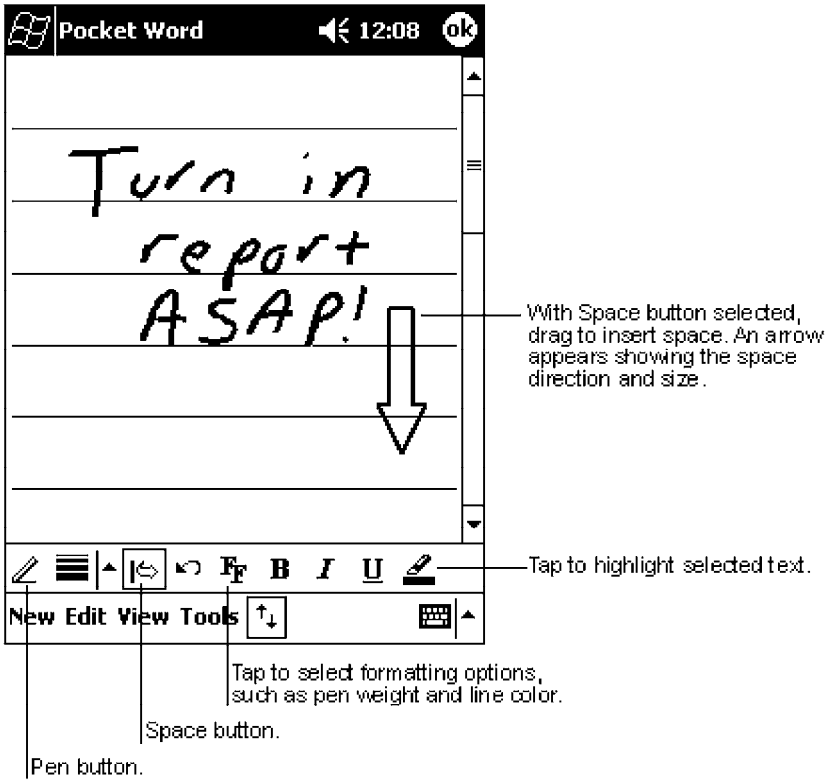
Using the input panel, enter typed text into the document. For more information on entering typed text, see "*Basic Skills*" on page 2-4.

To format existing text and to edit text, first select the text. You can select text as you do in a Word document, using your stylus instead of the mouse to drag through the text you want to select. You can search a document to find text by tapping **Edit** → **Find/Replace**.



Writing Mode

In writing mode, use your stylus to write directly on the screen. Ruled lines are displayed as a guide, and the zoom magnification is greater than in typing mode to allow you to write more easily. For more information on writing and selecting writing, see "*Basic Skills*" on page 2-4.

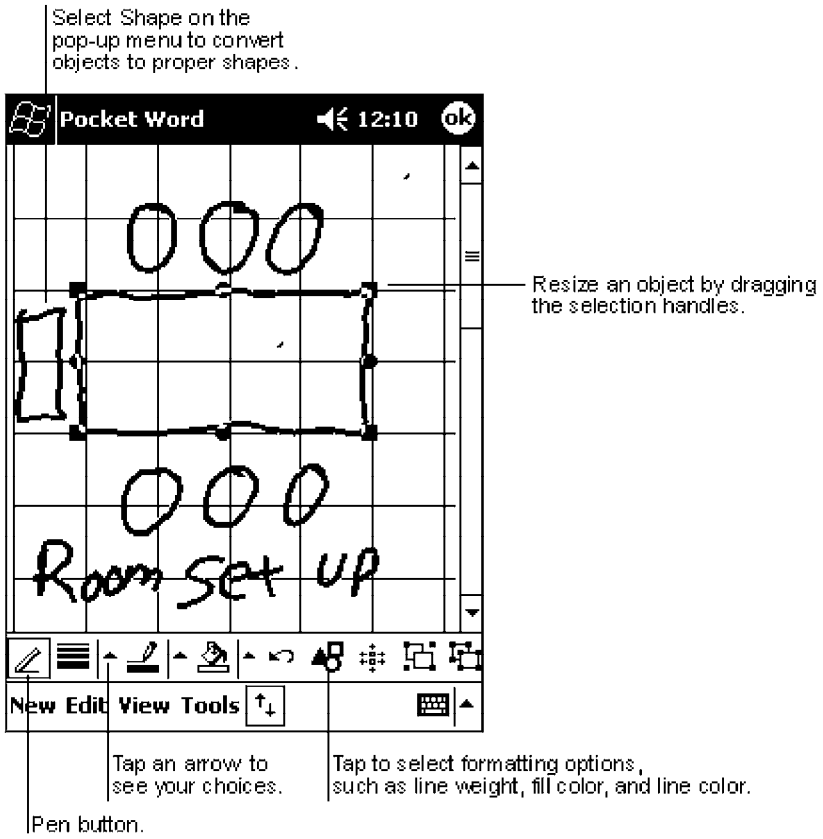


▶ **NOTE:** *If you cross three ruled lines in a single stylus stroke, the writing becomes a drawing, and can be edited and manipulated as described in "Drawing Mode" on the next page.*

Written words are converted to graphics (metafiles) when a Pocket Word document is converted to a Word document on your desktop computer.

Drawing Mode

In drawing mode, use your stylus to draw on the screen. Grid lines appear as a guide. When you lift your stylus off the screen after the first stroke, you will see a drawing box indicating the boundaries of the drawing. Every subsequent stroke within or touching the drawing box becomes part of the drawing. For more information on drawing and selecting drawings, see "*Basic Skills*" on page 2-4.



Recording Mode

In recording mode, embed a recording into your document. Recordings are saved as .WAV files. For more information on recording, see "*Basic Skills*" on page 2-4.

For more information on using Pocket Word, tap **Start → Help**.

Pocket Excel

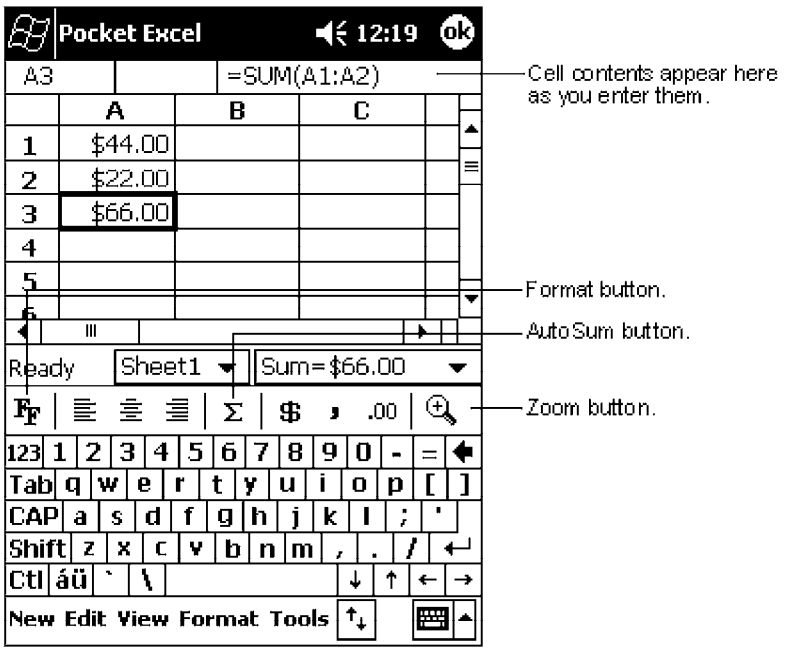
Pocket Excel works with Microsoft Excel on your desktop computer to give you easy access to copies of your workbooks. You can create new workbooks on your 700 Series Computer, or you can copy workbooks from your desktop computer to your 700 Series Computer. Synchronize workbooks between your desktop computer and your 700 Series Computer so that you have the most up-to-date content in both locations.

Creating a Workbook

Use Pocket Excel to create workbooks, such as expense reports and mileage logs. To create a new file, tap **Start → Programs → Pocket Excel → New**. A blank workbook appears. Or, if you have selected a template for new workbooks in the Options dialog box, that template appears with appropriate text and formatting already provided. You can open only one workbook at a time; when you open a second workbook, you will be asked to save the first. You can save a workbook you create or edit in a variety of formats, including Pocket Excel (.PXL) and Excel (.XLS).

Pocket Excel contains a list of the files stored on your 700 Series Computer. Tap a file in the list to open it. To delete, make copies of, and send files, tap and hold a file in the list. Then select the appropriate action from the pop-up menu.

Pocket Excel provides fundamental spreadsheet tools, such as formulas, functions, sorting, and filtering. To display the toolbar, tap **View → Toolbar**.



▶ **NOTE:** *If your workbook contains sensitive information, you can protect it with a password. To do so, open the workbook, tap **Edit → Password**. Every time you open the workbook, you will need to enter the password, so choose one that is easy for you to remember but hard for others to guess.*

Tips for Working in Pocket Excel

Note the following when working in large worksheets in Pocket Excel:

- View in full-screen mode to see as much of your worksheet as possible. Tap **View → Full Screen**. To exit full-screen mode, tap **Restore**.
- Show and hide window elements. Tap **View** and then tap the elements you want to show or hide.
- Freeze panes on a worksheet. First select the cell where you want to freeze panes. Tap **View → Freeze Panes**. You might want to freeze the top and leftmost panes in a worksheet to keep row and column labels visible as you scroll through a sheet.
- Split panes to view different areas of a large worksheet. Tap **View → Split**. Then drag the split bar to where you want it. To remove the split, tap **View → Remove Split**.
- Show and hide rows and columns. To hide a hidden row or column, select a cell in the row or column you want to hide. Then tap **Format** , **Row** or **Column → Hide**. To show a hidden row or column, tap **Tools → Go To**, and then type a reference that is in the hidden row or column. Then tap **Format → Row** or **Column → Unhide**.

For more information on using Pocket Excel, tap **Start → Help**.

MSN Messenger

▶ **NOTE:** *MSN Messenger is **only** available on the Premium Edition of Pocket PC 2002.*

MSN Messenger on your 700 Series Computer is an instant messaging program that lets you:

- See who is online.
- Send and receive instant messages.
- Have instant message conversations with groups of contacts.

To use MSN Messenger, you must have a Microsoft Passport account or a Microsoft Exchange e-mail account. You must have a Passport to use MSN Messenger Service. If you have a Hotmail or MSN account, you already have a Passport. Once you have obtained either a Microsoft Passport or a Microsoft Exchange account, you are ready to set up your account.

▶ **NOTE:** *Sign up for a Microsoft Passport account at: http://www.passport.com.*
Get a free Microsoft Hotmail e-mail address at: http://www.hotmail.com.

To switch to MSN Messenger, tap **Start → Programs → MSN Messenger**.

Setting Up

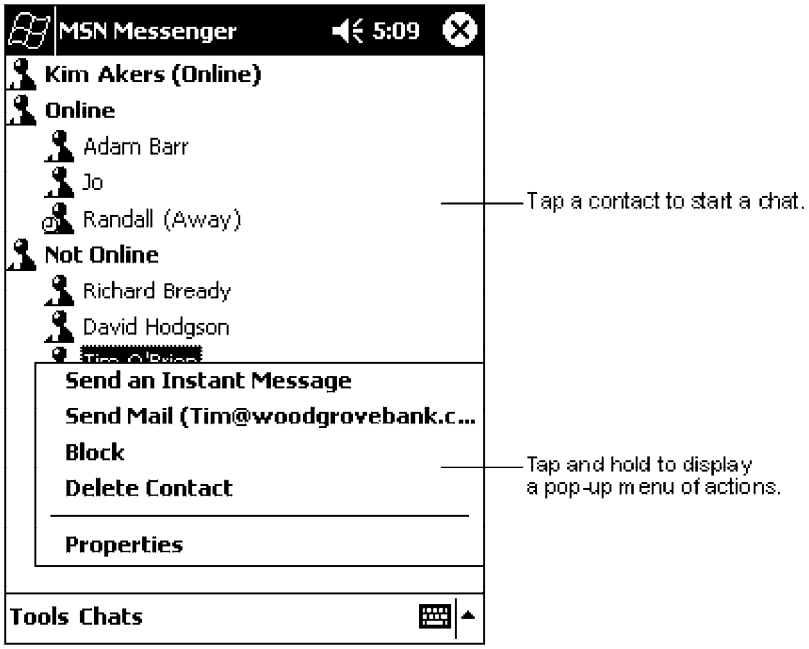
Before you can connect, you must enter Passport or Exchange account information. To set up an account and sign in:

1. In the **Tools** menu, tap **Options**.
2. In the **Accounts** tab, enter your Passport or Exchange account information.
3. To sign in, tap the sign-in screen and enter your e-mail address and password.

▶ **NOTE:** *If you already use MSN Messenger on your desktop computer, your contacts will show up on your 700 Series Computer without being added again.*

Working with Contacts

The MSN Messenger window shows all of your messenger contacts at a glance, divided into Online and Not Online categories. From this view, while connected, you can chat, send e-mail, block the contact from chatting with you, or delete contacts from your list using the pop-up menu.

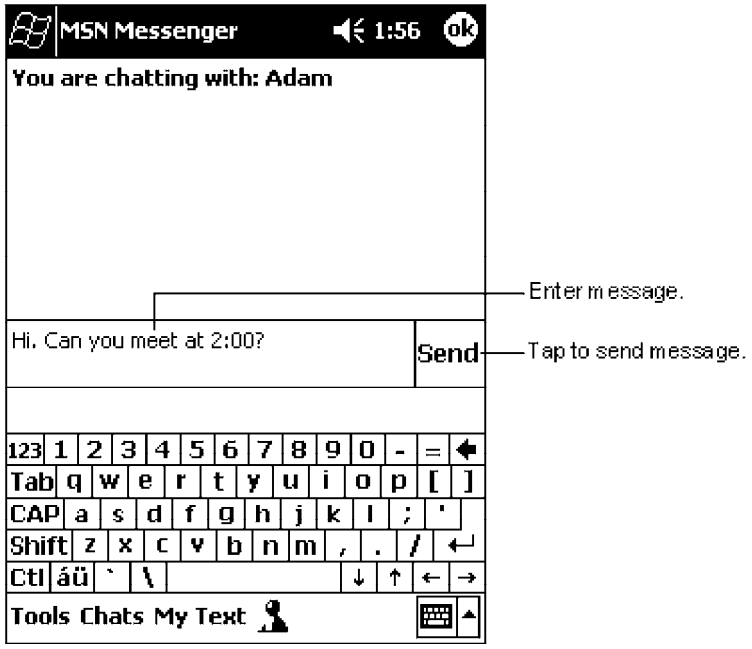


▶ **NOTE:** *To see others online without being seen, in the **Tools** menu, tap **My Status** → **Appear Offline**.*

*If you block a contact, you will appear offline but will remain on the blocked contact's list. To unblock a contact, tap and hold the contact, then tap **Unblock** on the pop-up menu.*

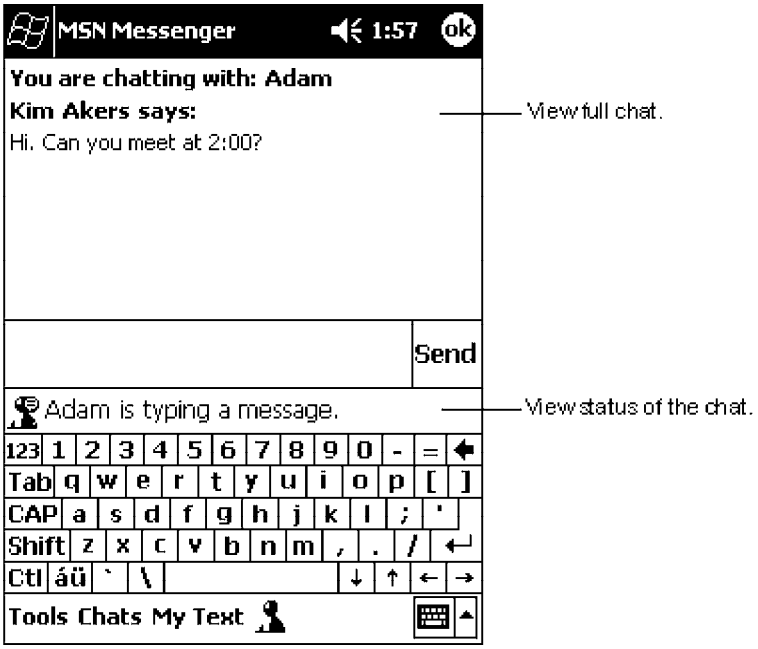
Chatting with Contacts

Tap a contact name to open a chat window. Enter your message in the text entry area at the bottom of the screen, or tap **My Text** to enter a preset message, and tap **Send**. To invite another contact to a multi-user chat, in the **Tools** menu, tap **Invite** and tap the contact you want to invite.



▶ **NOTE:** *To switch back to the main window without closing a chat, tap the **Contacts** button. To revert back to your chat window, tap **Chats** and select the person whom you were chatting with.*

To know if the contact you are chatting with is responding, look for the message under the text entry area.

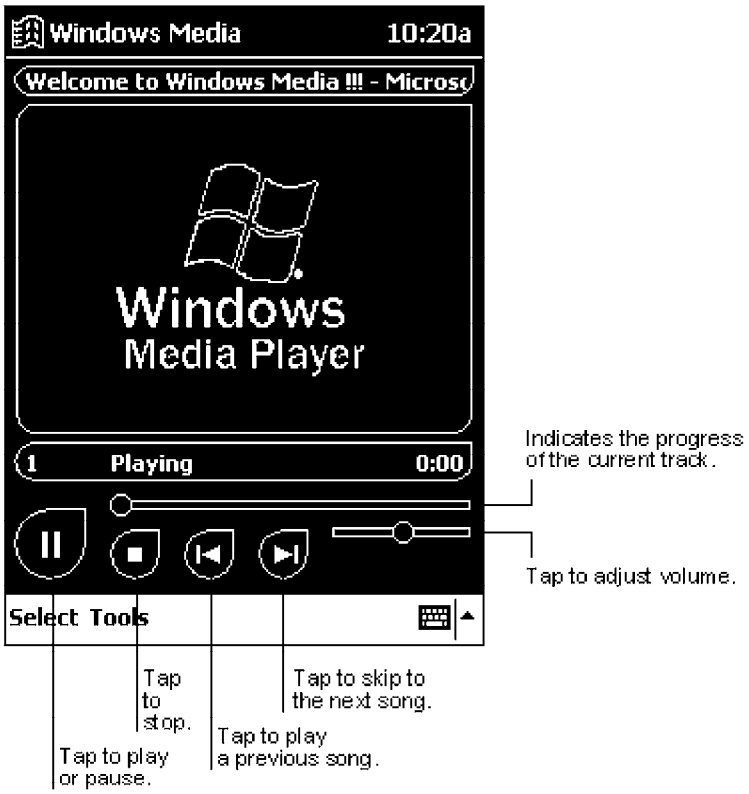


For more information on using MSN Messenger, tap **Start → Help**.

Windows Media Player for Pocket PC

Use Microsoft Windows Media Player for Pocket PC to play digital audio and video files that are stored on your 700 Series Computer or on a network. To switch to Windows Media Player for Pocket PC, tap **Start → Programs → Windows Media**.

Use Microsoft Windows Media Player on your desktop computer to copy digital audio and video files to your Pocket PC. You can play Windows Media and MP3 files on your Pocket PC.



For more information about using Windows Media Player for Pocket PC, tap **Start → Help**.

Microsoft Reader

Use Microsoft Reader to read eBooks on your 700 Series Computer. Download books to your desktop computer from your favorite eBook Web site. Then, use ActiveSync to copy the book files to your activated 700 Series Computer. The books appear in the Reader Library, where you can tap them in the list to open them. Each book consists of a cover page, an optional table of contents, and the pages of the book. You can:

- Page through the book by using the Up/Down control on your 700 Series Computer or by tapping the page number on each page.
- Annotate the book with highlighting, bookmarks, notes, and drawings.
- Search for text and look up definitions for words.

The Guidebook contains all the information you will need to use the software. To open the Guidebook, tap **Help** on the Reader command bar. Or, on a book page, tap and hold on the book title, and then tap **Help** on the pop-up menu. To switch to Microsoft Reader, tap **Start → Programs → Microsoft Reader**.

Getting Books on Your 700 Series Computer

You can download book files from the Web. Just visit your favorite eBook retailer and follow the instructions to download the book files.

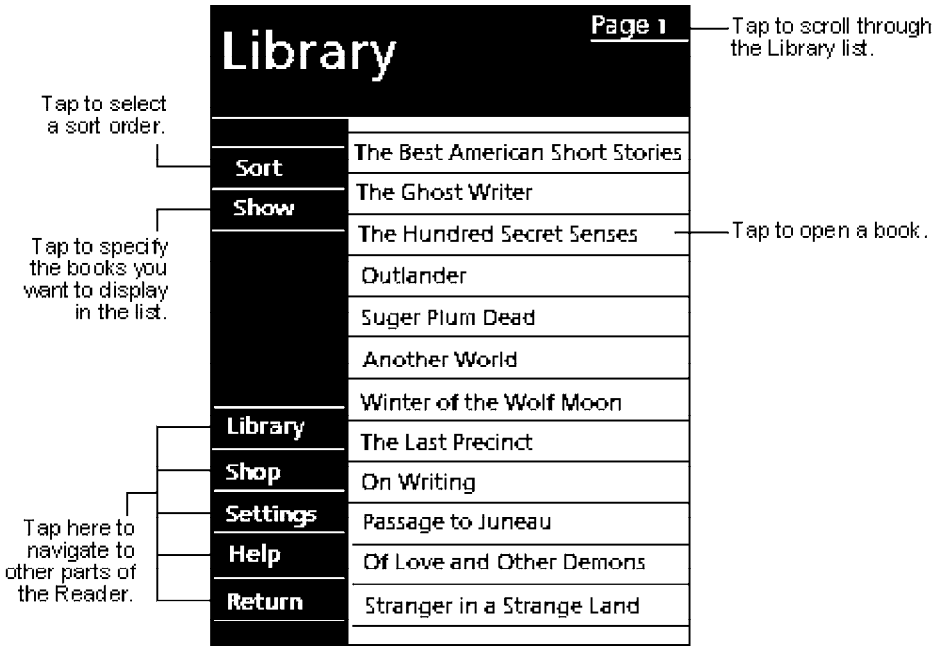
Sample books and a dictionary are also included in the MSReader folder in the Extras folder on the Pocket PC Companion CD.

Use ActiveSync to download the files from your desktop computer to your activated mobile computer as described in the Read Me file in the MSReader folder.

Using the Library

The Library is your Reader home page; it displays a list of all books stored on your 700 Series Computer or storage card. To open the Library:

1. On the Reader command bar, tap **Library**.
2. On a book page, tap the book title, then tap **Library** on the pop-up menu.
3. To open a book, tap its title in the Library list.

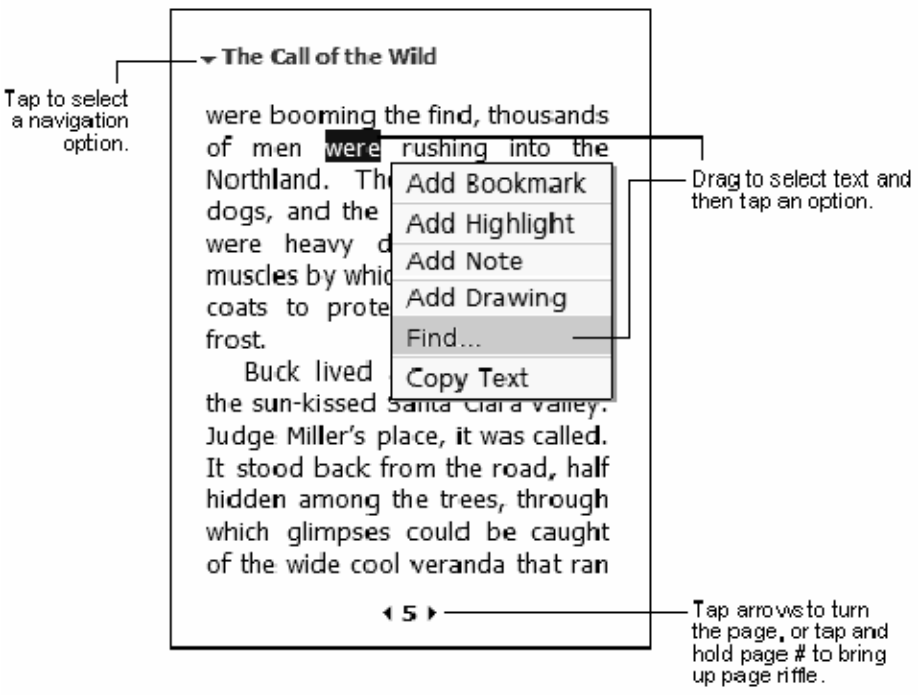


Reading a Book

Each book consists of a cover page, an optional table of contents, and the pages of the book. Navigation options are listed in the bottom portion of the cover page.

The first time you open a book, you will probably want to go to the first page or to the table of contents, if there is one. Subsequently, whenever you open the book, you will be automatically taken to the last page read.

In addition to the text, each book page includes a page number and book title.



You can also page through a book by using the Up/Down control on your 700 Series Computer.

Using Reader Features

Reading a book electronically gives you several options not available with paper books. These options are available from any book page.

Select text by dragging across the text on the page. Then, tap an option on the pop-up menu, as described here:

Searching for Text
Find text in a book by tapping **Find** on the pop-up menu. Enter the word you want to search for, and tap the desired **Find** option. Reader highlights found text on the page. To close **Find**, tap outside the box. To return to your original page, tap the title and then tap **Return** on the pop-up menu.

Copying Text
You can copy text from books that support this feature into any program that accepts text. On a book page, select the text you want to copy. Then, tap **Copy Text** on the pop-up menu. The text can be pasted into the program of your choice.

Adding Bookmarks
When you add a bookmark to a book, a color-coded bookmark icon appears in the right margin. You can add multiple bookmarks to a book. Then, from anywhere in the book, tap the bookmark icon to go to the bookmarked page.

Highlighting Text
When you highlight text, it appears with a colored background.

Attaching Notes to Text
When you attach a note to text, you enter the text in a notepad that appears on top of the book page. A **Note** icon will display in the left margin. To show or hide the note, tap the icon.

Adding Drawings
When you add a drawing, a **Drawing** icon appears in the bottom-left corner of the page, and drawing tools appear across the bottom of the page. Draw by dragging your stylus.

Annotations Index
To see a list of a book's annotations, including bookmarks, highlights, text notes, and drawings, tap **Annotations Index** on the book's cover page. You can tap an entry in the list to go to the annotated page.

Removing a Book

When you finish reading a book, you can delete it to conserve space on your 700 Series Computer. If a copy of the book is stored on your desktop computer, you can download it again at any time.

To remove a book from your 700 Series Computer, tap and hold the title in the Library list, and then tap **Delete** on the pop-up menu.

Pocket Internet Explorer

▶ **NOTE:** *The Professional Edition of Pocket Internet Explorer does not support WAP pages.*

Use Microsoft Pocket Internet Explorer to view Web or WAP pages in either of these ways:

- During synchronization with your desktop computer, download your favorite links and mobile favorites that are stored in the Mobile Favorites subfolder in Internet Explorer on the desktop computer.
- Connect to an ISP or network and browse the Web. To do this, you will need to create the connection first, as described in "*Getting Connected*" on page 2-50.

When connected to an ISP or network, you can also download files and programs from the Internet or intranet.

To switch to Pocket Internet Explorer, tap **Start → Internet Explorer**.

The Mobile Favorites Folder

Only items stored in the Mobile Favorites subfolder in the Favorites folder in Internet Explorer on your desktop computer will be synchronized with your 700 Series Computer. This folder was created automatically when you installed ActiveSync.

Favorite Links

During synchronization, the list of favorite links in the Mobile Favorites folder on your desktop computer is synchronized with Pocket Internet Explorer on your 700 Series Computer. Both computers are updated with changes made to either list each time you synchronize. Unless you mark the favorite link as a mobile favorite, only the link will be downloaded to your 700 Series Computer, and you will need to connect to your ISP or network to view the content. For more information on synchronization, see ActiveSync Help on the desktop computer.

Mobile Favorites

If you are using Microsoft Internet Explorer 5.0 or later on your desktop computer, you can download mobile favorites to your 700 Series Computer. Synchronizing mobile favorites downloads Web content to your 700 Series Computer so that you can view pages while you are disconnected from your ISP and desktop computer.

Use the Internet Explorer plug-in installed with ActiveSync to create mobile favorites quickly. To create a mobile favorite:

1. In Internet Explorer on your desktop computer, click **Tools → Create Mobile Favorite**.
2. To change the link name, enter a new name in the **Name** box.
3. Optionally, in **Update**, select a desired update schedule.
4. Click **OK**. Internet Explorer downloads the latest version of the page to your desktop computer.
5. If you want to download the pages that are linked to the mobile favorite you just created, in Internet Explorer on the desktop computer, right-click the mobile favorite just created and then click **Properties**. In the **Download** tab, specify the number of links deep you want to download. To conserve 700 Series Computer memory, go only one level deep.
6. Synchronize your 700 Series Computer and desktop computer. Mobile favorites that are stored in the Mobile Favorites folder in Internet Explorer are downloaded to your 700 Series Computer.

► **NOTE:** *If you did not specify an update schedule in step 3 above, you will need to manually download content to keep the information updated on your desktop computer and 700 Series Computer. Before synchronizing with your 700 Series Computer, in Internet Explorer on your desktop computer, click **Tools → Synchronize**. You will see the last time content was downloaded to the desktop computer, and you can manually download content if needed.*

*You can add a button to the Internet Explorer toolbar for creating mobile favorites. In Internet Explorer on your desktop computer, click **View → Toolbars → Customize**.*

Mobile favorites take up storage memory on your 700 Series Computer. To minimize the amount of memory used:

- In the settings for the Favorites information, type in ActiveSync options, turn off pictures and sounds, or stop some mobile favorites from being downloaded to the 700 Series Computer. For more information, see ActiveSync Help.
- Limit the number of downloaded linked pages. In Internet Explorer on the desktop computer, right-click the mobile favorite you want to change and then **Properties**. In the **Download** tab, specify “0” or “1” for the number of linked pages you want to download.

Using AvantGo Channels

AvantGo is a free interactive service that gives you access to personalized content and thousands of popular Web sites. You subscribe to AvantGo channels directly from your 700 Series Computer. Then, you synchronize your 700 Series Computer and desktop computer, or connect to the Internet to download the content. For more information, visit the AvantGo Web site. To sign up for AvantGo:

1. In ActiveSync options on the desktop computer, turn on synchronization for the AvantGo information type.
2. In Pocket Internet Explorer on your 700 Series Computer, tap the **Favorites** button to display your list of favorites.
3. Tap the **AvantGo Channels** link.
4. Tap the **Activate** button.
5. Follow the directions on the screen. You will need to synchronize your 700 Series Computer with your desktop computer and then tap the **My Channels** button to complete the AvantGo setup.

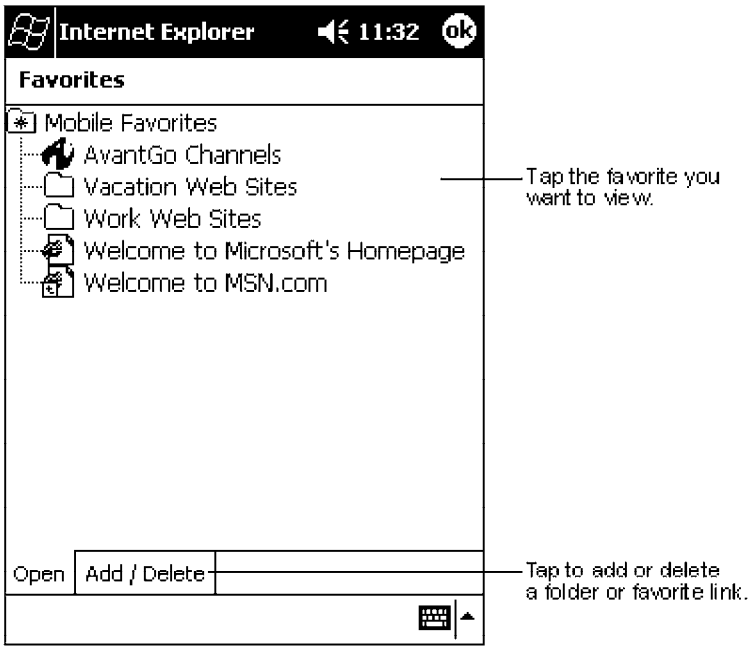
When synchronization is complete, tap the **AvantGo Channels** link in your list of favorites to see a few of the most popular channels. To add or remove channels, tap the **Add** or **Remove** link.

Using Pocket Internet Explorer

You can use Pocket Internet Explorer to browse mobile favorites and channels that have been downloaded to your 700 Series Computer without connecting to the Internet. You can also connect to the Internet through an ISP or a network connection and browse the Web.

Viewing Mobile Favorites and Channels

1. Tap the **Favorites** button to display your list of favorites.
2. Tap the page you want to view.



You will see the page that was downloaded the last time you synchronized with your desktop computer. If the page is not on your 700 Series Computer, the favorite will be dimmed. You will need to synchronize with your desktop computer again to download the page to your 700 Series Computer, or connect to the Internet to view the page.

Browsing the Internet

1. Set up a connection to your ISP or corporate network using **Connections**, as described in "*Getting Connected*" on page 2-50.
2. To connect and start browsing, do one of the following:
   - Tap the **Favorites** button, and then tap the favorite you want to view.
   - Tap **View → Address Bar**. In the address bar that appears at the top of the screen, enter the Web address you want to visit and then tap **Go**. Tap the arrow to choose from previously entered addresses.

▶ NOTE: *To add a favorite link while using the 700 Series Computer, go to the page you want to add, tap and hold on the page, and tap **Add to Favorites**.*

Getting Connected

You can use your 700 Series Computer to exchange information with other 700 Series Computers as well as your desktop computer, a network, or the Internet. You have the following connection options:

- Use the infrared (IR) port on your 700 Series Computer to send and receive files between two 700 Series Computers. If this is the method you want to use, see "*Transferring Items Using Infrared*" below.
- Connect to your ISP. Once connected, you can send and receive e-mail messages by using Inbox and view Web or WAP pages by using Pocket Internet Explorer. The communication software for creating an ISP connection is already installed on your 700 Series Computer. Your service provider will provide software needed to install other services, such as paging and fax services. If this is the method you want to use, see "*Connecting to an Internet Service Provider*" on page 2-51.
- Connect to the network at your company or organization where you work. Once connected, you can send and receive e-mail messages by using Inbox, view Web or WAP pages by using Pocket Internet Explorer, and synchronize with your desktop computer. If this is the method you want to use, see "*Connecting to Work*" on page 2-52.
- Connect to your desktop computer to synchronize remotely. Once connected, you can synchronize information such as your Pocket Outlook information. If this is the method you want to use, see ActiveSync Help on your desktop computer or Connections Help on the 700 Series Computer.

Transferring Items Using Infrared

Using infrared (IR), you can send and receive information, such as contacts and appointments, between two 700 Series Computers.

Sending Information

1. Switch to the program where you created the item you want to send and locate the item in the list.
2. Align the IR ports so that they are unobstructed and within a close range.
3. Tap and hold the item, and tap **Beam Item** on the pop-up menu.

▶ **NOTE:** *You can also send items, but not folders, from File Explorer. Tap and hold the item you want to send, and then tap* ***Beam File*** *on the pop-up menu.*

Receiving Information

1. Align the IR ports so that they are unobstructed and within a close range.
2. Have the owner of the other 700 Series Computer send the information to you. Your 700 Series Computer will automatically receive it.

Connecting to an Internet Service Provider

You can connect to your ISP, and use the connection to send and receive e-mail messages and view Web or WAP pages. You can connect to your ISP in one of two ways:

- Create a modem connection. If this is the method you want to use, see "*Creating a Modem Connection to an ISP*" below.
- Use an Ethernet card and a net tap to connect to the network. If this is the method you want to use, see "*Creating an Ethernet Connection to an ISP*" on page 2-52.

Creating a Modem Connection to an ISP

1. Obtain the following information from your ISP. Some ISPs require information in front of the user name, such as MSN/username.
   - ISP dial-up access telephone number
   - User name
   - Password
   - TCP/IP settings
2. If your 700 Series Computer does not have a built-in modem, install a modem card, or use or use a NULL modem cable and appropriate adapters to connect an external modem to your 700 Series Computer through the serial port.
3. Tap **Start** → **Settings** → the **Connections** tab → **Connections**. Under The Internet settings, select **Internet Settings** → **Modify**.
4. In the **Modem** tab, tap **New**.
5. Enter a name for the connection, such as "ISP Connection."
6. In **Select a modem list**, select your modem type. If your modem type does not appear, try reinserting the modem card. If you are using an external modem that is connected to your 700 Series Computer with a cable, select "Hayes Compatible on COM1."
7. You should not need to change any settings in **Advanced**. Most ISPs now use a dynamically-assigned address. If the ISP you are connecting to does not use a dynamically-assigned address, tap **Advanced** → the **TCP/IP** tab, then enter the address. When finished, tap **OK** → **Next**.
8. Enter the access phone number, and tap **Next**.
9. Select other desired options, and tap **Finish**.
10. In the **Dialing Locations** tab, specify your current location and phone type (most phone lines are tone). These settings will apply to all connections you create.

To start the connection, simply start using one of the following programs. Your 700 Series Computer will automatically begin connecting. Once connected, you can:

- Send and receive e-mail messages by using Inbox. Before you can use Inbox, you need to provide the information it needs to communicate with the e-mail server. For specific instructions, see "*Connecting Directly to an E-mail Server*" on page 2-54.
- Visit Web and WAP pages by using Pocket Internet Explorer. For more information, see "*Pocket Internet Explorer*" on page 2-46.
- Send and receive instant messages with MSN Messenger. For more information, see "*MSN Messenger*" on page 2-40.

Creating an Ethernet Connection to an ISP

1. You do not need to create a new connection on your 700 Series Computer. Instead, you must purchase and configure an Ethernet card that is compatible with your 700 Series Computer.
2. Obtain the following information from your ISP:
   - User name
   - Password
   - Domain name
3. Insert the Ethernet card into your 700 Series Computer. For instructions on inserting and using the Ethernet card, see the owner's manual for the card.
4. The first time you insert the card, **Network Settings** will appear automatically so that you can configure the Ethernet card. Most networks use DHCP, so you should not have to change these settings unless your network administrator instructs you to do so. Tap **OK**. (If it does not appear or to change settings later, tap **Start** → **Settings** → the **Connections** tab → **Network**, tap the adapter you want to change, and then tap **Properties**.)
5. Connect the Ethernet card to the network by using a network cable. For information, see your owner's manual.
6. Tap **Start** → **Settings** → the **Connections** tab → **Connections**. From the **My network card connects to** list, select "Internet."

To start the connection, simply start using one of the programs listed in the preceding section. Once connected, you can perform the same activities as listed in the preceding section.

Connecting to Work

If you have access to a network at work, you can send e-mail messages, view intranet pages, synchronize your 700 Series Computer, and possibly access the Internet. You can connect to work in one of two ways:

- Create a modem connection by using a RAS (Remote Access Server) account. Before you can create this modem connection, your network administrator will need to set up a RAS account for you. If this is the method you want to use, see "*Creating a Modem Connection to Work*" below. Your network administrator may also give you VPN settings.
- Use an Ethernet card and a net tap to connect to the network. If this is the method you want to use, see "*Creating an Ethernet Connection to Work*" on page 2-53.

Creating a Modem Connection to Work

1. Get the following information from your network administrator:
   - Dial-up access telephone number
   - User name
   - Password
   - Domain name
   - TCP/IP settings
2. If your 700 Series Computer does not have a built-in modem, install a modem card.
3. Tap **Start** → **Settings** → the **Connections** tab → **Connections**. Under The Internet settings, select **Internet Settings** and tap **Modify**.

4. In the **Modem** tab, tap **New**.
5. Enter a name for the connection, such as "Company Connection."
6. In the **Select a modem list**, select your modem type. If your modem type does not appear, try reinserting the modem card. If you are using an external modem that is connected to your 700 Series Computer with a cable, select "Hayes Compatible on COM1."
7. You should not need to change any settings in **Advanced**. Most servers now use a dynamically-assigned address. If the server you are connecting to does not use a dynamically-assigned address, tap **Advanced** → the **TCP/IP** tab and then enter the address. When finished, tap **OK** → **Next**.
8. Enter the access phone number, and tap **Next**.
9. Select other desired options, and tap **Finish**.
10. In the **Dialing Locations** tab, specify your current location and phone type (most phone lines are tone). These settings will apply to all connections you create.

To start the connection, simply start using one of the following programs. Your 700 Series Computer will automatically begin connecting. Once connected, you can:

- Send and receive e-mail messages by using Inbox. Before you can use Inbox, you need to provide the information it needs to communicate with the e-mail server. For specific instructions, see "*Connecting Directly to an E-mail Server*" on page 2-54.
- Visit Internet or intranet Web or WAP pages by using Pocket Internet Explorer.
- Send and receive instant messages with MSN Messenger. For more information, see "*MSN Messenger*" on page 2-40.
- Synchronize. For more information, see ActiveSync Help on the desktop computer.

Creating an Ethernet Connection to Work

1. You do not need to create a new connection on your 700 Series Computer. Instead, you must purchase and configure an Ethernet card that is compatible with your 700 Series Computer.
2. Get the following information from your network administrator:
   - User name
   - Password
   - Domain name
3. Insert the Ethernet card into your 700 Series Computer. For instructions on inserting and using the Ethernet card, see the owner's manual for the card.
4. The first time you insert the card, **Network Settings** will appear automatically so that you can configure the Ethernet card. Most networks use DHCP, so you should not have to change these settings unless your network administrator instructs you to do so. Tap **OK**. (If it does not appear or to change settings later, tap **Start** → **Settings** → the **Connections** tab → **Network**, tap the adapter you want to change, and then tap **Properties**.)
5. Connect the Ethernet card to the network by using a network cable. For information, see your owner's manual.
6. Tap **Start** → **Settings** → the **Connections** tab → **Connections**. From the **My network card connects to** list, select "Work."

7. If you want to synchronize your 700 Series Computer, tap **Start → ActiveSync**. In the **Tools** menu, tap **Options**. In the **PC** tab, select **Include PC when synchronizing remotely and connect to**, and select your computer's name. Remote synchronization with a desktop computer will work only if you have set up a partnership with that computer through ActiveSync and have set ActiveSync to allow remote connections. Other restrictions apply. For more information on synchronizing remotely, see ActiveSync Help on the desktop computer.

To start the connection, simply start using one of the programs listed in the preceding section. Once connected, you can perform the same activities as listed in the preceding section.

Ending a Connection

To disconnect, do one of the following:

- When connected via dial-up or VPN, tap the **Connection** icon *(shown left)* on your navigation bar, and then tap End.
- When connected via cable or cradle, detach your 700 Series Computer from the cable or cradle.
- When connected via Infrared, move the 700 Series Computer away from the PC.
- When connected via a network (Ethernet) card, remove the card from your 700 Series Computer.

Connecting Directly to an E-mail Server

You can set up a connection to an e-mail server so that you can send and receive e-mail messages by using a modem or network connection and Inbox on your 700 Series Computer.

▶ **NOTE:** *The ISP or network must use a POP3 or IMAP4 e-mail server and an SMTP gateway.*

You can use multiple e-mail services to receive your messages. For each e-mail service you intend to use, first set up and name the e-mail service. If you use the same service to connect to different mailboxes, set up and name each mailbox connection.

Setting Up an E-mail Service

- In Inbox on your 700 Series Computer, tap **Services → New Service**. Follow the directions in the New Service wizard.

For an explanation of a screen, tap **Start → Help**. When finished, to connect to your e-mail server, tap **Services → Connect**. For more information on using the Inbox program, see "*Inbox: Sending and Receiving E-mail Messages*" on page 2-30.

Section 3

Installing Applications and Updating System Software

There are multiple ways to get an application to your 700 Series Color Mobile Computer; just as there are multiple ways to package the application for delivery. Use any of the following methods to package an application for installation:

- For very simple applications, the application itself might be the only file that needs to be delivered.
- It could be a directory structure that contains the application, supporting files like ActiveX controls, DLLs, images, sound files, and data files.
- Via a CAB file.

Consider either of the following when choosing a location into which to store your application:

- In the basic 700 Series Computer, there are no built-in storage options other than the Object Store. The Object Store is RAM that looks like a disk. Anything copied here will be deleted when a cold-boot is performed on the 700 Series Computer.
- If the optional SecureDigital or CompactFlash storage card is in the system, then consider this card the primary location for placing an applications install files. The following folders represent either card:

  The SecureDigital storage card creates the "\SDMMC Disk" folder.

  The CompactFlash storage card creates the "\Storage Card" folder.

  Files copied to either of these locations will be safe when a cold-boot is performed on a 700 Series Computer — *providing the AutoRun system is installed onto the storage card*. You can find this system on the Tools CD. Copying a CAB file to the "\CABFILES" folder on one of these cards will automatically extract that CAB file on every cold boot to ensure that your system is properly set up. See page 3-11 for more details on how this works.

Installing Applications

Consider any of the following options to get the package to the preferred location on your 700 Series Computer.

- Microsoft ActiveSync
- FTP Server *(page 3-3)*
- Application Manager in Unit Manager *(page 3-3)*
- SecureDigital or CompactFlash storage card *(page 3-3)*

Using Microsoft ActiveSync

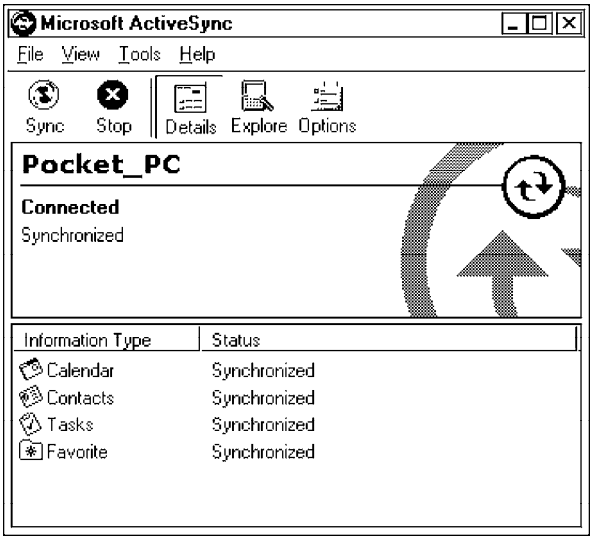
▶ **NOTE:** *These instructions assume that the 700 Color Management Tools portion of the 700 Series Color (700C) Software Tools CD was installed onto your desktop.*

The Microsoft ActiveSync tool is located on the 700C Companion CD, which contains Microsoft products, such as Outlook and ActiveSync. See Section 2, "*Pocket PC 2002*," for information about this tool as provided by Microsoft Corporation.

This can be a serial, USB, Ethernet, InfraRed, or 802.11b ActiveSync connection. Files can then be copied using File Explorer on a PC or a laptop computer. This option is usually only good when updating a few 700 Series Computers.

These instructions assume that Microsoft ActiveSync had been installed onto your desktop computer and is up and running. If not, go to Section 2, "*Pocket PC 2002*," for an URL from which you can download the latest application.

1. Connect your 700 Series Computer to your desktop computer via an ActiveSync cable or IrDA.
2. Wait for a "Connected" message to appear in the Microsoft ActiveSync application to signal a connection to the 700 Series Computer. If necessary, select **File → Get Connected** to initiate a connection.



3. Click **Explore** to access the Mobile Device directory on your 700 Series Computer.
4. From your desktop, select **Start → Windows Explorer**, then browse the applicable path for any of the system files needed for your 700 Series Computer *(listed with their paths)*. Select to highlight the appropriate file, right-click the file for a pop-up menu, then select **Copy**.
   - Base operating system files:
     "C:\Intermec\Intermec 700 Color Mgmt Tools\Drive Images"
   - CAB files:
     "C:\Intermec\Intermec 700 Color Mgmt Tools\Cab Files"
5. Within the Mobile Device directory, go to the directory where you want the files located on the 700 Series Computer, do a right-click for a pop-up menu, then select **Paste**.
6. When all of the files are pasted, perform a warm-boot on the 700 Series Computer. When the computer reboots, wait for the LED on the top left of your keypad to stop blinking. Tap **Start → Programs → File Explorer** to locate the newly copied executable files, then tap these files to activate their utilities.

Using the FTP Server

The 700 Series Computer has a built-in FTP Server that connects to a network via Ethernet or 802.11b. This "ftp"s to the IP address of the 700 Series Computer and places files. The benefit of using FTP is that a script can be created that will automate the process of copying files to the 700 Series Computer. This option is good for when a large number of 700 Series Computers need to be updated. See Section 7, "*Programming*," for more information.

Using the Application Manager in Unit Manager

This requires the 700 Series Computer to connect to the network via Ethernet or 802.11b. The process is still manual so it would take longer than the FTP method but it would still be a better option than ActiveSync where many 700 Series Computers need to be updated. See the *Software Tools Help* for information.

Using a Storage Card

The following steps pertain to installing an application using a storage card.

Copying to a CompactFlash Card

Follow the steps below to install your application on the device using a CompactFlash storage card:

1. Suspend the 700 Series Color Mobile Computer and remove its CompactFlash drive, which holds a SanDisk CompactFlash storage card.
2. Using a CompactFlash Adapter card, place the CompactFlash Drive in your desktop PC card drive.
3. Create a subdirectory on the PCMCIA CompactFlash drive in which to store your application.
4. Add the autorun system to the storage card using the CEImager application. See the *Software Tools Help* for information about CEImager.
5. Copy your application, data files, and all required DLLs and drivers to the subdirectory created on the CompactFlash drive.

6. Add your application to the AUTOUSER.DAT file on the "\Storage Card\2577" directory that contains the following statement, where *your directory* is the directory on the CompactFlash storage card where the application was installed, and *yourapp.exe* is the name of your application. Finish the "RUN=" statement with a carriage return linefeed combination. There may be multiple run statements in the file.

   ```
   RUN=\<your directory>\<yourapp.exe>
   ```

7. Remove the CompactFlash drive from your desktop computer and reinstall it into the 700 Series Computer.
8. Warm-boot the 700 Series Computer to add these files to the CompactFlash storage card.

If the AUTOUSER.DAT file is found and the "RUN=" statement is correct, then the task manager will launch and execute your program on startup.

Copying to a SecureDigital Storage Card

Do the same steps as for the CompactFlash storage Card, except replace the "\Storage Card\2577" directory with the "\SDMMC Disk\2577" directory.

Updating the System Software

▶ NOTE: *Before performing this update, please ensure that the 700 Series Computer to be updated has a fully charged battery. A power failure during the update will corrupt the computer flash, thus the computer would not boot. You will have to send the 700 Series Computer to service to recover the computer flash.*

Use any of the following instructions to update your 700 Series Computer:

- Microsoft ActiveSync
- SecureDigital or CompactFlash storage card *(page 3-6)*

Updating with Microsoft ActiveSync

▶ NOTE: *These instructions assume that the 700 Color Management Tools portion of the 700 Series Color (700C) Tools CD was installed onto your desktop.*

The Microsoft ActiveSync tool is located on the 700C Companion CD, which contains Microsoft products, such as Outlook and ActiveSync. See Section 2, "*Pocket PC 2002*," for information about this tool as provided by Microsoft Corporation.

This can be a serial, USB, Ethernet, or 802.11b ActiveSync connection. Files can then be copied using File Explorer on a PC or laptop computer. This option is usually only good when updating a few 700 Series Computers.

The upgrade will install the latest version of bootloader and operating system. To see what version is on your 700 Series Computer, tap **Start** → **Internet Explorer** → the **Intermec** logo.

These instructions assume that Microsoft ActiveSync had been installed onto your desktop computer and is up and running. If not, go to Section 2, "*Pocket PC 2002*," for an URL from which you can download the latest application. Use Microsoft ActiveSync to update the bootloader in the Flash ROM on the 700 Series Computer as instructed below:

1. Connect your 700 Series Computer to your desktop computer via an ActiveSync cable or IrDA.
2. Wait for a "Connected" message to appear in the Microsoft ActiveSync application to signal a connection to the 700 Series Computer. If necessary, select **File → Get Connected** to initiate a connection (see page 3-2).

3. Click **Explore** to access the Mobile Device directory on your 700 Series Computer.
4. From your desktop, select **Start → Windows Explorer** to browse the "C:\Intermec\Intermec 700 Color Mgmt Tools\Drive Images" directory. Select to highlight the appropriate file, right-click the file for a pop-up menu, then select **Copy**.
   - If you want to update both the bootloader and the operating system image on your 700 Series Computer, then double-click the "\AutoFlash" folder, and copy the AUTOFLASH.NB0 file.
   - If you want to update a particular region in your operating system, then double-click the "\BinBackup" folder and copy the respective .BIN file and the EFLASH.EXE executable file. *See page 3-6 for more information about EFlash*.
   - To update just the bootloader, double-click the "\Eboot" folder and copy the EBOOT.NB0 file.
5. Within the Mobile Device directory, go to the **root** of the 700 Series Computer, do a right-click for a pop-up menu, then select **Paste**.
6. From the 700 Series Computer, tap **Start → Programs → File Explorer**. Tap **My Documents → My Device**, then double-tap EFLASH.EXE to execute the flash. *Do **not** cold-boot the 700 Series Computer as instructed via the Flash program.*
7. If the 700 Series Computer has a laser scanner and a S9CUPGRADEUTILITY.CAB file is in the "\Intermec 700 Color Mgmt Tools\Cab Files\Programs" directory on the 700C Tools CD, you will need to upgrade the scanner firmware. Copy this CAB file to the 700 Series Computer, then extract this file. The S9CUpgradeUtility application will start automatically. Tap **Upgrade** to initiate the process.
8. Copy any of the additional system files needed for your 700 Series Computer *(listed with their paths)*:
   - Base operating system files:
     "C:\Intermec\Intermec 700 Color Mgmt Tools\Drive Images"
   - CAB files:
     "C:\Intermec\Intermec 700 Color Mgmt Tools\Cab Files"
9. When all of the files are pasted, perform a cold-boot on the 700 Series Computer. When the computer reboots, wait for the LED on the top left of your keypad to stop blinking. Tap **Start → Programs → File Explorer** to locate the newly copied executable files, then tap these files to activate their utilities.

After the update, you can remove the AUTOFLASH.NB0, EFLASH.EXE, or EBOOT.NB0 file from the 700 Series Computer.

Updating with a Storage Card

To update the 700 Series Computer image, copy the AUTOFLASH.NB0 file to either a CompactFlash or SecureDigital storage card, insert this card into the 700 Series Computer, then perform a cold-boot (or hard reboot) on the 700 Series Computer.

Do the following to automatically update the system software on a 700 Series Computer. *Note that the 700 Series Computer needs to be on power while upgrading the operating system to prevent corrupting the ROM. Should ROM be corrupted, you will have to send your 700 Series Computer in to be serviced.*

1. Copy the AUTOFLASH.NB0, FLASHKBD.EXE, and 740CUP.KBF files to the root of either the SecureDigital or CompactFlash storage card. See page 3-2 to choose a method with which to copy these files.
2. Place AUTORUN.EXE and AUTORUN.DAT in the "\2577" directory on the storage card.
3. Put FLASHKBD.EXE and 740CUP.KBF in the "\Update" directory on the SecureDigital storage card or the "\Update" directory on the CompactFlash storage card.
4. Edit the AUTORUN.DAT file to include the following command. Include "SDMMC Disk" if you are using a SecureDigital storage card, or "Storage Card" if you are using a CompactFlash storage card:

```
RUN   "\SDMMC Disk\Update\FlashKbd.exe" \SDMMC Disk\Update\740cup.kbf
```

▶ **NOTE:** *Use quotes in the first path — do **not** include quote marks in the second path.*

5. Perform a cold-boot on the 700 Series Computer. The Autoflash procedure will automatically start on the cold-boot. The 700 Series Computer will cold-boot again when the process is complete.
6. The files that were copied to the storage card can now be deleted.

EFlash

EFlash is a process that reflashes the Pocket PC operating system, in whole or in part, to nonvolatile flash memory. Its primary purpose is to install operating system upgrades on ROM-based systems.

The biggest advantage EFlash has over other flashing techniques is that it can be run remotely over FTP (or other remote support service) and requires no user interaction at the level of the 700 Series Computer. Thus system upgrades can be managed from a remote central location. Normally, this would be done off-duty while 700 Series Computers are docked. EFlash also has the advantage in that there is no restriction as to what part of the system can be upgraded.

How it Works

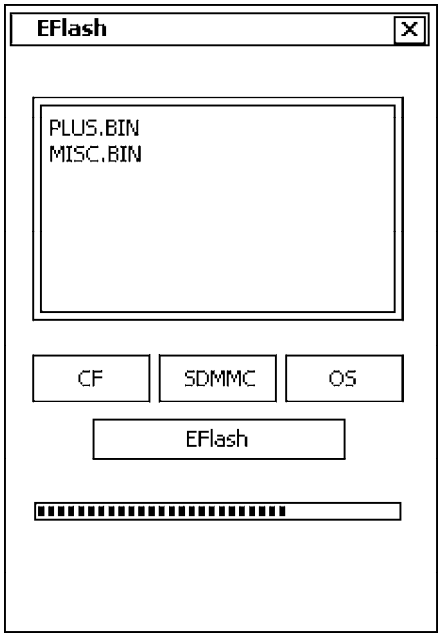
An upgrade file must exist in the root of the Object Store or the root of a storage card, such as CompactFlash or SecureDigital. EFlash supports BIN, LST, and NB0 files *(see "File Types" on page 3-8)*.

1. When EFLASH.EXE is executed on a 700 Series Computer, as described in step 4 on page 3-5, the following dialog appears with a list of available upgrade files. Select the file to be flashed onto the 700 Series Computer, then tap the appropriate button, then tap **EFlash** to flash the selected file.

   CF
   To flash the upgrade file onto a CompactFlash storage card.

   SDMMC
   To flash the upgrade file onto a SecureDigital storage card.

   OS
   To flash the upgrade file into the root of the Object Store directory.



   The name of the desired upgrade file can also be supplied via a command line argument, like the following example. Using the command line argument to specify an upgrade file bypasses the selection dialogue.

   ```
   EFLASH -F\sdmmc disk\misc.bin
   ```

2. After selecting an upgrade file, the file is copied into memory and a special warm boot is performed. During this warm boot, the file that was copied into memory is written to flash according to the rules of the file type selected.
3. Once the reflash is completed without error, a cold boot is performed, a standard procedure. *Note that data in volatile RAM will be lost during this cold-boot.* After the cold-boot, the 700 Series Computer is ready for use.
4. If this process is done remotely, download the upgrade file before running the EFlash process. This process will usually execute remotely with the downloaded upgrade file included as a command line argument.

File Types

EFlash supports BIN, LST, and NB0 files. Their characteristics are as follows:

BIN
A file with an extension of .BIN also contains ROM image data, but is more compact than an NB0. A BIN file includes only the ROM data, which needs to be programmed, eliminating large areas of unnecessary programming. BIN files also contain information as to where exactly the ROM data is to be placed in flash as well as error detection. BIN files are the most common type of upgrade file.

LST
A file with an extension of .LST is simply an ASCII text file containing zero or more BIN and NB0 files to be flashed. Use this type of file when you want to reflash more than one upgrade file in a single session. Only BIN and NB0 files can be in the list file. You cannot call a list file from within another list file.

NB0 *(NBzero)*
A file with an extension of .NB0 is a binary ROM image. Its contents are an exact copy of how flash should look after it is written. These files tend to be bigger than BIN files and have less error checking and are thus less desirable than the BIN files. NB0 files are always written starting at the beginning of flash. The boot loader is typically an NB0 file.

Command Line Syntax

EFlash can be invoked as process with the following syntax:

```
EFlash    [-f<filename>]
```

where the optional *-f<filename>* argument specifies the file to flash (BIN, LST, or NB0). When specified, EFlash bypasses the file selection dialog and runs in noninteractive mode.

IFTP Server Command Line Syntax

EFlash can be invoked as process with the following syntax after logging into a 700 Series Computer using any conventional FTP client that allows interactive mode. The following site command can be used with Microsoft FTP Client:

```
ftp> QUOTE SITE RUN [program] [arguments]
```

An example of a session is as follows:

```
C:\>ftp 136.179.84.165
Connected to 136.179.84.165.
220 Connected to Intermec IFTP Server. Version 2.01.
User (136.179.84.165:(none)): intermec
331 Password required!
Password: cr52401
230-============================
     Intermec FTP Server ready!
     Version 2.01.  WINCE
     00C0B26E8C47 6024866 740
     Binary transfer mode only!
    ============================
230 Logon successful.
ftp> put \kernel.bin
200 Port command successful.
150 STOR: Command accepted!
226 Transfer Complete.
ftp: 2341747 bytes sent in 40.31Seconds 58.10Kbytes/sec.
ftp>
ftp> quote site run \windows\eflash.exe -f\kernel.bin
250 \eflash.exe launched, process id is 0x4053038E
ftp>ls
Connection closed by remote host.
```

Application Migration

▶ **NOTE:** *These instructions assume that the 700 Color Management Tools portion of the 700C Tools CD was installed onto your desktop and that a storage card has been added to the base configuration of the 700 Color Computer.*

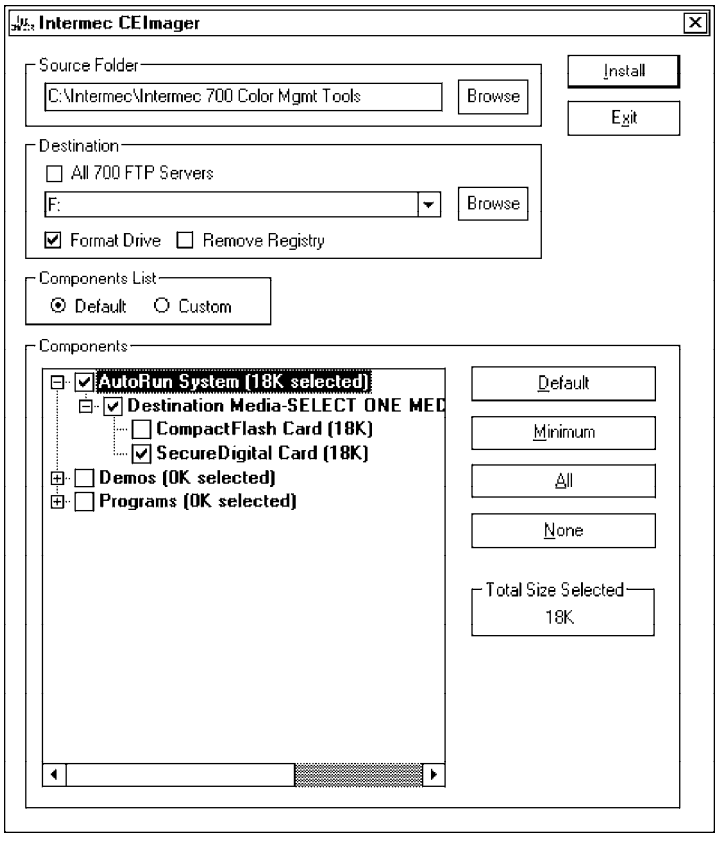
The following steps are required to ensure that the following will happen on a cold-boot:

- CAB files can be restored,
- applications will automatically start,
- and the registry will be restored.

Do the following for the cold-boot procedure:

Intermec CE Imager

1. From your desktop, double-click the **Intermec CE Imager** desktop icon to access the Intermec CEImager application. If this icon is not on your desktop, then double-click the CEIMAGER.EXE executable from the "C:\Intermec\Intermec 700 Color Mgmt Tools\Tools\CEImager" folder.
2. Click **Default** under **Components List** to activate the components.
3. Click (+) to expand the **AutoRun System** component, click (+) to expand the **Destination Media** option, then select *either* the **CompactFlash Card** option or the **SecureDigital Card** option. *Do not select both storage cards, as the AutoRun files copied will work for one storage card, but* ***not*** *work on the other storage card.*



4. Click **Install** to install the AUTORUN files onto the storage card.
5. Create a "\Cabfiles" folder on the storage card. Copy any CAB files that are to be extracted on every startup into this folder.

6. In the "\2577" directory, add your custom AUTOUSER.DAT file. See the *Software Tools Help* for more information on how to set up an AUTOUSER.DAT file.
7. If you are using the RegFlushKey() API, the application must use a special API to make sure the registry is written to the appropriate card; or you can use the **Utilities** control panel applet, as follows:

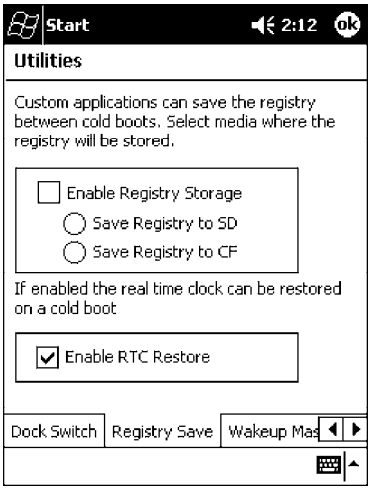
a. From the 700 Series Computer, tap **Start** → **Settings** → the **Utilities** icon → the **Registry Save** tab.

b. Tap **Enable Registry Storage**, then tap either of the following:

Save Registry to SD
To write the registry to the SecureDigital storage card.

Save Registry to CF
To write the registry to the CompactFlash storage card.

c. Tap **ok** to save your entry and exit the **Utilities** control panel applet.



▶ **NOTE:** *If you are using a SecureDigital storage card, you must change any disk access from "Storage Card" to "SDMMC Disk."*

8. Remove the storage card from the desktop PC and install the card into the 700 Series Color Computer.
9. Perform a cold-boot on the 700 Series Color Computer. Files will automatically install from the storage card upon reboot. Any calls to the RegFlushKey() API will automatically write the registry to the appropriate location.

When converting a 700 Series Monochrome Computer application to run on the 700 Series Color Computer, most APIs should work without changes. Below are a few exceptions:

- The 700 Series Monochrome Computer used the "\Storage Card" folder for nonvolatile storage. You may need to change the application to store data in a volatile location or onto the "SDMMC Disk" if a SecureDigital storage card is present in the system.
- If the application uses the RegFlushKey() API, it must first verify that the proper media is available in the system and call the special API mentioned in Step 7 on the previous page.
- If the application will be using the 700 Color switchable dock, use the API to set the proper port on the dock before communications.

- Some WAN radio options have changed. Review the WAN radio section to determine if any changes will be required in your application.
- The arrow and tab keys are swapped from the way they were on the 700 Series Monochrome Computers. Keyboard remapping is available on the 700 Series Color Computer if these keys need to be changed. See page 3-4 for more details.
- No special SDK is needed to compile applications for the Xscale processor. Targeting the SA1110 processor will create applications that run on the 700 Series Color Computer.

Cabinet File Installation

CAB files *(short for cabinet files)* are like .ZIP files, plus they register DLLs, create shortcuts, modify registry entries, and run custom set up programs. Tap a CAB file to extract that file or place the CAB file on one of the approved storage devices in the "\Cab Files" folder, then perform a warm-boot on the 700 Series Computer. There are two methods available to extract a CAB file:

- Tap a CAB file to extract it. When using this method, the CAB file is automatically deleted when the extraction process is successful, *unless* the CAB file is set with the read-only attribute.
- Use the AUTOCAB method where all files are extracted when a cold-boot is performed on the 700 Series Computer. This AutoCab application is on the Tools CD, see its "*Software Tools Help*" for more information.

Section 4

Network Support

The 700 Series Color Mobile Computer can integrate up to three radios in a single unit, and will automatically install the appropriate software for radio use when the unit is powered on. The Intermec CORE application defaults to the most recently used module. If a module has not yet been used or set, CORE will default to the first module as listed alphabetically.

The following communication options on the 700 Series Computer provide wired and wireless connectivity:

Onboard wired Ethernet *(standard)*

Wireless Local Area Network (LAN)
This 802.11b radio option provides up to 11 Mb/sec throughput.

Wireless Wide Area Network (WAN)
Includes support for GSM/GPRS and CDMA/1xRTT radios.

Wireless Printing
This allows for cable-free communications with peripheral devices, such as printers, over a ten-meter range. This compatibility is provided via a Bluetooth qualified module by Socket Communications.

CORE

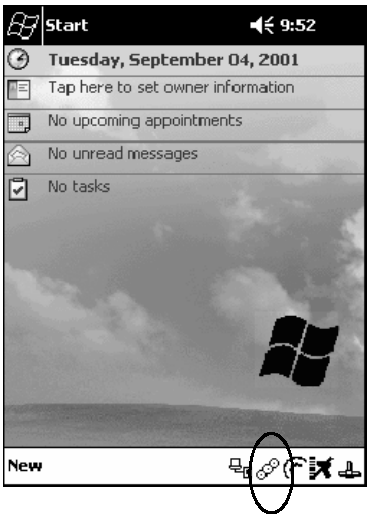
The Intermec Common Object Resource Environment (CORE) application provides a framework for various modules that let you configure and manage your Intermec products. These modules are software plug-ins that can be configuration tools, such as the 802.11b radio configuration module, or they can provide information on your environment, such as a battery life module.

CORE is built into the operating system of every 700 Series Computer. On the 700 Series Computer, tap **Start** → **Programs** → the **Intermec CORE** icon to access this application.

CORE modules are collections of specific information. This information is usually related to a particular radio technology, but not always. Each module can display general and detailed information. Tap the **General** and **Details** tabs near the bottom to switch between general and detailed information. Note that not all modules will have detailed information.

To learn more about this application, see its online help. Tap **Start** → **Help** from the menu to see the CORE online help.

▶ **NOTE:** *Once CORE is running, you can return to it by tapping its icon from the System Tray via the Today screen. Tap* ***Start*** *→* ***Today*** *→ the* ***Intermec CORE*** *three-ring icon (circled in the following illustration).*



Network Adapters

Your 700 Series Computer can have up to three radios installed. The default network adapter or radio is dependent on what card is inserted in your 700 Computer. Below are the the network adapters that exist as of this publication. See the Developer's Support web site for the latest information on network adapters for your unit.

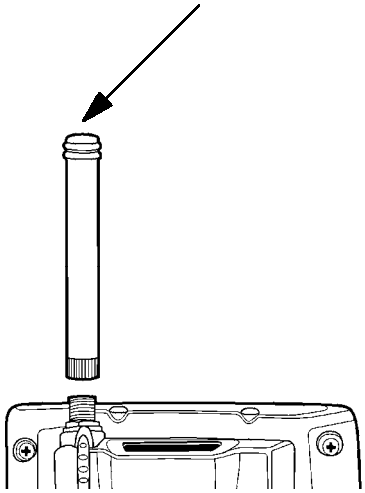
- Ethernet Communications *(LAN9000)* — page 4-3.
- 802.11b Radios *(802.11b Wireless LAN driver)* — page 4-4.
- WWAN *(Wireless WAN)* — page 4-25.
- Wireless Printing *(PAN)* — page 4-34.

Note that the tip of the antenna attached to your 700 Series Computer is color-coded to identify its radio type. Refer to the following to determine your radio type:

Green
802.11b diversity

Red
CDMA/GPRS US/Canada

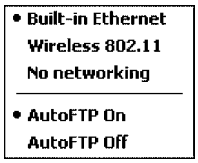
Blue
GPRS International



Ethernet Communications

Follow the steps below to start Ethernet communications on the 700 Series Computer. If your system does not contain an 802.11b radio, then **Ethernet networking using DHCP** will be selected as the default.

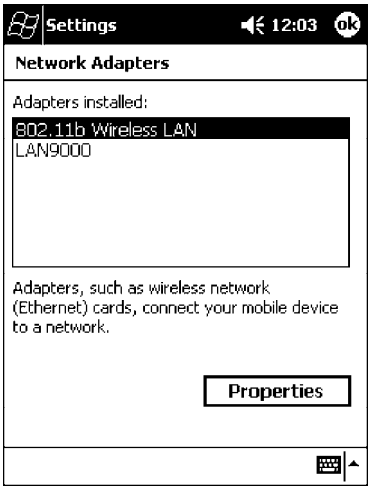
When "Built-in Ethernet" is selected from the NDISTRY pop-up menu, then the antenna shown to the left will appear in the System Tray. When "No networking" is selected, then this icon will appear with a red "X" above it.



Network Adapters

From the 700 Series Computer, tap **Start** → **Settings** → the **Connections** tab → **Network Adapters** to access the network connections for this unit. Make the changes necessary for your network, then tap **ok** when finished.

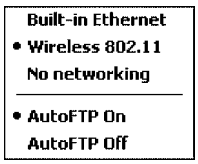
▶ **NOTE:** *"LAN9000" is for Ethernet and "802.11b Wireless LAN" is for 802.11b radios.*



802.11b Communications

The 700 Series Computer can integrate the 802.11b radio module along with either the GSM/GPRS or the CDMA/1xRTT radio and the Wireless Printing option. The 802.11b radio module accommodates any Wireless LAN (WLAN) requirements, such as using WLAN access points for cross-docking or load-planning applications.

When "Wireless 802.11" is selected via the NDISTRY pop-up menu, then the antenna shown to the left will appear in the System Tray.



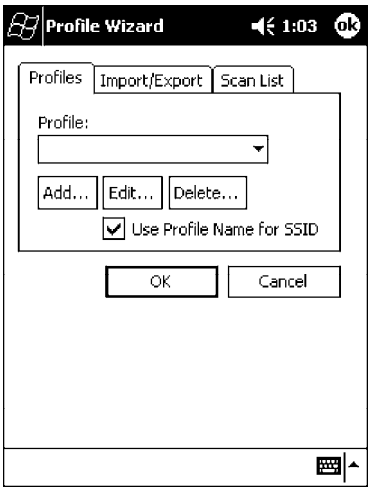
Wireless Network

To start 802.11b communications on the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Wireless Network** icon to access the Profile Wizard for the 802.11b radio module. The Profile Wizard defaults to the Profiles page.

Profiles

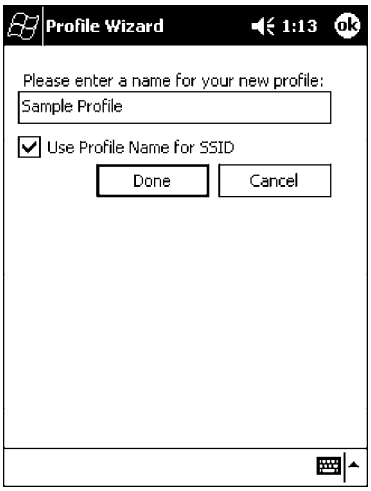
Use the Profiles page to add, edit, or delete multiple networking environments for this 802.11b radio. To add a profile from this screen, enter up to 32 alphanumeric characters in the **Profile** field, then tap **Add**. See "*Basic*" on page 4-6 and "*Security*" on page 4-7 for more information.

Leave **Use Profile Name for SSID** checked for the SSID (or Network Name) to use this profile name. If this is cleared (check mark removed), the SSID will default to using the factory-assigned "INTERMEC" network name.



To add a profile:
Tap **Add**, enter up to 32 alphanumeric characters to name this profile if you have not already entered a description in the Profiles page, configure the basic and security information for this profile, then click **Done** to configure its basic and security information.

Leave **Use Profile Name for SSID** checked for the SSID to use this assigned profile name. If this is cleared (check mark removed), the SSID will default to using the factory-assigned "INTERMEC" network name. Go to page 4-6 to continue.



To edit a profile:
Select an existing profile from the **Profile** drop-down list, tap **Edit**, make the changes needed for this profile (starting on page 4-6), then tap **OK** to return to the Profiles page.

To delete a profile:
Select a profile from the **Profile** drop-down list, tap **Delete**, then tap **Yes** to remove the selected profile.

Basic

Use the Basic page to set the network type, transmission rate, and radio channel for this profile. Click **OK** to return to the Profiles page.

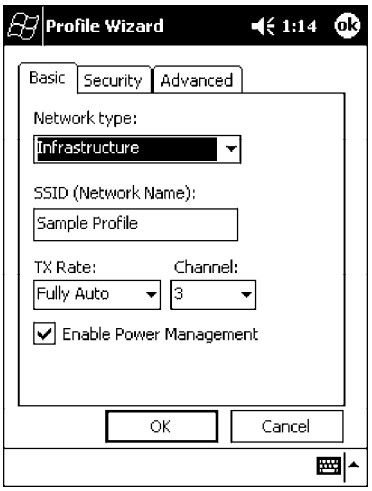
Network type:
Tap the drop-down list to select either Infrastructure or Ad-hoc.

SSID (Network Name):
This assumes the profile name when **Use Profile Name for SSID** is checked on the previous screen*, 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.

TX Rate:
Tap the drop-down list to select the speed at which to transmit connections: Fully Auto *(default)*, 11 Mbps, 5.5 Mbps, 2 Mbps Auto, 2 Mbsp, or 1 Mbps.

Channel:
If "Ad-hoc" were selected as the network type, then this is enabled. Tap the drop-down list to select a channel, from 1–15, through which to handle connections *(default is 3)*.

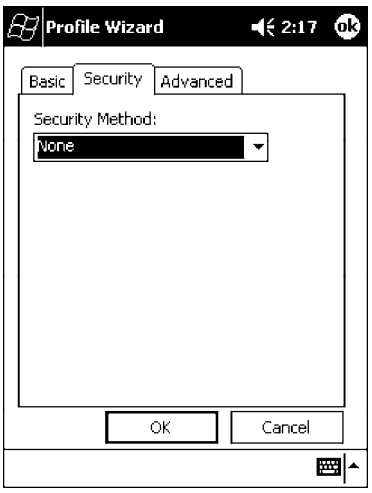
Enable Power Management:
Check this box to conserve battery power *(default)*, or clear this box to disable this feature.



Security

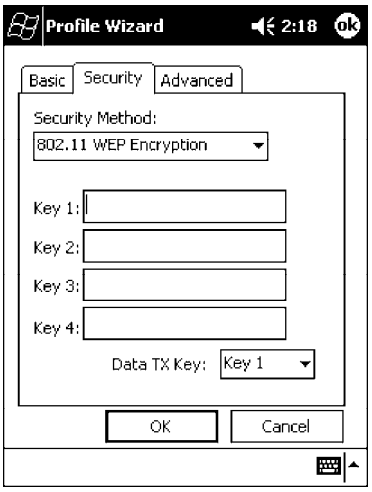
Use the Security page to set this profile as read-only or to enable WEP (Wired Equivalent Privacy) encryption. Click **OK** to return to the Profiles page. The following securities are available from the **Security Method** drop-down list. *Note that the last three methods are available if you have purchased the security package. Contact your Intermec Representative for more information.*

- 802.11 WEP Encryption
- 802.1x TLS *(page 4-8)*
- 802.1x TTLS *(page 4-9)*
- LEAP *(page 4-9)*



802.11 WEP Encryption:
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). These can be entered as either ASCII (12345) or Hex (0x3132333435).

To enter WEP keys, select "802.11 WEP Encryption" from the **Security Method** drop-down list. 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.



802.1x TLS *(Transport Layer Security)*:
TLS is a protocol that ensures privacy between communicating applications and their users on the Internet. To use this protocol, select "802.1x TLS" from the **Security Method** drop-down list, then enter the following:

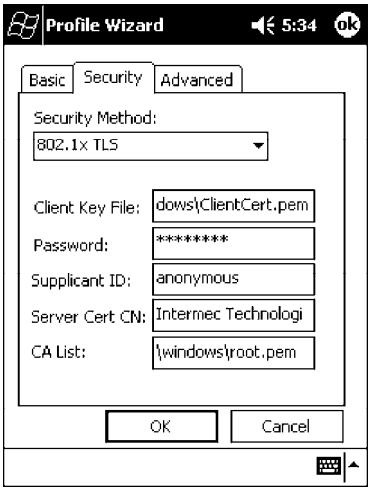
Client Key File:
Enter the file location where the certificate for your identity is stored.

Password:
Enter the password for the certificate in this field.

Supplicant ID:
Enter the user ID associated with this certificate.

Server Cert CN *(Certificate Common Name)*:
Enter the common name of your authentication server.

CA List *(Certificate Authority)*:
Enter the file location, or path, of the server certificates.



802.1x TTLS *(EAP-Tunneled TLS)*:
To use this protocol, select "802.1x TTLS" from the **Security Method** drop-down list, then enter the following:

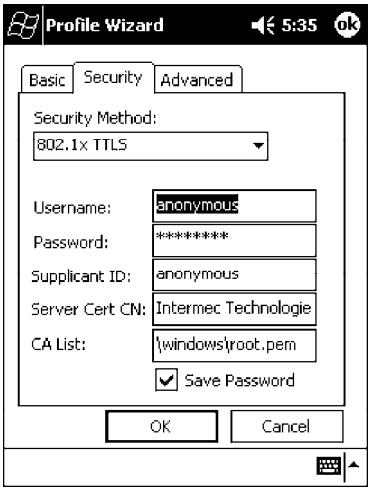
Username:
Enter your user name for this security protocol.

Password:
Enter your password for this security protocol.

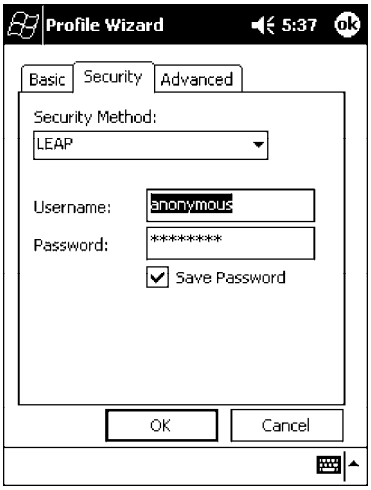
Supplicant ID:
Enter "anonymous" unless your administrator indicates otherwise.

Server Cert CN *(Certificate Common Name)*:
Enter the common name of your authentication server.

CA List:
Enter the file location, or path, of the server certificates.



LEAP *(Cisco Wireless EAP (Extensible Authentication Protocol))*:
Enter your unique user name and password to use this protocol.



Advanced

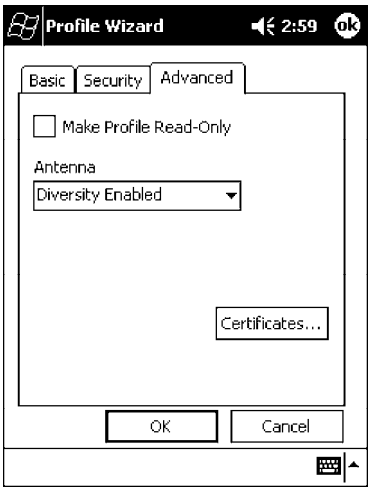
Use this page to secure the configuration for this profile, to make all fields read-only and to select the antenna to be used with this profile.

Make Profile Read-Only:
Check this box, then enter and reenter a password to "lock" or render "read-only" all configurations for this profile. To reverse this step, clear the check box, then enter the password assigned with the "read-only" status.

Antenna:
Select from the drop-down list the antenna to be used by this profile:

"Diversity Enabled"	Both the internal and external antennas.
"Primary Only"	Just the internal antenna.
"Secondary Only"	Just the external antenna.

Certificates:
If "802.1x TLS," "802.1x TTLS," or "LEAP" were enabled via the **Security** tab, then this button will appear. Tap this button to configure the available certificates. See "*Certificates*" on the next page for more information.



Certificates

Use this page to view, modify, or remove certificates assigned to your particular security method. *Note that you can also access this page by tapping* ***Start → certificates*** *from the Today screen.*

CA Name/IP:
Enter a valid CA name or IP address assigned to the certificate in question. This allows you to enroll the certificate or to browse for its latest information.

Enroll File Name:
Enter the file name of the certificate to be enrolled.

Store Location:
Enter the path where the certificate is to be stored within your 700 Series Computer.

Enroll:
Tap this to assign the file entered in **Enroll File Name** to the location specified in **Store Location**.

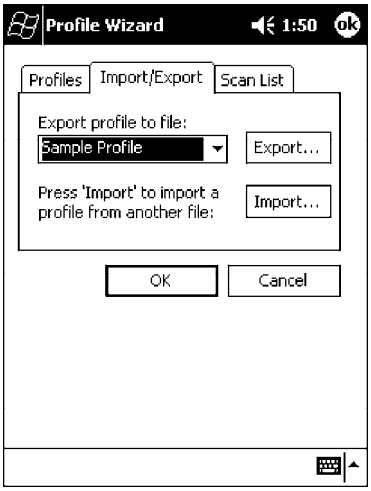
CA Vert:
Tap this to view the contents of the certificate via the Internet Explorer.

View:
Tap this to view information about the certificate, such as to whom this certificate was issued, who issued the certificate, and the span of time the certificate is valid.

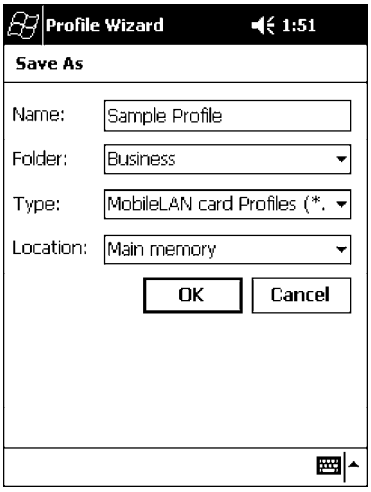
Remove:
Select a file from the list, then tap this button to delete that file.

Import/Export

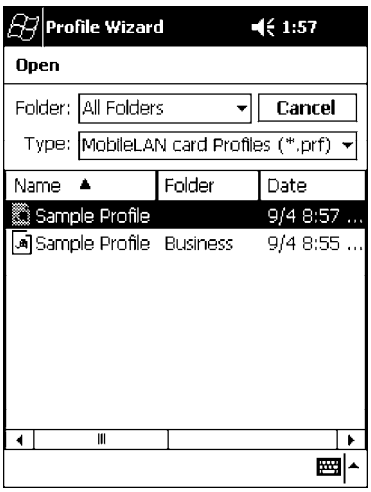
Use this page to send a profile or to retrieve a profile to or from another location within your 700 Series Computer.



To export a profile:
Select to highlight a profile, then tap **Export**. Select from the drop-down lists, the folder, type of files, and location within the folder where the profile is to go, tap **OK** to export the profile, tap **ok** to close the confirmation screen, then tap **OK** again to exit the Profile Wizard.



To import a profile:
Tap **Import** to access the Open screen, from the drop-down lists, select a folder and file type, then tap a profile from the list provided. Tap **ok** to close the confirmation screen, then tap **OK** again to exit the Profile Wizard.



Scan List

Use this Scan List page to monitor network connections, and if lost, to attempt to reestablish connections with these networks.

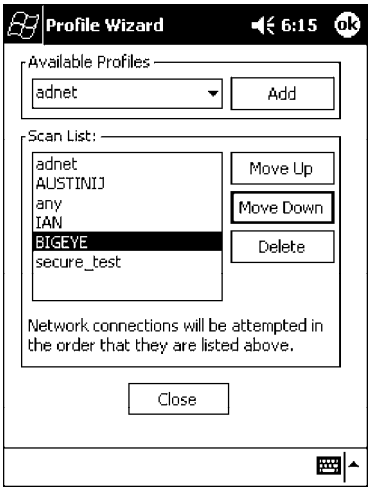
Selected Profile

Select this option to use the profile defined in the Profiles page, then tap **OK** to exit the Profile Wizard. When connections are lost, attempts will be made to connect to the specified profile.

Scan List

Use this option to select a number of profiles with which to establish connections. When connections are lost, attempts will be made to contact each of the profiles listed, in the order they appear in the list.

1. Tap this option, then tap **Edit Scan List**.
2. Select profiles from the **Available Profiles** drop-down list, then tap **Add** to include each selection in the Scan List.
3. To arrange the hierarchy of profiles, tap to select a profile, then tap either **Move Up** or **Move Down** to move each profile. To remove a profile from the list, tap to select that profile, then tap **Delete**.
4. Click either **ok** or **Close** to return to the Scan List page, then click **OK** to exit the Profile Wizard.



API

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 will also read/write 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 that has the 802.11b networking installed.

NETWLAN.DLL
This file is the 802.11b driver. It will be 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.

The 80211API.DLL supports an unlimited number of 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 4-22 for more information.

Function Summary

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.

Syntax:

UINT **RadioConnect( );**

Returns:

ERROR_SUCCESS when successful, otherwise ERR_CONNECT_FAILED.

RadioDisconnect()

Cleans up the connection from the RadioConnect() function after an application closes.

Syntax:

UINT **RadioDisconnect( );**

Returns:

ERROR_SUCCESS when successful, otherwise ERR_CONNECT_FAILED.

GetMac()

Gets the radio MAC address.

Syntax:

UINT **GetMac(** TCHAR * **);**

Parameters:

Pointer to a character array, which is populated with MAC addresses.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetBSSID()

Gets the associated access point name, the BSSID.

Syntax:

UINT **GetBSSID(** TCHAR * **);**

Parameters:

Pointer to a character array, which is populated with the current BSSID.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed

GetSSID()

Gets the current SSID (network name).

Syntax:

UINT **GetSSID(** TCHAR * **);**

Parameters:

Pointer to a character array, which is populated with the current SSID.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetLinkSpeed()

Retrieves the current link speed of the radio connection.

Syntax:

UINT **GetLinkSpeed(** int *&* **);**

Parameters:

& References an integer.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetNetworkType()

Retrieves the network type.

Syntax:

UINT **GetNetworkType(** ULONG *&* **);**

Parameters:

& References a ULONG value, populated with one of the following:

NDIS_NET_TYPE_FH Frequency Hopping Radio
NDIS_NET_TYPE_DS Direct Sequence Radio
NDIS_NET_TYPE_UNDEFINED
Unknown or information not available.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetTXPower()

Gets the current TX power of the radio in milliwatts.

Syntax:

UINT **GetTXPower(** ULONG *&* **);**

Parameters:

& References a ULONG value, populated with one of the following in milliwatts (mW):

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.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetNetworkMode()

Retrieves the network mode.

Syntax:

UINT **GetNetworkMode(** ULONG *&* **);**

Parameters:

& References a ULONG value, populated with one of the following:

NDIS_NET_MODE_IBSS	802.11 Ad-Hoc Mode.
NDIS_NET_MODE_ESS	802.11 Infrastructure Mode.
NDIS_NET_MODE_UNKNOWN	Unknown Value or Error.
NDIS_NET_AUTO_UNKNOWN	Automatic Selection. *Use of this option is not recommended.*

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

SetNetworkMode()

Sets the radio and updates the CE registry.

Syntax:

UINT **SetNetworkMode(** ULONG *&* **);**

Parameters:

& References a ULONG value, populated with one of the values defined in GetNetworkMode().

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

AddWep()

Adds a WEP key to the radio configuration.

Syntax:

UINT **AddWep(** ULONG *1*, BOOL *2*, TCHAR * *3* **);**

Parameters:

ULONG Pointer that identifies what key to be set.

BOOL Specifies whether the key being set is the default TX key.

TCHAR Pointer that specifies the key data either in hex (string lengths of 10 or 26) or ASCII (string lengths of 5 or 13).

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetRSSI()

Sets the current RSSI (Received Signal Strength Indication).

Syntax:

UINT **GetRSSI(** ULONG & **);**

Parameters:

& References a ULONG value.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetAssociationStatus()

Gets the current connection, or association status.

Syntax:

UINT **GetAssociationStatus(** ULONG *&* **);**

Parameters:

& References a ULONG value, a current connection status as follows:

NDIS_RADIO_ASSOCIATED
The radio is associated with an access point.

NDIS_RADIO_SCANNING
The radio is scanning for an available network.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetWepStatus()

Gets the current WEP status.

Syntax:

UINT **GetWepStatus(** ULONG *&* **);**

Parameters:

& References a ULONG status value which include the following:

NDIS_RADIO_WEP_ENABLED WEP is currently enabled.
NDIS_RADIO_WEP_DISABLED WEP is currently disabled.
NDIS_RADIO_WEP_ABSENT WEP key is absent.
NDIS_RADIO_WEP_NOT_SUPPORTED
WEP is not supported.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetAuthenticationMode()

Gets the current authentication mode.

Syntax:

UINT **GetAuthenticationMode(** ULONG *&* **);**

Parameters:

& References a ULONG value which include the following current authentication mode:

NDIS_RADIO_AUTH_MODE_OPEN
Open System is in use.

NDIS_RADIO_AUTH_MODE_SHARED
Shared Key is in use.

NDIS_RADIO_AUTH_MODE_AUTO
Automatic Detection.

NDIS_RADIO_AUTH_MODE_ERROR
Unknown value or Error.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

SetAuthenticationMode()

Sets the radio authentication mode with a value defined in the GetAuthenticationMode() function.

Syntax:

UINT **SetAuthenticationMode(** ULONG **);**

Parameters:

Passes in a ULONG set to one of the values as defined in GetAuthenticationMode().

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

SetChannel()

Sets the radio channel, ranging from 1 to 14.

Syntax:

UINT **SetChannel(** USHORT **);**

Parameters:

USHORT set to a desired channel (1–14).

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

EnableWep()

Enables or disables WEP encryption.

Syntax:

UINT **EnableWep(** BOOL **);**

Parameters:

Set to TRUE (0) to enable WEP encryption or FALSE (1) to disabled WEP encryption.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

GetPowerMode()

Gets the current power management mode of the radio.

Syntax:

UINT **GetPowerMode(** ULONG *&* **);**

Parameters:

& References a ULONG value which include the following current radio power management mode:

NDIS_RADIO_POWER_MODE_CAM
Continuous Access Mode (uses most power).

NDIS_RADIO_POWER_MODE_MAX
Maximum Power Savings.

NDIS_RADIO_POWER_MODE_PSP
Power Saving Mode, best balance of power and performance.

NDIS_RADIO_POWER_UNKNOWN
Unknown mode reported or error.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

SetSSID()

Passes the desired SSID (network name).

Syntax:

UINT **SetSSID(** TCHAR * **);**

Parameters:

Pointer to a character array that contains the desired SSID.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

isOrinoco()

Confirms whether the present radio is an ORiNOCO radio.

Syntax:

UINT **isOrinoco( );**

Parameters:

None.

Returns:

TRUE when an ORiNOCO radio.
FALSE when other than an ORiNOCO radio.

EncryptWepKeyForRegistry()

Encrypts a key for registry storage. Requires TCHAR pointers for a destination and a source.

Syntax:

UINT **EncryptWepKeyForRegistry(** TCHAR * *szDest*, TCHAR * *szSource* **);**

Parameters:

szDest String for the destination.

szSource String for the source.

Returns:

ERROR_SUCCESS when successful,
ERR_QUERY_FAILED when the query failed, or
ERR_CONNECT_FAILED if a connection with the radio failed.

SetRTSThreshold()

Sets the radio RTS (Request To Send) threshold.

Syntax:

UINT **SetRTSThreshold(** USHORT *&* **);**

Parameters:

& References a USHORT value.

Returns:

None.

GetRTSThreshold()

Gets the radio RTS threshold.

Syntax:

UINT **GetRTSThreshold(** USHORT *&* **);**

Parameters:

& References a USHORT value.

Returns:

None.

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 string that contains the name of the profile to be activated.

Returns:

None.

StartScanList()

If a scan list is configured, this will start the API looking for a network on that scan list and configuring the radio appropriately. This call can take a long time to process.

Syntax:

UINT **StartScanList( );**

Parameters:

None.

Returns:

None.

802.11b Radio CORE Module

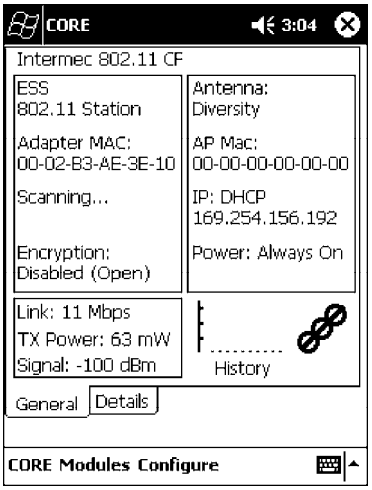
The 802.11b radio CORE module displays helpful information about the 802.11b radio option built into your 700 Series Computer.

Note that you can configure the 802.11b radio module from this CORE application. Select **Configure → Configure 802.11 CF** from the bottom menu bar to access the Profile Wizard application. Information about this application starts on page 4-4.

Select **Modules → Intermec 802.11 CF Help** for more information on the contents of this CORE module.

General

Below are descriptions and meanings for each piece of information provided via the **General** tab, reading from top to bottom starting on the left.



ESS 802.11 Station:
This identifies the type of network to which you are attached, either an ESS (Embedded Security Subsystem) Station, or Ad-hoc.

Adapter MAC:
This identifies the MAC address for this 802.11b adapter.

Scanning:
Status of association. When connected to a network, this changes to "Connected to NET" with NET being the name of the network to which you are connected.

Encryption:
This indicates whether WEP encryption is "Enabled" or "Disabled (Open)." See page 4-7 for more information.

Link:
This indicates the speed at which a connection is made.

TX Power:
This shows the speed (in milliwatts) at which transmissions are made. See page 4-6 for more information.

Signal:
This identifies the radio signal strength (in dBm).

Antenna:
This identifies the antenna being used with the assigned profile. See page 4-10 for more information.

AP Mac:
This identifies the MAC address of the access point to which this 700 Series Computer is connected.

IP:
This provides the IP address which can be set as either DHCP (Dynamic Host Configuration Protocol) or statically.

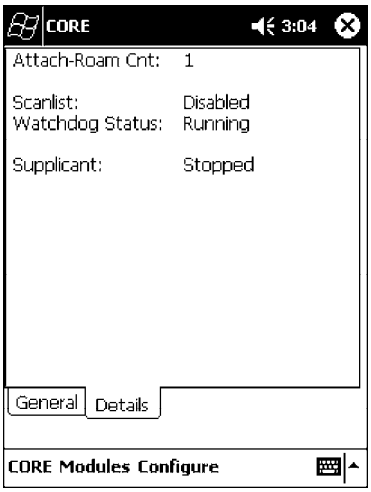
Power:
This indicates the power status of this 700 Series Computer – "Always On" is the default.

History:
This bar graph displays an active history of this radio module's quality of connections.

Friendly Indicator:
If the radio stack is loaded, then all three dots are filled. These dots are left empty if the stack is not loaded. These dots do vary based on the CORE application's perception of the overall connection quality.

Details

Below are descriptions and meanings for each piece of information provided via the **Details** tab, reading from top to bottom.



Attach-Roam Cnt:
This includes the number of new associations made during the current session, including any found roaming.

Scanlist:
This indicates whether the Scan List option was enabled or disabled. See page 4-13 for more information.

Watchdog Status:
This monitors the activity of the Scan List — "Running" or "Stopped."

Supplicant:
This monitors the 802.1x security activity on the client — "Running" or "Stopped."

WWAN Radio Options

The 700 Series Computer can integrate either the GSM/GPRS or the CDMA/1xRTT radio along with the 802.11b radio and the Wireless Printing option. The WWAN radio option accommodates any Wireless WAN requirements, such as taking the 700 Series Computer off the premises in a delivery vehicle to cover a much larger area.

GSM/GPRS

The GSM (Global System for Mobile communications) and GPRS (General Packet Radio Service) wireless infrastructure increases voice capacity, enables personalized "user-aware" services, and creates networking efficiencies to help wireless service providers drive reduced operating costs.

GSM is an open, nonproprietary system. One of its great strengths is the international roaming capability. This provides seamless and same-standardized same-number contactability world-wide. GSM satellite roaming has extended service access to areas with unavailable terrestrial coverage.

GPRS is the high-speed data evolution of GSM. GPRS supports Internet Protocol (IP), enabling access to Internet and intranet content and applications from GPRS wireless devices. The anticipated data rate for GPRS is 115 Kbps and throughput rates of 30–60 Kbps have been achieved initially. This high speed capability enables vehicle applications to become real-time and to use the Internet for access to corporate data or information in the form of traffic or navigation.

CDMA/1xRTT SB555

Code Division Multiple Access (CDMA) is a form of "spread-spectrum," a family of digital communication techniques used in military applications for years. The core principle of spread-spectrum is the use of noise-like carrier waves, and, as the name implies, bandwidths much wider than that required for simple point-to-point communication at the same data rate.

1XRTT, the first phase of CDMA2000, is designed to support up to 144 KB per second packet data transmission and to double the voice capacity of current generation CDMA networks.

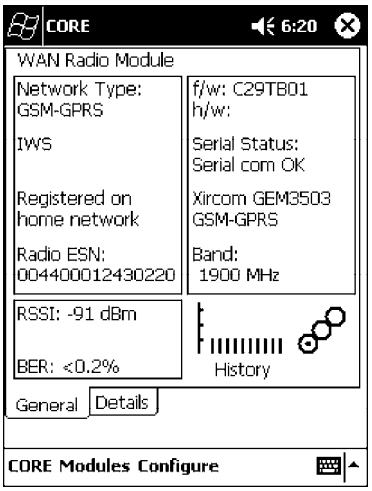
SB555 Embedded Module, from Sierra Wireless, provides complete wireless functionality and integrates easily into the most compact and slender mobile applications with its small flexible design. The SB555 offers maximum coverage and access to entire CDMA networks.

WAN Radio CORE Module

The WAN radio CORE module displays helpful information about either the GSM/GPRS radio or the CDMA/1xRTT radio option built into your 700 Series Computer. *The following illustrations are for a GSM/GPRS radio.*

General

Below are descriptions and meanings for each piece of information provided via the **General** tab, reading from top to bottom starting on the left. *The information applies to both the GSM/GPRS and the CDMA/1xRTT radio modules unless otherwise indicated.*



Network Type:
This is the network type which would list either "GSM-GPRS" or "CDMA-1XRTT." *Scatternets are not supported.*

IWS *(GSM/GPRS)* or **Sprint PCS Network** *(CDMA/1xRTT)*:
This lists the name of the service providing the network support. "IWS" is short for Iowa Wireless Service.

Registered on home network:
If the WAN radio module has registered with a service provider network, then one of the following will appear:

Registered on home network:
The radio module is registered on its "home" network.

Roaming:
The radio module is registered on another service provider's network.

Radio Not Registered:
There is no network within range of this radio module.

Radio ESN:
This displays the Electronic Serial Number (ESN) assigned to this radio module or lists "Unavailable" if a number cannot be read from the radio.

RSSI:
This displays the Received Signal Strength Indicator (RSSI) frequency or lists "Unavailable" if there is no signal or the signal cannot be retrieved from the radio module.

BER:
This shows the Bit Error Rate (BER), the percentage of bits with errors divided by the total number of bits transmitted, received, or processed over a given period of time.

f/w:
This identifies the firmware version, if available.

h/w:
This identifies the hardware version, if available.

Serial Status:
This indicates whether serial communications passed ("Serial com OK") or failed ("Serial com FAIL") in its last transaction. A status of "Serial com FAIL" typically indicates that the 700 Series Computer is unable to establish communication with the radio module installed within.

Xircom GEM3503 *(GSM/GPRS)* or **Sierra Wireless SB555** *(CDMA/1xRTT)*:
This identifies the product name for this radio module.

Band *(GSM/GPRS)* or **Channel** *(CDMA/1xRTT)*:
This identifies the bandwidth or channel available for this radio module, if any.

History:
This bar graph displays an active history of this radio module's quality of connections.

Friendly Indicator:
Usually indicates the signal strength for this radio module. Three filled dots indicate a high quality or strong signal. Three empty dots indicate that the signal is out of range or there is no signal detected.

Details

Below are descriptions and meanings for each piece of information provided via the **Details** tab, reading from top to bottom. Most of this is similar to what is shown under the **General** tab. *The information applies to both the GSM/GPRS and the CDMA/1xRTT radio modules unless otherwise indicated.*

Serial Status:
This indicates whether serial communications passed ("PASS") or failed ("FAIL") in its last transaction. A status of "FAIL" typically indicates that the 700 Series Computer is unable to establish communication with the radio module installed within.

Manufacturer:
This lists the name of the manufacturer that developed this radio module, such as "Xircom," "Siemens," or "Sierra Wireless."

Model:
This identifies the product name for this radio module, such as "SB555" or "GM350X."

IMEI # *(GSM/GPRS)*:
This is the IMEI (International Mobile station Equipment Identity) serial number of the GSM/GPRS radio module.

RSSI:
This displays the RSSI frequency or lists "Unavailable" if there is no signal or the signal cannot be retrieved from the radio module.

Operator:
This lists the name of the service providing the network support.

Band *(CDMA/1xRTT)*:
This identifies the frequency bands used by this radio module.

IMEI # *(GSM/GPRS)*:
This is the IMEI (International Mobile station Equipment Identity) serial number of the GSM/GPRS radio module.

Radio Temp *(CDMA/1xRTT)*:
This identifies the temperature of the radio module, or lists "Unavailable degrees" if there is no information or the temperature cannot be measured.

Firmware Rev:
This identifies the firmware version, if available.

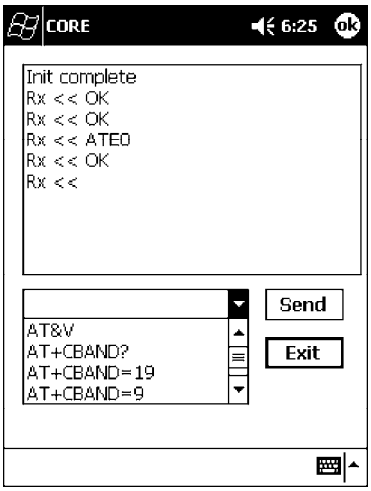
Firmware Date *(GSM/GPRS)*:
This provides the last date when this firmware was updated, if available.

IMSI # *(GSM/GPRS)*:
This shows the IMSI (International Mobile Subscriber Identity) number assigned to the SIM card installed in this 700 Series Computer.

Terminal Application

Tap **Terminal App** from the **Details** page to send standard AT commands. Information about these AT commands are available under "*AT Command Interface*" on page 4-30.

Select an AT command from the drop-down list, then tap **Send**. The results of each test appears in the text box. Tap **Exit** or **ok** to close this screen and return to the **Details** page.

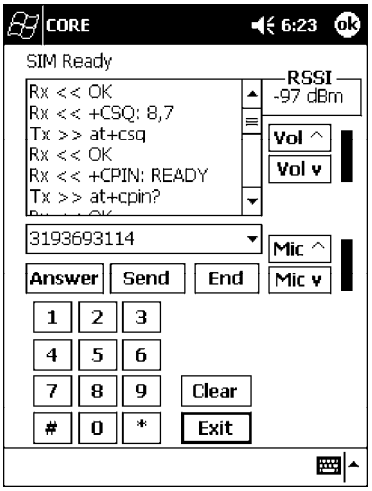


Phone Application

With the WAN radio module installed in your 700 Series Computer, you can send and receive telephone calls. Use the speaker on the back of the computer as your earpiece and use the connector on the bottom of the computer for your mouthpiece.

Tap **Phone App** from the **Details** page to access the application which will process your phone calls. Tap **Exit** or **ok** to close this application and return to the **Details** page.

- Tap the numbers for a phone call, using **Clear** to erase each digit, then tap **Send** to initiate the call.
- Tap **Answer** to receive an incoming call.
- Tap **End** to disconnect a transaction.
- Tap **Vol ^** or **Vol v** to adjust the speaker volume.
- Tap **Mic ^** or **Mic v** to adjust the microphone sensitivity.



AT Command Interface

This interface specification is based on the following recommendation:

ETSI GSM 07.05:
European Digital Cellular Telecommunication System (phase 2)
Use of DTE-DCE interface for Short message and cell broadcast service.

ETSI GSM 07.07:
European Digital Cellular Telecommunication System (phase 2)
AT command set for GSM Mobile Equipment.

ITU-T Recommendation V.25 ter
Serial asynchronous automatic dialing and control.

Command Set for CDMA/1xRTT SB555

Use the AT command interface from Sierra Wireless to program the CDMA/1xRTT SB555. Documentation for this interface is available via the following URL. Click the "General AT command reference" link for a PDF document, which is 680 KB in size. *Note that this URL is subject to change.*

http://www.sierrawireless.com/ProductsOrdering/embedded_docs.html

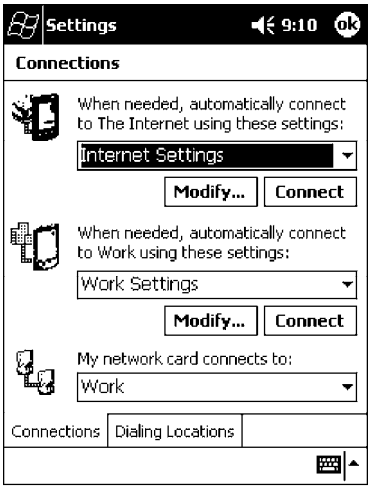
▶ **NOTE:** *You will need the Adobe Acrobat Reader application to view this document. Go to "http://www.adobe.com/prodindex/acrobat/readstep.html" to install or download the latest Adobe Acrobat Reader according to Adobe's instructions.*

Testing the AT Commands

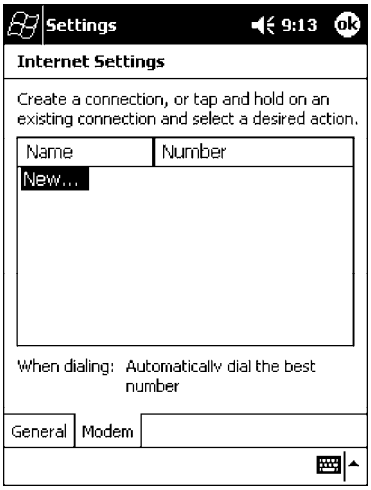
These commands can be sent to either WAN radio by setting up a dial-up networking connection to COM4. Do the following to initiate this connection and test these commands to your radio:

Connections

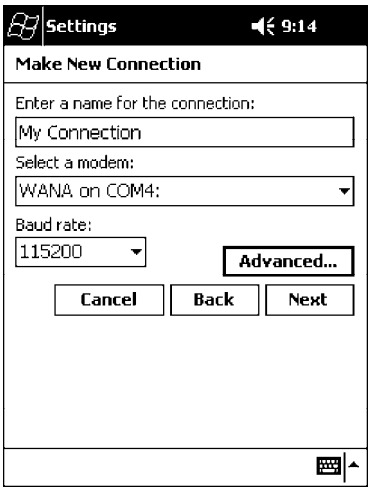
1. From the 700 Series Computer, select **Start** → **Settings** → the **Connections** tab → the **Connections** icon.
2. Tap **Modify** beneath the **Internet Settings** drop-down list.



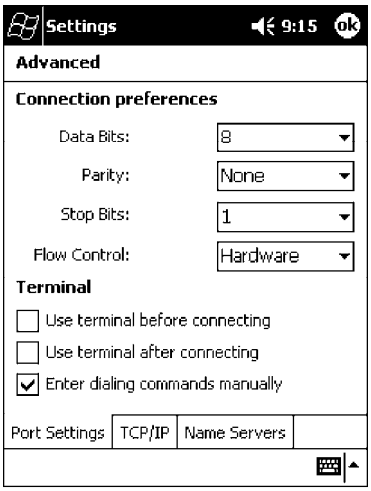
3. Tap **New..** to make a new connection.



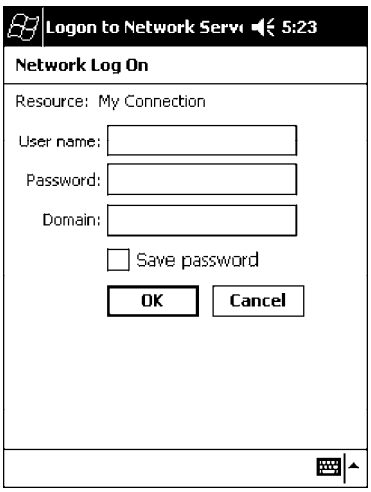
4. Enter a name for the connection, select "WANA on COM4" from the **Select a modem** drop-down list, and select "115200" from the **Baud rate** drop-down list. Tap **Advanced** to continue.



5. On the **Port Settings** tab, check **Enter dialing commands manually**, then tap **ok**, **Next**, then **Finish** to return to the Internet Settings screen with your new connection.



6. Press and hold the new connection for a pop-up menu, then tap **Connect** to initiate the connection. Wait for about ten seconds for the Network Log On screen, then tap **OK**. *Note: You do not need to enter any information within the Network Log On screen.*



7. Use either the onscreen keyboard, or press the keys to type any of the AT commands provided by Sierra Wireless. Press or tap **Enter** to send each command. The results of each command sent will print onscreen — see the sample illustration below. *Note that each "AT" command **must** start with either the "at" or "at+" characters.*
   - To see what you typed onscreen, type "ate1" to initiate the AT Echo command, then press **Enter**.

Wireless Printing

The 700 Series Computer can integrate the Wireless Printing option *(which is equipped with a Bluetooth qualified module by Socket Communications)* along with either the GSM/GPRS or the CDMA/1xRTT radio and the 802.11b radio. This option uses the network to print information stored on the 700 Series Computer.

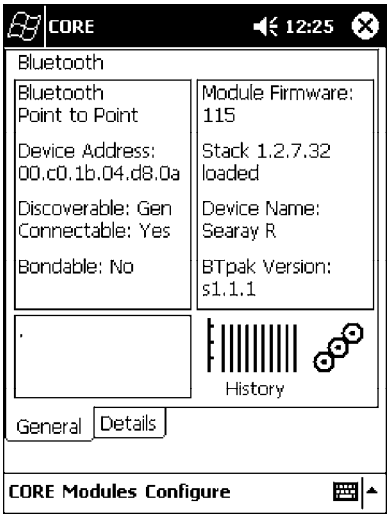
"Bluetooth" is the name given to a technology standard using short-range radio links, intended to replace the cables connecting portable and fixed electronic devices. The standard defines a uniform structure for a wide range of devices to communicate with each other, with minimal user effort. Its key features are robustness, low complexity, low power, and low cost. The technology also offers wireless access to LANs, the mobile phone network, and the internet for a host of home appliances and portable hand-held interfaces.

Documentation

Information about additional "Bluetooth" software, including the Bluetooth Device Manager and the BTctrl program, can be found within the Wireless Printing SDK. This is located on the 700C Software Tools CD, via the directory off the root of the toolkit called "*Wireless Printing SDK*." It also can be found in the *Wireless Printing Development Guide*, also on the 700C Tools CD.

Bluealps CORE Module

The Bluealps CORE module displays helpful information about this Wireless Printing option within your 700 Series Computer. Below are descriptions and meanings for each piece of information provided via the **General** tab, reading from top to bottom starting on the left.



Bluetooth Point to Point:
This is the network type. "Point to Point" is the type of connection supported as of this publication. *Scatternets are not supported.* The only supported application is wireless printing to Intermec wireless printers, such as the 781T Belt-Mount Printer.

Device Address:
This provides the network address, which in this case, will be replaced by the device address of the Bluetooth compatible module within your 700 Series Computer. *Note that this address is universally unique.*

Discoverable:
The following is displayed depending on whether the 700 Series Computer is configured to be discoverable:

"Gen" Generally discoverable
"Lim" Limited discovery
"No" Not discoverable

Connectable:
This defines whether the 700 Series Computer is able to accept other devices with Bluetooth compatible modules connecting to it. "Yes" if the connection is doable, "No" if not.

Bondable:
This defines the security element of the 700 Series Computer, which is the bondable setting. If the unit is bondable, then "Yes" is displayed, otherwise "No" is displayed.

Module Firmware:
This reflects the firmware (hardware) version of the 700 Series Computer. When the CORE module first installs onto the 700 Series Computer, the firmware level is unknown, thus "...reading" is displayed. Once the firmware level is read from the unit, then a three-digit decimal is displayed.

Stack *[Stack Version]* ***[loaded/not loaded]***:
[Stack Version] displays the Bluetooth stack version, which appears in the "1.2.3.4" format. If the stack is loaded, then "loaded" is displayed after the stack version, otherwise "not loaded" is displayed.

Device Name:
This displays the device name as assigned to the Bluetooth compatible module by the end-user. If the configured name is longer than the space allowed, it will be truncated.

BTpak Version:
This displays the driver version of additional Bluetooth components within the 700 Series Computer and is usually presented in the "1.2.3" format. The version may also contain a letter at either end of the version number.

History:
This bar graph displays an active history of this wireless printer driver's quality of connections.

Friendly Indicator:
If the Bluetooth stack is loaded, then all three dots are filled. These dots are left empty if the stack is not loaded. These dots do vary based on the CORE application's perception of the overall connection quality.

AutoIP/DHCP

Automatic Private IP Addressing (AutoIP) is enabled by default in Pocket PC 2002. To remain compatible with other Pocket PC devices, this setting needs to be enabled. You can configure the registry settings in the following to set the required AutoIP/DHCP behavior:

For Ethernet:

```
HKEY_LOCAL_MACHINE\Comm\LAN9001\TcpIp
```

Fot 802.11b:

```
HKEY_LOCAL_MACHINE\Comm\NETWLAN1\TcpIp
```

Other registry keys that can modify the behavior of AutoIP are as follows. You can find the appropriate settings and behavior of each of these keys in Microsoft Help.

- AutoInterval
- AutoMask
- AutoSubnet
- AutoIP
- AutoSeed

When a TCP/IP client cannot find a DHCP server, it generates an AutoIP address from the 169.254.xxx.xxx block. The client then tries to check for a DHCP server every 300 seconds (5 minutes) and if a DHCP server is found, the client drops the AutoIP address and uses the address from the DHCP server.

In the MSDN Windows CE documentation, see "*Automatic Client Configuration*" for more information on AutoIP.

To disable AutoIP, set the AutoCfg registry entry to "0." If a DHCP server cannot be found, instead of using AutoIP, the system will display the "Unable to obtain a server assigned IP address" message.

▶ NOTE: *If AutoIP is defined using CAB files, the EnableDHCP registry key must also be defined and set to "1" before the system will attempt to obtain a DHCP address.*

To extend the number of attempts that a DHCP client makes to get a DHCP address, use the DhcpRetryDialogue and DhcpMaxRetry registry settings.

Change the AutoInterval registry key value to make the client retry more often to obtain a DHCP address.

SNMP Configuration

Simple Network Management Protocol (SNMP) was developed in the late 1980s to provide a general-purpose internetworking management protocol. Its primary goal was to be simple so nothing would stand in the way of its ubiquitous deployment. To this end, it has been very successful as it is currently deployed in almost every major internetworking product on the market. However, like many achieved goals, the primary strength can also become a weakness.

The Focus was "Simple"

An extreme example of simplicity versus power can be realized by comparing SNMP against the Common Management Information Protocol (CMIP), the ISO entry to the standard management protocol world. CMIP has a very rich set of primitives and a core set of data elements. However, to implement CMIP, a subset of the protocol must be selected. Then, to achieve interoperability, this subset must be agreed upon with other implementors. As SNMP was specified completely and with no options, one implemented what was there and interoperability was assured.

Returning to simplicity, SNMP was built simply for a number of reasons other than time to market: robustness in the face of network failure, low overhead in the devices running the protocol; and ease of debugging the protocol itself *(the last thing you want to debug is the management protocol that is supposed to be helping you debug your network)*. Thus, the SNMP limited itself to the User Datagram Protocol (UDP). This gave the implementor the ability and responsibility to manage lost packets and perform any necessary retransmissions. As network debugging in the face of changing routes will certainly mean losing packets, retaining this control from the transport service *(layer 4)* was considered essential. Since a network management protocol will run continuously, it is mandatory that it consume as minimal a network resource as possible. UDP allows the necessary control over packet transmissions, packet size and content *(packetization)*. It is a natural choice.

Using SNMP

SNMP has three control primitives that initiate data flow from the requester *(get, get-next, and set)*. There are two control primitives the responder uses to reply. One is used in response to the requester's direct query *(get-response)* and the other is an asynchronous response to obtain the requester's attention *(trap)*. All five of these primitives are carried by UDP and are thus limited in size by the amount of data that can fit into a single UDP packet. The relatively small message size was a goal of the design but for some reasonable set of network management functions, it imposes a limitation.

Often in network management, it is necessary to obtain bulk information without knowing at first what is in that bulk. In one case, there is a set of problems having to do with packets not going where they are supposed to, due to device misconfiguration that prevents proper protocol operation where one needs to view the entire set of data.

Retrieval of Management Information

SNMP has the get-next primitive which permits the viewing of data without requiring prior knowledge. If you know what you are looking for, the get primitive will return it. When you want an entire table of information, the get-next primitive will obtain it. However, unless employed with care, the get-next primitive can be extremely resource-intensive in real time, network bandwidth, and the agent's CPU time. The simplest use of the get-next primitive is to start at the beginning of a table, await the response and then issue another get-next with the name returned. As an example, say you wanted the next-hop address, next-hop interface, and route-type from a routing table containing 1000 entries. Using the simplest form of get-next, this would require 2x3x1000 or 6000 packets *(get-next and get-response packets, columns, and rows)*. A straight-forward optimization would be to request the three columns in a single packet. This puts the number of packets at 2x1000 or 2000 packets. In real time, it is the product of the round trip by the number of request. In agent CPU time, this is still 6000 lookups in the routing table for both cases.

An Early Approach to Getting More than One Item at a Time

The ability to retrieve only one piece or object at a time has been a problem for SNMP. This is particularly an issue with the use of this protocol in wireless environments where the wireless datapipe is small and overhead due to network management it is considered overhead. One approach creates multiple get-next requests running concurrently. A second algorithm, reduces the packet count by combining the multiple concurrent get-nexts into a single packet. Neither approach has been implemented which makes network management in wireless environment, though essential to the success of the operation, tenuous. The issue has been resolved in SNMP V2 protocol where a get-bulk primitive has been defined.

Conclusion

Software development moves forward by evolving the unknown into the known and wireless environments are moving from vertical only application to wide spread implementation. At the time of the SNMP inception, it was not possible to conceive of a reliable transport based network management protocol. Today's problems require more sophisticated data to analyze a problem. This puts the burden back on the protocol to send and receive data quickly and efficiently. Work continues in subcommittees to improve SNMP and resolve the issues that are developing with new applications and new network architectures.

SNMP Configuration on the 700 Series Computer

In short, SNMP is an application-layer protocol that facilitates the exchange of management information between network devices. The 700 Series Computer is such an SNMP-enabled device. Use SNMP to control and configure the 700 Series Computer anywhere on an SNMP-enabled network.

The 700 Series Computer supports four proprietary Management Information Bases (MIBs) and Intermec Technologies provides SNMP support for MIB-II through seven read-only MIB-II (RFC1213–MIB) Object Identifiers (OIDs).

▶ **NOTE:** *You can only query these seven OIDs through an SNMP management station, these are not available in the Unit Manager applications.*

Management Information Base

The Management Information Base is a database that contains information about the elements to be managed. The information identifies the management element and specifies its type and access mode (Read-Only, Read-Write). MIBs are written in ASN.1 (Abstract Syntax Notation.1) – a machine independent data definition language. *Note: Elements to be managed are represented by objects. The MIB is a structured collection of such objects.*

You will find the following MIB files either on the 700C Tools CD or on the web via *http://www.intermec.com*:

INTERMEC.MIB
Defines the root of the Intermec MIB tree.

ITCADC.MIB
Defines objects for Automated Data Collection (ADC), such as bar code symbologies.

ITCSNMP.MIB
Defines objects for Intermec SNMP parameters and security methods, such as an SNMP security IP address.

ITCTERMINAL.MIB
Defines objects for 700 Series parameters, such as key clicks.

Object Identifiers

Each object has a unique identifier called an OID. OIDs consist of a sequence of integer values represented in dot notation. Objects are stored in a tree structure. OIDs are assigned based on the position of the object in the tree. Seven MIB OIDs are shown in Table 4-1 below and on the next page.

EXAMPLE: The internet OID = 1.3.6.1.

Table 4-1
MIB Object Identifiers

MIB-II Item	OID	Group or Table	Description
ifNumber	1.3.6.1.2.1.2.1.0	Interfaces Group	Indicates the number of adapters present in the system. For the 700 Series Computer, if one adapter is present in the system, then *ifNumber* = 1 and ifIndex = 1.
ifIndex	1.3.6.1.2.1.2.2.1.1.ifIndex	Interfaces Table (ifTable)	A unique value for each interface. The value ranges between 1 and the value of *ifNumber*.
ifDescr	1.3.6.1.2.1.2.2.1.2.ifIndex	Interfaces Table (ifTable)	A textual string containing information about the interface.
ifType	1.3.6.1.2.1.2.2.1.3.ifIndex	Interfaces Table (ifTable)	An integer containing information about the type of the interface. It is equal to 1 for Other.
ipAdEntAddr	1.3.6.1.2.1.4.20.1.1.IpAddress	IP address Table (ipAddrTable)	The IP address to which this entry's addressing information pertains *(same as 700 IP address)*, where IP Address is the valid non-zero IP address of the 700 Series Computer.

Table 4-1 (Continued)
MIB Object Identifiers

MIB-II Item	OID	Group or Table	Description
ipAdEntIfIndex	1.3.6.1.2.1.4.20.1.2.IpAddress	IP address Table (ipAddrTable)	The index value that uniquely identifies the interface to which this entry is applicable *(same as ifIndex)*.
ipAdEntNetMask	1.3.6.1.2.1.4.20.1.3.IpAddress	IP address Table (ipAddrTable)	The subnet mask associated with the IP address of this entry *(same as Subnet Mask)*.

Configuring with SNMP

The community string allows an SNMP manager to manage the 700 Series Computer with a specified privilege level. The default read-only community string is "public" and "private" is the default read/write community string. See the specific configuration parameter to find its OID.

To configure the 700 Series Computers using SNMP:

1. Configure your 700 Series Computers for RF or Ethernet communications.
2. Determine the OID (Object Identifier) for the parameter to be changed. The Intermec base OID is 1.3.6.1.4.1.1963.
3. Use your SNMP management station to get and set variables that are defined in the Intermec MIBs. You can set the traps, identification, or security configuration parameters for SNMP. *See Appendix A, "Control Panel Applets," to learn more about these parameters.*

Network Selection APIs

The Network Selection APIs allow the user to change 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 FWV1: 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(“FWV1:”) };
   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(“FWV1:”) };
   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);
      }
   }
}
```

Section 5

Printer Support

The 700 Series Color Mobile Computer works with the following printers from Intermec Technologies. Contact an Intermec Representative for information about these printers.

6820 A full-page, 80-column printer.

6808 A 4-inch belt-mount printer.

781T A 2-inch belt-mount printer with a Bluetooth compatible module from Socket Communications.

782T A 2-inch workboard printer.

Printing ASCII

The following methods for printing using Pocket PC at this time is as follows:

- Add port drivers to print ASCII directly to the port.
- Use LinePrinter ActiveX Control from the Software Developer's Kit (SDK) — *see the SDK User's Manual for more information.*
- Via wireless printing — *see the Wireless Printing Development Guide on the 700C Software Tools CD for more information.*

Directly to a Port

Printing directly to the port sends RAW data to the printer. The format of this data depends upon your application and the printer capabilities.

You must understand the printer commands available for your specific printer. Generally, applications just send raw ASCII text to the printer. Since you are sending data to the printer from your application directly to the port you are in complete control of the printers operations. This allows you to do line printing *(print one line at a time)* rather than the page format printing offered by the GDI approach. It is also much faster since data does not have to be converted from one graphics format to the other (display to printer). Most Intermec® printers use Epson Escape Sequences to control print format operations.

These commands are available in documentation you receive with your printers or from technical support. Win32 APIs are required to print directly to the port.

Directly to a Generic Serial Port

To print directly to a generic serial port printer (non-Intermec printers):

- Use CreateFile() to open ports — COM1: can be opened on most devices.
- Use WriteFile() to send data directly to the printer.
- Use CloseHandle() when you are finished printing to close the port.

IrDA Printer Driver

IrDA printing is only available on the certain devices and is supported directly by the Windows CE load via the IrSock API provided by the Microsoft Win32 API without need for additional drivers. Intermec 6804, 6805, 6806, 6808 and 6820 and other IrDA printers are supported.

NPCP Printer Driver

The NPCP printer communications driver (NPCPPORT.DLL) is a Stream Device Driver built into the operating system. The driver supports only NPCP communications to and from the 6820 and 4820 printers over a selected serial port.

All applications use WIN32 API functions to access the drivers. Basic operations are easily implemented by applications through the CreateFile(), WriteFile(), ReadFile(), DeviceIOControl(), and CloseHandle() Win32 APIs.

Operations to upgrade printer modules, perform printer diagnostics, and get printer configuration are performed largely via DeviceIOControl() functions.

About NPCP

NPCP (Norand® Portable Communications Protocol) is a proprietary protocol that provides session, network, and datalink services for Intermec mobile computers in the Intermec LAN environment used with printers and data communications.

NPCP Driver Installation and Removal

Use LPT9: for the NPCP printer device and COM1 for the last parameter. COM1 is the connection available via the 700 Series Mobile Computer.

Applications use the RegisterDevice() function to install the driver. DeregisterDevice() uninstalls the device driver and frees memory space when the driver is not required. Use the HANDLE returned by RegisterDevice() as the parameter to DeregisterDevice().

Use the RegisterDevice() function call as demonstrated below. Specify the full path name to the driver starting at the root for the RegisterDevice() function to work properly. The last parameter to RegisterDevice() is a DWORD that represents the name of the port for the NPCP stream driver to use. Build this parameter on the stack if it is not to be paged out during the call. The first parameter "LPT" (Device Name) and the second parameter '9' (index), indicate the name of the registered device, such as LPT9. This is used in the CreateFile() function call.

```
Install()
{
   HANDLE hDevice;
   TCHAR port[6];
   port[0] = TCHAR('C');
   port[1] = TCHAR('O');
   port[2] = TCHAR('M');
   port[3] = TCHAR('1');
   port[4] = TCHAR(':');
   port[5] = TCHAR(0);
   hDevice = RegisterDevice ( (TEXT("LPT"), 9,
   TEXT("\\STORAGE CARD\\WINDOWS\\NPCPPORT.dll"), (DWORD)port);
}
```

Opening the NPCP Driver

The application opens the NPCP driver by using the CreateFile() function. The call can be implemented as follows. The first parameter "LPT9:" must reflect the device name and index used in the RegisterDevice() function call and will fail for any of the following reasons:

```
hFile = CreateFile(_T("LPT9:"), GENERIC_WRITE | GENERIC_READ, 0, NULL, OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);
```

- The port associated with the device during RegisterDevice() is in use.
- The NPCP device is already open.
- The share mode is not set to zero. The device cannot be shared.
- Access permissions are not set to GENERIC_WRITE | GENERIC_READ. Both modes must be specified.

Closing the NPCP Driver

Using the CloseHandle() (hFile) function closes the NPCP driver. Where *hFile* is the handle returned by the CreateFile() function call.

- TRUE = the device is successfully closed.
- FALSE = an attempt to close NULL HANDLE or an already closed device.

Reading from the NPCP Driver

Reading of the NPCP printers is not supported since all responses from the printer are the result of commands sent to the printer. DeviceIoControl() functions are provided where data is to be received from the printer.

Writing to the NPCP Driver

All Print data can be sent to the printer using the WriteFile() function. The print data written to the driver must contain the proper printer commands for formatting. If the function returns FALSE, the NPCP error may be retrieved using IOCTL_NPCP_ERROR. See the description on the next page.

NPCP Driver I/O Controls

An application uses the DeviceIoControl() function to specify an printer operation to be performed. Certain I/O controls are required to bind and close communication sessions with the printer, and must be completed before any other commands to the driver can execute properly.

The function returns TRUE to indicate the device successfully completed its specified I/O control operation, otherwise it returns FALSE. The following I/O control codes are defined:

```
#define IOCTL_NPCP_CANCEL
CTL_CODE(FILE_DEVICE_SERIAL_PORT,0x400,METHOD_BUFFERED,FILE_ANY_ACCESS)

#define IOCTL_NPCP_BIND
CTL_CODE(FILE_DEVICE_SERIAL_PORT,0x401,METHOD_BUFFERED,FILE_ANY_ACCESS)

#define IOCTL_NPCP_CLOSE
CTL_CODE(FILE_DEVICE_SERIAL_PORT,0x402,METHOD_BUFFERED,FILE_ANY_ACCESS)

#define IOCTL_NPCP_ERROR
CTL_CODE(FILE_DEVICE_SERIAL_PORT,0x403,METHOD_BUFFERED,FILE_ANY_ACCESS)

#define IOCTL_NPCP_FLUSH
CTL_CODE(FILE_DEVICE_SERIAL_PORT,0x404,METHOD_BUFFERED,FILE_ANY_ACCESS)

#define IOCTL_NPCP_IOCTL
CTL_CODE(FILE_DEVICE_SERIAL_PORT,0x405,METHOD_BUFFERED,FILE_ANY_ACCESS)

#define IOCTL_NPCP_PRTVER
CTL_CODE(FILE_DEVICE_SERIAL_PORT,0x406,METHOD_BUFFERED,FILE_ANY_ACCESS)
```

IOCTL_NPCP_CANCEL
This command cancels all printing at the printer. It flushes the printer buffers and reinitializes the printer to its default state. No parameters are required.

IOCTL_NPCP_BIND
This command is required before any data is sent or received by the printer. Once the driver is opened, the application must bind the communications session with the printer before any data can be sent or received by the printer. If an error occurs during the bind, the application may use IOCTL_NPCP_ERROR to get the current extended error code. No parameters are required.

IOCTL_NPCP_CLOSE
This command closes the current session with the printer. This function always returns TRUE. No parameters are required.

IOCTL_NPCP_ERROR
This command returns the extended NPCP error code in PL/N format. The word returned will contain the PL/N compatible error code in the low byte and completion flags in the high byte. If the frame that returned an error was not received correctly by the printer the FRAME_NOT_ACKED bit will be set in the high byte. This operation always returns TRUE. An output buffer of at least 2 bytes is required. See "*NPCP Error Codes*" on page 5-5.

IOCTL_NPCP_FLUSH
This command allows the application to poll the printer for errors while the report is completing the print process at the printer. If an error occurs during the polling process, the operation will return FALSE and the application can get the extended error code by using IOCTL_NPCP_ERROR. No parameters are required.

NPCP Printer Communications

All NPCP printer communications should be based on the following flow:

1. Use CreateFile(); to open the printer driver.
2. Use IOCTL_NPCP_BIND to bind a session with the printer; IOCTL_NPCP_ERROR to check for errors on the bind to ensure success; and IOCTL_NPCP_CANCEL to cancel any outstanding print jobs.
3. Use IOCTL_NPCP_FLUSH to poll the printer to free up printer buffer resources. Use IOCTL_NPCP_FLUSH to poll the printer's status. If an error is reported by the IOCTL, then use IOCTL_NPCP_ERROR to get the error and determine the correct recovery procedure.
4. Use WriteFile(); to write your data to the printer. Check for errors and that all data were written. Use IOCTL_NPCP_ERROR to get the extended error. If the error is critical in nature, use IOCTL_NPCP_CLOSE, followed by CloseFile(), to end the communications session. Start a new session, beginning with step 1 to ensure proper printing. For noncritical errors display the error and retry the operation.
5. After all data is sent to the printer, ensure that the printer continues to print the report properly by polling the printer's status. Use IOCTL_NPCP_FLUSH to poll the printer's status. If an error is reported by the IOCTL, then use IOCTL_NPCP_ERROR to get the error and determine the correct recovery procedure.

Sample Code

See sample code in the "\700 Color Dev Tools\Installable Drivers\Port Drivers\Npcp\NPCPPrint\" directory for more details on printing, printer communications and error code handling.

NPCP Error Codes

Call the IOCTL_NPCP_ERROR I/O control function to receive PL/N compatible error codes. Applications must decide how to act upon the data returned.

```
// Definition of NPCP communications Errors and Printer Errors
#define PNRDY (BYTE)102 // link not ready error
#define RXTMO (BYTE)104 // link no receive error
#define TXTMO (BYTE)106 // link no transmit error
#define BADADR (BYTE)111 // frame address error
#define GAPERR (BYTE)112 // link gap error (timeout) in receive data
#define LSRPE (BYTE)113 // frame parity error on length field
#define IFTS (BYTE)120 // session layer - invalid frame this state
#define NS_NE_VR (BYTE)121 // session layer sequence error
#define NR_NE_VS (BYTE)122 // session layer sequence error
#define MAC_CRCERR (BYTE)124 // MAC CRC error
#define RLENERR (BYTE)123 // MAC too much data received
#define FRMERR (BYTE)200 // Frame Reject
#define FRMERR_IF (BYTE)201 // Frame Reject - Invalid Frame
#define FRMERR_NR (BYTE)202 // Frame Reject - NR Mismatch
#define FRMERR_NS (BYTE)203 // Frame Reject - NS Mismatch
#define NDMERR (BYTE)204 // Normal Disconnect mode error
#define BINDERR (BYTE)210 // bind error
#define IPLDUR (BYTE)221 // invalid presentation layer response
#define HEADJAM (BYTE)222 // printer head jam
#define PAPEROUT (BYTE)223 // printer paper out
#define LOWVOLTS (BYTE)224 // printer low voltage
#define HIVOLTS (BYTE)225 // printer over voltage
#define LOWBAT (BYTE)226 // printer low battery
#define COVEROFF (BYTE)227 // printer cover off error
#define HEADFAULT (BYTE)228 // printer head short or driver short error
#define PFFAULT (BYTE)229 // paper feed motor fault.
#define FRAME_NOT_ACKED 0x8000 // frame was not received by printer and need to be resent.
```

O'Neil Printer Driver

The DTR printer communications driver is a Stream Device Driver named ONEIL.DLL.

All applications use WIN32 API functions to access drivers. Basic operations are easily implemented by applications through the CreateFile(), WriteFile(), DeviceIOControl() and CloseHandle() Win32 APIs.

The driver supports communications to 6804DM, 6804T, 6805A, 6806, 6808, 681T, and 781 printers over a selected serial port.

DTR Driver Installation and Removal

Your application must install the device driver by using the RegisterDevice() function. The driver name is ONEIL.DLL. We recommend that you use "DTR" for the Device Name parameter, "1" for the Device Driver index parameter, and use any of the following strings for the last parameter:

- NULL (==0) Defaults to COM1 @ 9600
- "COM1" only COM port specified defaults to 9600
- "COM1:9600" sets to COM port and specified bit rate
- "COM1:19200" sets to COM port and specified bit rate

Use the HANDLE returned by RegisterDevice() as the parameter to DeregisterDevice(). The correct usage of the RegisterDevice() function call is demonstrated below. You may use DeregisterDevice() to uninstall the driver.

```
Install()
{

   HANDLE hDevice;
   TCHAR port[6];
   port[0] = TCHAR('C');
   port[1] = TCHAR('O');
   port[2] = TCHAR('M');
   port[3] = TCHAR('1');
   port[4] = TCHAR(':');
   port[5] = TCHAR(0);
   hDevice = RegisterDevice ( (TEXT("DTR"), 1, TEXT("\\WINDOWS\\ONEIL.DLL"),
   (DWORD)port);

}
```

Opening the DTR Driver

The application opens the DTR driver by using the CreateFile() function. The call can be implemented as follows:

```
hFile = CreateFile(_T("DTR1:"), GENERIC_WRITE, 0, NULL, OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);
```

The first parameter "DTR1:" must reflect the device name and index used in the RegisterDevice() function call.

The function call will fail for any of the following reasons:

- The port associated with the device during RegisterDevice() is currently in use.
- The DTR device is already open.
- The share mode is not set to zero. The device cannot be shared.
- Access permissions are not set to GENERIC_WRITE.

Closing the DTR Driver

Using the CloseHandle() (hFile) function closes the DTR driver. Where *hFile* is the handle returned by the CreateFile() function call.

- TRUE indicates the device is successfully closed.
- FALSE indicates an attempt to close a NULL HANDLE or an already closed device.

Writing to the DTR Driver

You can use the WriteFile() function to send all Print data to the printer. The print data being written must contain the proper formatting printer commands.

DTR Printer Communications

All DTR printer communications should be based on the following flow:

1. Use CreateFile(); to open the printer driver.
2. Use WriteFile() to write your data to the printer. Check for errors and that all data were written.
3. Use CloseHandle() to close the driver.

Section 6

Scanner Support

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

APS linear imager:
Reads 1D symbologies and PDF 417 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:
This decodes several stacked 1D and 2D symbologies, including PDF 417 and Datamatrix without "painting." It can also read 1D codes from any orientation, for example the scan beam does not need to be aligned perpendicular to the symbol in order to read it. Photography is a secondary application; the lens in the device will favor bar code reading. Photos are 640x480, 256 gray-scale.

1D laser scanner:
Traditional laser scanner that decodes 1D bar codes.

PDF 417 laser scanner:
Higher speed laser scanner that can read PDF 417 labels by "painting" the label.

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. See Section 3, "Installing Applications and Updating System Software" for more information.*

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. For more information, see the *SDK User's Manual*.

ITCAxReaderCommand functions:
Use these ActiveX controls to modify and retrieve configuration information using the reader interface commands. For more information, see the *SDK User's Manual*.

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.

Automatic Data Collection COM Interfaces

Data collection configuration and functionality cannot be accessed by any means *(including control panel applets or remote management applications)* until after the 700 Series Computer has completed initialization, which occurs during a warm- or cold-boot or after a firmware upgrade.

When initialization is complete, the green LED on the 700 Series Computer stops flashing. Changes made to configuration settings remain after a warm boot. After a cold-boot, all configuration settings are reset to their defaults with the exception of scanner configurations, which remain except for the Symbology Identifier transmission option or the Preamble and Postamble strings. To reset all configuration settings to the factory defaults, the S9C scanner firmware must be reloaded.

The Automatic Data Collection (ADC) functions are accessed through custom COM interfaces. These interfaces allow the application to receive bar code data and configure and control the bar code reader engine. The COM interfaces include the following functions:

- IADC *(starting on page 6-12)*
- IBarCodeReaderControl *(starting on page 6-17)*
- IS9CConfig *(starting on page 6-28)*
- IS9CConfig2 *(starting on page 6-54)*
- IS9CConfig3 *(starting on page 6-61)*
- IImage Interface *(starting on page 6-65)*

Multiple ADC COM Object Support

A 700 Series Computer may have multiple reader engines to decode different types of ADC data. For example, a bar code reader engine decodes raw bar code data and passes it to a bar code reader COM object. An RFID reader engine decodes raw RFID tag data and passes it to an RFID tag reader COM object.

ADC COM interfaces are implemented as in-process COM objects. An instance of the ADC COM object creates a logical connection to access or control the reader engine. Specifically, the IBarCodeReadConfig or IBarCodeReaderControl COM objects can manage the bar code scanner configuration while the ADC COM object can gather data simultaneously. These ADC COM objects or connections can be created in a single application or multiple applications. Up to seven instances of a COM object can be created for a reader engine. For more information, see "*How to Create and Use the ADC COM Interfaces*" below.

For data collection features, ADC COM objects also provide for read ahead and non-read ahead data access and grid data editing.

How to Create and Use the ADC COM Interfaces

You can also use the Input Device Functions *(starting on page 6-11)* to create and use the ADC COM interfaces.

1. Create and initialize the in-process Bar Code Reader object using ITCDeviceOpen() *(see page 6-11)*. This function returns a COM Interface pointer to the Bar Code Reader Object created by the function.
2. Set the data grid if data filtering is desired *(default grid gives the application all the data)*. Below is a sample code of how to set the grid to accept Code 39 data that starts with the letter "A" and is not a reader command.

```
ITC_BARCODEREADER_GRID stBCGrid;
stBCGrid.stDIGrid.szDataMask = TEXT("A%s");
stBCGrid.stDDGrid.dwSymbologyMask = BARCODE_SYMBOLOGY_CODE39;
stBCGrid.dwDataSourceTypeMask = ITC_DATASOURCE_USERINPUT;
HRESULT hrStatus = pIBCControl->SetAttribute(
ITC_RDRATTR_GRID,
reinterpret_cast<BYTE *>(&stBCGrid),
sizeof(stBCGrid)
);
```

3. Issue a read to accept the bar code data. The timestamp, symbology, and data type are put into the ITC_BARCODE_DATA_DETAILS structure. Passing in a pointer to this structure is optional. The following sample code uses an infinite timeout.

```
ITC_BARCODE_DATA_DETAILS stBCDetails;
BYTE rgbBCData[1024]; // Buffer used to accept the bar code data
DWORD dwBytesReceived; // Number of bytes in the return data.
HRESULT hrStatus = pIBCControl->Read(
rgbBCData,
sizeof(rgbBCData),
&dwBytesReceived,
& stBCDetails,
INFINITE
);
```

4. Compile and link the application.

Read-Ahead Bar Code Data Access

The Bar Code Reader COM object delivers ADC data to the connection in read-ahead mode. In this mode, the data is queued until a COM connection is ready to read it. Read-ahead mode decouples reader device performance from the application performance. That is, data is read as fast as the user can scan it, independent of the connection processing load. No data will be scanned until the first Read() function is posted.

Grid Data Filtering

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. The data can be filtered so that only data conforming to a certain text pattern or symbology will be sent to an application. After the data is filtered, it can be edited by adding, deleting, or rearranging portions of the text or by extracting portions of text for further editing. To filter and edit data, you need to define the virtual wedge grid parameters.

Grid Processing:

Grid processing takes place in two steps:

Compilation:

In which the user's grid expressions are checked for errors and reduced to a binary form for faster matching. This is done whenever the virtual wedge grid is set or changed by configuration software.

Matching:

In which data is tested against the grids set in compilation. Matching can be performed multiple times after a compilation. The AIM symbology ID of the data being tested, including the enclosing angle brackets, must be prepended to the incoming data.

Syntax:

The basic syntax of each grid expression is:

```
<symID> filter-expression= > editing-expression
```

symID

Is the AIM symbology ID *(see the AIM Symbology ID Defaults table starting on page 6-63)*.

filterexpression

Is any character string that includes valid filter expression values *(see the "Filter Expression Values" table below)*.

editing-expression

Is any character string that includes valid editing expression values *(see the "Editing Expression Values" table on page 6-6)*.

Filter Expression Values

A filter-expression can be any string of text containing the operators listed below.

Operator	Meaning	Example
Any character string not containing the special characters: . ? [ ] { } or \ *(period, question mark, left/right brackets, left/right curly brackets, backslash)*	Match the string literally.	super20 matches super20
\c where c is any of the special characters: . ? [ ] { } or \ *(period, question mark, left/right brackets, left/right curly brackets, backslash)*	Remove any special meaning of c.	* matches *
. (period)	Any character.	. matches x
^ (tarot)	Anchor the match at the beginning of the string.	^abc matches abc, but not aabc
$ (dollar sign)	Anchor the match at the end of the string.	abc$ matches abc but not abcc
? (question mark)	Repeat the preceding expression zero or one time.	aa? matches a or aa
* (asterisk)	Repeat the preceding expression zero or more times.	ab*c matches ac, abc, abbc, etc.
+ (plus symbol)	Repeat the preceding expression one or more times.	ab+c matches abc, abbc, etc.
[characterclass]	A series of non–repeating characters denoting a class.	[abcdefhijklmnopqrstuvwxyz] is the class of all lowercase alphas.
[rangel–rangeh]	A sequential range of nonrepeating characters denoting a class.	[a–z] is the class of all lowercase alphas.
[^characterclass]	Any character except those denoted by a character class.	[^a–z] matches a numeric digit or a punctuation mark.
[characterclass_tag]	[:alnum:] — Alphanumeric characters [:alpha:] — Alphabetic characters [:blank:] — Tab and space [:cntrl:] — Control characters [:digit:] — Numeric characters [:graph:] — All printable characters except space [:lower:] — Lowercase letters [:print:] — All printable characters [:punct:] — Punctuation [:space:] — White space characters [:upper:] — Uppercase letters [:xdigit:] — Hexadecimal digits	[[:alpha:]]* matches Dynaction, Selmer, or NewWonder but not Super20
{num}	Matches exactly *num* repetitions.	a{3} matches only aaa
{min,}	Matches at least *num* repetitions.	a{3,} matches aaaa but not aa
{min,max}	A repetition operator like + or *, except the number of repetitions is specified by *min* and *max*.	[a–z]{1,3} matches a, ab, or aab, but not aabc

Operator *(Continued)*	**Meaning**	**Example**
(expr1)|(expr2)	Matches *expr1* or *expr2*.	a|b matches a or b
(subexpression)	Grouping operator to consolidate terms into a subexpression, which can override the order of evaluation. The subexpression is available for later matching or editing by means of *index*, where *index* is between 1–9 and refers to the index–th group in the string, counting from left to right. *\0* refers to the whole expression.	Overriding evaluation order: (ab)*c matches c, abc, ababc, etc. Back-referencing: (aa)bb\1 matches aabbaa.

Editing Expression Values

This table lists the valid operators for editing expressions.

Operator	**Meaning**	**Example**
\index	The *index-th* subexpression (reading left-right) in the matched string. *index* must be between 0–9. *\0* is the matched expression itself.	M([0–9]{6})= > \1 produces 270494 when M270494 is scanned, stripping off the first character.
& or \0	The matched expression itself.	M[0–9]{6}= > \0–Conn and M[0–9]{6}= > &–Conn both produce M270494–Conn when M270494 is scanned.
\xhh	A concise representation of the lower 256 characters in the Unicode set. When converted, this is still a 16-bit value.	\x0d inserts a carriage return.
any character string	Inserts any character string in the output string.	See previous examples.

<*symID*> is optional. If present, only data in the indicated symbology is accepted.

If the entire expression is blank, all data is passed unchanged. If = > *editing-expression* is omitted, then all data that passes through the filter is returned unchanged. If = > *editing expression* is present, the data is transformed by editing-expression.

Multiple grid expressions can be compiled and are related in a logical OR fashion. These are expressed as single grid expressions separated by semicolons. When matching is attempted, the first grid expression from left to right that achieves a match will cause the data to be accepted.

All pattern expressions and parsed data are in Unicode.

EXAMPLE 1: **Grid Filter Example 1**

This accepts a serial number in which the encoded number is a six-character string beginning with M followed by six numeric characters.

Filter

M[0–9]{6}

Effect

When a bar code, such as M270494, is scanned, all data is passed.

EXAMPLE 2: **Grid Filter Example 2**

This formats a scanned Social Security number and forms it into an XML element tagged "SSN".

Filter

([0–9]{3})([0–9]{2})([0–9]{4})= > <SSN > \1–\2–\3</SSN >

Effect

A bar code, such as 123456789, is passed and reformatted to
<SSN > 123-45-6789</SSN >

EXAMPLE 3: **Grid Filter Example 3**

This deletes the first three and last five characters of a 21-character Code 128 label and deletes the first two characters of a 10-character Interleaved 2 of 5 label.

Filter

<]C > ...(.{13}).....= > \1; <]I > ..(.....)= > \1

Effect

If Code 128, AAA1234567890123BBBBB becomes 1234567890123
If Interleaved 2 of 5, AA12345678 becomes 12345678

EXAMPLE 4: **Grid Filter Example 4**

This inverts data such that the first alphabetic string (like a first name) and second alphabetic string (like a last name) are reversed and separated by a comma and a space.

Filter

([[:alpha:]])+ ([[:alpha:]])+= > \2, \1

Effect

When a bar code with the data "Dexter Gordon" is scanned, the data is modified to read "Gordon, Dexter".

ADC Connection

A 700 Series Computer can have both Bar Code and RFID reader engines with each engine supporting multiple connections. Each connection allows an application to access data or manage a configuration. An application could have multiple connections.

```
   // Get an instance of the ADC COM object that corresponds integrated scanner
IBarCodeReaderControl *pIBCControl;
   // Pointer to the Bar Code Reader object
HRESULT hrStatus = ITCDeviceOpen( TEXT(”default”),
IID_IBarCodeReaderControl, ITC_DHDEVFLAG_READAHEAD,
(LPVOID *) &pIBCControl);
   // If the ADC object was successfully created and initialized, accept bar code data.
ITC_BARCODE_DATA_DETAILS stBCDetails;
stBCDetails.wStructSize = sizeof(stBCDetails);
BYTE rgbBCData[1024];
   //Buffer used to accept the bar code data
DWORD dwBytesReceived;
   // Number of bytes in the return data.
HRESULT hrStatus = pIBCControl->Read(
rgbBCData,
sizeof(rgbBCData),
&dwBytesReceived,
& stBCDetails,
INFINITE
);
```

2D Imager Overview

The 700 Color optional integrated 2D Imager captures 640x480 256-grayscale images at 20 frames per second. The imager features can be categorized into data collection features and image acquisition features as follows:

Data Collection Features

The imager includes a decode engine capable of decoding 2D matrix symbologies such as Datamatrix as well as the traditional 1D and stacked symbologies *(see the table on the next page for supported symbologies)*. The application programming interfaces used to collect bar code data and configure the imager are the same as those used for the laser scanner. This includes the keyboard wedge as well as the ADC COM interfaces and includes functionality such as data editing and data filtering. In addition, the imager has the following configuration features (see "*IS9CConfig3 Functions*" starting on page 6-61 for configuration details):

Aimer LED:
A small, rectangular-aiming LED is displayed periodically during the image capture and decoding process. The initial duration *(after scan buttons are pressed)* of the aimer LED can be configured. This helps to select the specific bar code to be scanned with multiple bar codes in the image.

Scaled Illumination LED:
When the ambient light is not sufficient to decode the bar code, the red illumination LEDs will be turned on to brighten the image. The intensity of the illumination LEDs is scaled to brighten the image just enough for decode. The reduces power consumption and the effect of specular reflection.

Window size and position:
The default window size (640x480) can be reduced in size and positioned. This is useful in applications where multiple bar codes may be present in the image and the specific bar code must be selected to be read. For example, the window can be sized and positioned around the aimer LED. The entire bar code must reside in the configured window for a good decode.

Omni-directional scanning is a feature that does not require configuration. 1D and stacked symbologies as well as 2D matrix symbologies can be scanned with the 700 Series Computer in any orientation. Thus, time is not needed to orient the 700 horizontal as with laser scanners.

The following table shows which bar code symbologies are supported either by an imager or by a laser scanner.

Bar Code Symbology	Imager	Laser Scanner
Code 39	X	X
Interleaved 2 of 5	X	X
Standard 2 of 5	X	X
Matrix 2 of 5		X
Code 128	X	X
Code 93	X	X
Codabar	X	X
MSI		X
Plessey		X
UPC	X	X
EAN/EAN 128	X	X
Code 11		X
PDF 417	X	X
Micro PDF 417		X
Telepen		X
Datamatrix	X	
QR Code	X	

Image Acquisition Features

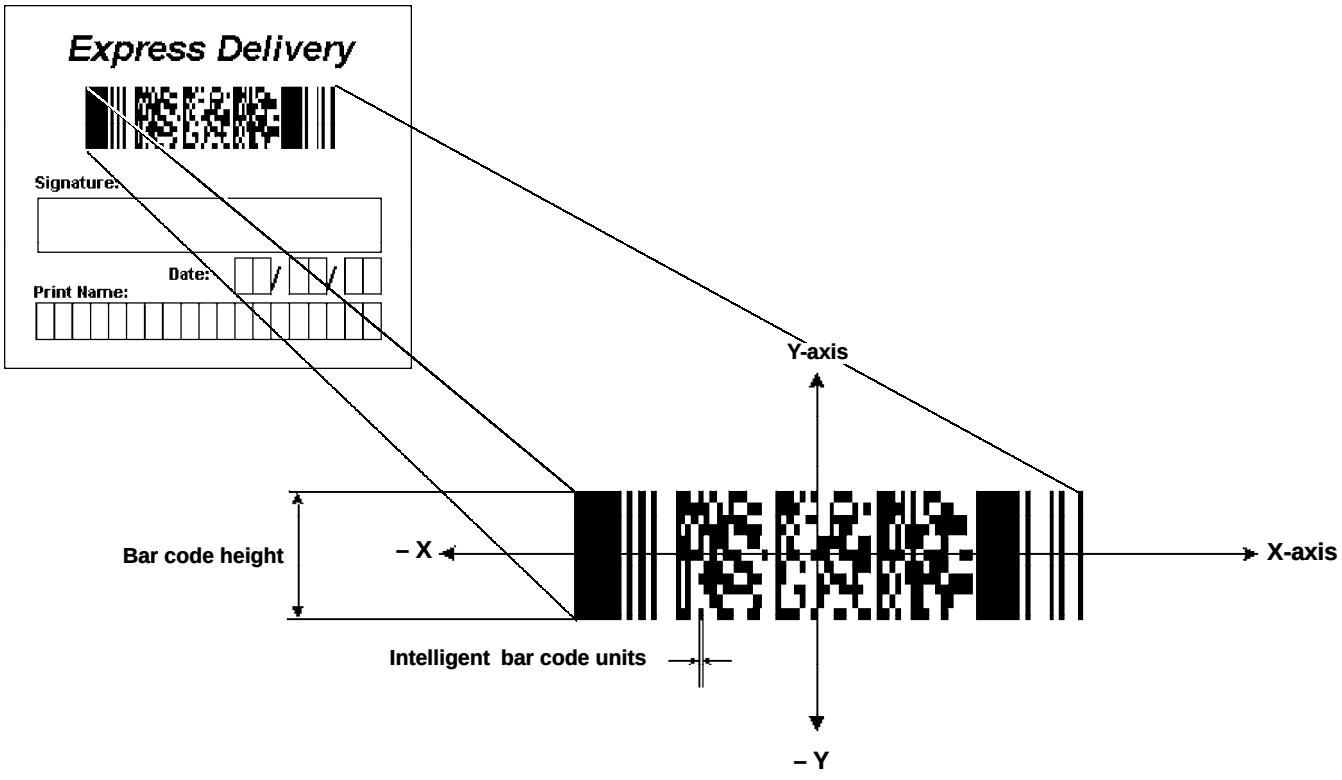
The integrated imager provides the following image acquisition features:

Real-time and Still Image Acquisition:
This includes functions that start and stop image acquisition and read acquired images.

Signature Capture:
This allows the application to retrieve an image of the normalized signature. This means the image is always oriented as if the picture were taken at right angles to the signature, at the same distance, and in the center of the image no matter in what orientation the picture was taken.

Signature capture requires a PDF 417 or Code 128 bar code symbology to be present in the image and requires the application to identify the X,Y offsets relative to the center the bar code, the X,Y dimension of image to be captured, and the aspect ratio of the bar code. Note the units are in terms of the narrow element width of the bar code.

See the following example signature capture label and dimensions. These image acquisition features are provided through the IImage Interface defined on page 6-65.



Create and Delete ADC COM Object Functions

Use these functions to create and release ADC COM interfaces. ITCDEVMGMT.H is the header file and ITCDEVMGMT.LIB is the library.

ITCDeviceOpen

This function opens and initializes a communication channel to the device. In C++, this function returns a pointer to an interface on which the methods are called. In C, this function returns a handle, which is the first parameter in each of the interface function calls.

Syntax:

HRESULT **ITCDeviceOpen(** LPCTSTR *pszDeviceName*, REFIID *iid*, ITC_DEVICE_FLAGS *eDeviceFlags*, void** *ppvObject* **);**

Parameters:

pszDevice	[in]	Pointer to a string that contains the device name on which to initialize the logical connection. The device name (Merlin 1) identifies a communications port. Use "default" for all internal scanners, such as Imager, SE900, etc. Use "ExtScanner" for tethered scanners.
iid	[in]	The identifier of the interface being requested.
eDeviceFlags	[in]	Enumeration that identifies the read characteristics as follows: **ITC_DHDEVFLAG_READAHEAD** Data is buffered on behalf of the calling applications. Data Buffering starts after the first call to IADC::Read (). **ITC_DHDEVFLAG_NODATA** The client application is managing the device to set its configuration or control its interface but not to collect data from the device.
ppvObject	[out]	A pointer to the interface pointer identified by *iid*. If the object does not support this interface, *ppvObject* is set to NULL.

Return Values:

HRESULT that indicates success or failure.

See Also:

ITCDeviceClose

ITCDeviceClose

This function closes the interface opened with ITCDeviceOpen.

Syntax:

HRESULT **ITCDeviceClose(** IUnknown** *ppvObject* **);**

Parameters:

ppvObject	[in,out]	A pointer to the interface pointer created by ITCDeviceOpen. If successful on output, this pointer is set to NULL.

Remarks:

On Windows, this interface decrements the reference count. So alternatively, IUnknown::Release() could be used and must be used if reference counting is performed with IUnknown::AddRef(). On DOS, this function closes all resources associated with the channel.

IADC Functions

IADC functions provide ADC data in an input device independent manner. This interface can receive bar code data, RFID data, and other ADC data from data collection engines, such as a bar code scanner. Use IADC functions if bar code specifics such as symbology are not important to the application.

IADC functions are the following. IADC.H is the header file and ITCUUID.LIB contains the IID_IADC Interface GUID value used to obtain the interface.

IADC::CancelReadRequest

This function cancels a pending Read() request. This call can be made on a separate thread as a Read() or on the same thread. On a separate thread, the function is useful in unblocking a blocked Read() so that other operations can be performed. On the same thread, this function is useful in stopping data from being collected on behalf of a Read Ahead Client.

Syntax:

HRESULT **IADC::CancelReadRequest(** BOOL *FlushBufferedData*, WORD **pwTotalDiscardedMessages*, DWORD **pdwTotalDiscardedBytes* **);**

Parameters:

FlushBufferedData	[in]	True	Flush and discard all already buffered data.
		False	Do not discard data, data will be returned on the next read call.
pwTotalDiscardedMessages	[in/out]		Total number of discarded buffered labels or tags.
pdwTotalDiscardedBytes			Total number of discarded bytes.

Return Values:

HRESULT that indicates success or failure.

Remarks:

The return value indicates whether a read was pending.

See Also:

IADC::Initialize
IADC::QueryAttribute
IADC::QueryData
IADC::Read
IADC::SetAttribute

IADC::Initialize

This function initializes a connection by opening a communications channel with a logical reader engine. The communications port is implicitly identified. This communication channel is required to collect data or configure the device.

Syntax:

HRESULT **IADC::Initialize (** LPCTSTR *pszDeviceName*,
ITC_DEVICE_FLAGS *eDeviceFlags* **) ;**

Parameters:

pszDeviceName	[in]	Pointer to a string that contains the device name on which to initialize the logical connection. The device name (Merlin 1) identifies a communications port. Use "default" for all internal scanners, such as Imager, SE900, etc. Use "ExtScanner" for tethered scanners.
eDeviceFlags	[in]	Enumeration that identifies the read characteristic as follows: **ITC_DHDEVFLAG_READAHEAD** Data is buffered on behalf of the calling application. Data buffering starts after the first call to IADC::Read ().

Return Values:

HRESULT that indicates success or failure.

See Also:

IADC::CancelReadRequest
IADC::QueryAttribute
IADC::QueryData
IADC::Read
IADC::SetAttribute

IADC::QueryAttribute

This function retrieves a specified attribute that is device-independent. The specified attribute can be a grid or multiclient enable status.

Syntax:

HRESULT **IADC::QueryAttribute (**
ITC_ADC_ATTRIBUTE_ID *eAttribID*, BYTE *rgbBuffer[]*,
DWORD *dwBufferSize*, DWORD **pnBufferData* **);**

Parameters:

eAttribID	[in]	Specifies the attribute. Only one attribute can be queried at a time. See IADC::SetAttribute.
rgbBuffer	[out]	Contains buffer for the attribute to be queried. The structure of *lpBuffer* depends on the attribute being queried. See IADC::SetAttribute for a description of these structures.
dwBufferSize	[in]	The maximum number of bytes *rgbBuffer* can store.
pnBufferData	[out]	Pointer to the DWORD location to put the number of bytes stored in *rgbBuffer*.

Return Values:

HRESULT that indicates success or failure.

See Also:

IADC::CancelReadRequest
IADC::Initialize
IADC::QueryData
IADC::Read
IADC::SetAttribute

IADC::QueryData

This function returns the status of user input data that has been buffered.

Syntax:

HRESULT **IADC::QueryData (** DWORD **dwTotalBufferedBytes*,
WORD **wNumberOfMessages*, DWORD **dwNextMessageSize* **);**

Parameters:

dwTotalBufferedBytes	[out]	Total bytes buffered for connection.
wNumberOfMessages	[out]	Total messages buffered. For example, each buffer contains a single bar code scan.
dwNextMessageSize	[out]	Size (in bytes) of the next buffered message.

Return Values:

A standard status code that indicates success or failure.

See Also:

IADC::CancelReadRequest
IADC::Initialize
IADC::QueryAttribute
IADC::Read
IADC::SetAttribute

IADC::Read

This function requests user input data from the reader engine. This is a blocking function that returns either when there is data or after a timeout.

Syntax:

HRESULT **IADC::Read (** BYTE *rgbDataBuffer[]*,
DWORD *dwDataBufferSize*, DWORD *pnBytesReturned*,
SYSTEMTIME *pSystemTime*, DWORD *dwTimeout* **);**

Parameters:

rgbDataBuffer	[in]	Pointer to the buffer that receives the data from the device.
dwDataBufferSize	[in]	Maximum number of bytes that can be stored in *rgbDataBuffer*.
pnBytesReturned	[out]	Pointer to the DWORD location to store the number of bytes returned in *rgbDataBuffer*.
pSystemTime	[out]	Pointer to a SYSTEMTIME structure that will hold the time stamp of the received data. This can be NULL if a timestamp is not needed.
dwTimeout	[in]	Number of milliseconds caller waits for data. This parameter is ignored if the Read Ahead flag is not set. **0** If data is not available, returns quickly. **INFINITE** Waits until data is available.

Return Values:

HRESULT that indicates success or failure.

See Also:

IADC::CancelReadRequest
IADC::Initialize
IADC::QueryAttribute
IADC::QueryData
IADC::SetAttribute

IADC::SetAttribute

This function changes an attribute such as a grid specification.

Syntax:

HRESULT **IADC::SetAttribute (** ITC_ADC_ATTRIBUTE_ID *eAttribID*, BYTE *rgbData[]*, DWORD *nBufferSize* **);**

Parameters:

eAttribID [in] Identifies the attribute to set. Only one attribute can be set at a time. The attribute is:

ITC_MULTICLIENT_ENABLE
Indicates whether this client can coexist with other clients.

rgbData [in] Contains data for the attribute to be set. Depending on the *eAttribID*, this will be mapped to the appropriate structure as follows:

ITC_MULTICLIENT_ENABLE
BOOL is the *rgbData* Data Structure.

TRUE, Client can receive data with other clients *(default)*.

FALSE, Data stream to this client is turned off when there are other clients.

ITC_DHATTR_READFILTER
ITC_READFILTER is the *rgbData* Data Structure. The ITC_READFILE structure is defined in IADCDEVICE.H as follows:

```
typedef struct
      {
         #define ITC_MAXFILTER_CHARS  240
         WORD     nFilterChars;
         TCHAR    szFilter[ITC_MAXFILTER_CHARS];
      } ITC_READFILTER;
```

where:

ITC_MAXFILTER_CHARS
Maximum number of characters in a filter specification. Includes NULL termination.

nFilterChars Number of characters in *pszDataMask*.

szFilter Data mask specification. See "*Grid Data Filtering*."

nBufferSize [in] Number of bytes in *rgbData*.

ITC_DHATTR_READFILTER
Regular expression that performs data filtering and data editing. See "*Grid Data Filtering*" on page 6-4 for more information.

Return Values:

A standard status code that indicates success or failure**.**

See Also:

IADC::CancelReadRequest
IADC::Initialize
IADC::QueryAttribute
IADC::QueryData
IADC::Read

IBarCodeReaderControl Functions

IBarCodeReaderControl functions provide functionality for bar code collection and control only. These functions allow an application to:

- Trigger the bar code laser scanner
- Disable the scanner
- Receive a bar code with details such as symbology scanned, data type (Unicode, ASCII), and the time the data was received.

These functions include the following. IBARCODEREADER.H is the header file and ITCUUID.LIB contains the IID_IADC Interface GUID value used to obtain the interface.

IBarCodeReaderControl::CancelReadRequest

This function cancels a pending IBarCodeReaderControl::Read request. If the read request is blocked, issue the CancelReadRequest from a separate thread.

Syntax:

HRESULT **IBarCodeReaderControl::CancelReadRequest(**
BOOL *FlushBufferedData*, WORD **pwTotalDiscardedMessages*,
WORD **pwTotalDiscardedBytes* **);**

Parameters:

FlushBufferedData	[in]	TRUE	Flushes and discards all buffered data.
		FALSE	Does not discard data; data will be returned on the next read call.
pwTotalDiscardedMessages	[in/out]		Total number of discarded buffered labels or tags.
pwTotalDiscardedBytes			Total number of discarded bytes.

Return Values:

HRESULT that indicates success or failure.

See Also:

IBarCodeReaderControl::ControlLED
IBarCodeReaderControl::Initialize
IBarCodeReaderControl::IssueBeep
IBarCodeReaderControl::QueryAttribute
IBarCodeReaderControl::Read
IBarCodeReaderControl::SetAttribute
IBarCodeReaderControl::TriggerScanner

IBarCodeReaderControl::ControlLED

This function controls LED illumination on a tethered scanner. The good read LED and any valid LEDs will be turned on and off based on defined parameters.

Syntax:

HRESULT **IBarCodeReaderControl::ControlLED(**
ITC_BARCODE_LASER_LED_ID *eLED*, BOOL *fLedOn* **);**

Parameters:

eLED [in] The specified LED identifier.
ITC_BARCODE_LASER_GOOD_READ_LED
Identifies the good read LED.

fLedOn [in] TRUE turns on the LED. FALSE turns off the LED.

Return Values:

HRESULT that indicates success or failure.

Remarks:

This function does not coordinate LED control with the scanner. If the scanner LED control is enabled, function results will be unpredictable.

See Also:

IBarCodeReaderControl::CancelReadRequest
IBarCodeReaderControl::Initialize
IBarCodeReaderControl::IssueBeep
IBarCodeReaderControl::QueryAttribute
IBarCodeReaderControl::Read
IBarCodeReaderControl::SetAttribute
IBarCodeReaderControl::TriggerScanner

IBarCodeReaderControl::Initialize

This function opens and initializes a communications channel with a logical bar code reader engine.

Syntax:

HRESULT **IBarCodeReaderControl::Initialize (**
LPCTSTR *pszDeviceName*, ITC_DEVICE_FLAGS *eDeviceFlags* **) ;**

Parameters:

pszDeviceName [in] Pointer to a string with device on which to initialize the logical connection. The device identifies a communications port. Use "default" for all internal scanners, such as Imager, SE900, etc. Use "ExtScanner" for tethered scanners.

eDeviceFlags [in] Enumeration that identifies the read characteristic as follows:

ITC_DHDEVFLAG_READAHEAD
Data is buffered on behalf of the calling applications. Data Buffering starts after the first call to IADC::Read ().

Return Values:

HRESULT that indicates success or failure.

See Also:

IBarCodeReaderControl::CancelReadRequest
IBarCodeReaderControl::ControlLED
IBarCodeReaderControl::IssueBeep
IBarCodeReaderControl::QueryAttribute
IBarCodeReaderControl::Read
IBarCodeReaderControl::SetAttribute
IBarCodeReaderControl::TriggerScanner

IBarCodeReaderControl::IssueBeep

This function causes the reader engine to generate a high beep, a low beep, or a custom beep. The high beep and low beep are preconfigured beep tones and durations. The custom beep allows the client to specify the frequency and duration. The volume is the current volume setting. *Note this is not implemented.*

Syntax:

HRESULT **IBarCodeReaderControl::IssueBeep(**
ITC_BEEP_SPEC *rgBeepRequests[]*, DWORD *dwNumberOfBeeps* **);**

Parameters:

rgBeepRequests [in] Array of ITC_BEEP_SPEC structures that identifies the beep type. The beep structure is:

```
typedef struct tagITCBeepSpec
{
ITC_BEEP_TYPE eBeepType; // Identifies the type of beep
// Following fields used only if the beep type is ITC_CUSTOM_BEEP.
WORD wPitch; // Frequency, in Hz, of the beep.
WORD wOnDuration; // Duration, in milliseconds, of Beep On.
WORD wOffDuration; // Duration, in milliseconds, of Beep Off
// Beep Off is used to separate individual beeps
} ITC_BEEP_SPEC;
typedef enum tagITCBeepType
{
ITC_LOW_BEEP, // Issue the default low beep.
ITC_HIGH_BEEP, // Issue the default high beep.
ITC_CUSTOM_BEEP, // Issue a custom beep.
} ITC_BEEP_TYPE;
```

dwNumberOfBeeps [in] Identifies the total number of beeps in *rgBeepRequests*.

Return Values:

E_NOTIMPL as this function is not implemented.

See Also:

IBarCodeReaderControl::CancelReadRequest
IBarCodeReaderControl::ControlLED
IBarCodeReaderControl::Initialize
IBarCodeReaderControl::QueryAttribute
IBarCodeReaderControl::Read
IBarCodeReaderControl::SetAttribute
IBarCodeReaderControl::TriggerScanner

IBarCodeReaderControl::QueryAttribute

This function retrieves the device-specific grid, the scanner enable status, and the LED control status for the current bar code reader engine.

Syntax:

HRESULT **IBarCodeReaderControl::QueryAttribute (** ITC_BARCODEREADER_ATTRIBUTE_ID *eAttr*, BYTE *rgbAttrBuffer[]*, DWORD *dwAttrBufferSize* **);**

Parameters:

eAttr	[in]	Specifies the attribute. See IBarCodeReaderControl::SetAttribute on page 6-24 for the attributes.
rgbAttrBuffer	[out]	Contains buffer for the attribute to be queried. The structure of *rgbAttrBuffer* depends on the attribute being queried. See IBarCodeReaderControl::SetAttritbute for a description of these structures.
dwAttrBufferSize	[in]	The maximum number of bytes that *rgbAttrBuffer* can store.

Remarks:

The following attributes are not supported on the imager:

ITC_RDRATTR_TONE_ENABLE
ITC_RDRATTR_VOLUME_LEVEL
ITC_RDRATTR_TONE_FREQUENCY
ITC_RDRATTR_GOOD_READ_BEEPS_NUMBER
ITC_RDRATTR_GOOD_READ_BEEP_DURATION

Return Values:

A standard status code that indicates success or failure.

See Also:

IBarCodeReaderControl::CancelReadRequest
IBarCodeReaderControl::ControlLED
IBarCodeReaderControl::Initialize
IBarCodeReaderControl::IssueBeep
IBarCodeReaderControl::Read
IBarCodeReaderControl::SetAttribute
IBarCodeReaderControl::TriggerScanner

IBarCodeReaderControl::Read

This function reads data from the bar code input device. This method performs the same function as IADC::Read () except that it provides additional information about data received such as bar code symbology used, data type, and time stamp of received data.

Syntax:

HRESULT **IBarCodeReaderControl::Read (** BYTE *rgbDataBuffer[]*, DWORD *dwDataBufferSize*, DWORD *pnBytesReturned*, ITC_BARCODE_DATA_DETAILS *pBarCodeDataDetails*, DWORD *dwTimeout* **);**

Parameters:

rgbDataBuffer [in] Pointer to the buffer that receives data from the device.

dwDataBufferSize [in] Maximum number of bytes that can be stored in *rgbDataBuffer*.

pnBytesReturned [out] Pointer to the DWORD location that will store the bytes returned in *rgbDataBuffer*.

pBarCodeDataDetails [in] Address of data structure in which to put the data details. This field may be NULL. The ITC_BARCODE_DATA_DETAILS is:

```
typedef struct tagITCBarCodeDetails

{

WORD wStructSize,
ITC_BARCODE_SYMBOLOGY_ID eSymbology,
ITC_BARCODE_DATATYPE eDataType,
SYSTEMTIME stTimeStamp,
}ITC_BARCODE_DATA_DETAILS;

typedef enum tagBarCodeDataType

{

BARCODE_DATA_TYPE_UNKNOWN = 1,
BARCODE_DATA_TYPE_ASCII,
BARCODE_DATA_TYPE_UNICODE,
}ITC_BARCODE_DATATYPE;
```

where:

wStructSize Size of structure. Used for versioning structure.

eSymbology Symbology of the returned data.

eDataType Identifies data types as ASCII, UNICODE, etc.

stTimeStamp Timestamp of the received data.

BARCODE_DATA_TYPE_UNKNOWN Data in unknown.

BARCODE_DATA_TYPE_ASCII Data is ASCII.

BARCODE_DATA_TYPE_UNICODE Data is UNICODE.

dwTimeout [in] Number of milliseconds caller waits for data. If you set a timeout, the call will be blocked until data is received.

0
If data not available, returns quickly.

INFINITE
Waits until data is available.

Return Values:

HRESULT that indicates success or failure.

See Also:

IBarCodeReaderControl::CancelReadRequest
IBarCodeReaderControl::ControlLED
IBarCodeReaderControl::Initialize
IBarCodeReaderControl::IssueBeep
IBarCodeReaderControl::QueryAttribute
IBarCodeReaderControl::SetAttribute
IBarCodeReaderControl::TriggerScanner

IBarCodeReaderControl::SetAttribute

This function enables and disables the laser scanner, sets the bar code reader engine specific grid, and enables or disables the reader engine LED control.

Syntax:

HRESULT **IBarCodeReaderControl::SetAttribute (**
ITC_BARCODEREADER_ATTRIBUTE_ID *eAttr*, BYTE *rgbAttrBuffer[]*,
DWORD *dwAttrBufferSize* **);**

Parameters:

eAttr [in] Identifies the attribute to set. Only one attribute can be set at a time. The attributes are:

ITC_RDRATTR_SCANNER_ENABLE
Enable or disable scanner for all connections.
ITC_RDRATTR_GOOD_READ_LED_ENABLE
Enables and disables the reader engine from controlling the good read LED.
ITC_RDRATTR_TONE_ENABLE
Enables and disables the reader engine from issuing beeps.
ITC_RDRATTR_VOLUME_LEVEL
Sets beep volume level.
ITC_RDRATTR_TONE_FREQUENCY
Sets beep frequency.
ITC_RDRATTR_GOOD_READ_BEEPS_NUMBER
Sets number of beeps for a good read.
ITC_RDRATTR_GOOD_READ_BEEP_DURATION
Sets duration of beeps for a good read.
ITC_DHATTR_READFILTER
ITC_READFILTER is the *rgbData* Data Structure. The ITC_READFILE structure is defined in IADCDEVICE.H as follows:

```
typedef struct
      {
         #define ITC_MAXFILTER_CHARS  240
         WORD nFilterChars;
         TCHAR szFilter[ITC_MAXFILTER_CHARS];
      } ITC_READFILTER;
```

where:

nFilterChars Number of characters in *pszDataMask*.

szFilter Data mask specification. See "*Grid Data Filtering*."

ITC_MAXFILTER_CHARS
Maximum number of characters in a filter specification. Includes NULL termination.

rgbAttrBuffer [in] Contains data for the attribute to be set. Depending on *eAttr*, the *rgbAttrData* will be mapped to the appropriate structure as shown in the table on the next page.

eAttr	**Data Structure contained in *rgbAttrBuffer***
ITC_RDRATTR_GRID	ITC_BARCODEREADER_READER_GRID Reader Engine specific grid only.
ITC_RDRATTR_SCANNER_ENABLE	BOOL TRUE Enable scanner. FALSE Disable scanner.
ITC_RDRATTR_GOOD_READ_LED_ENABLE	BOOL TRUE Reader Engine controls good read LED. FALSE Good read LED is not controlled.
ITC_RDRATTR_DATA_VALID_LED_ENABLE	BOOL TRUE Reader Engine controls data valid LED. FALSE Data valid LED is not controlled.
ITC_RDRATTR_TONE_ENABLE	BOOL TRUE Reader Engine issues beeps. FALSE Beeps are not issued.

ITC_RDRATTR_VOLUME_LEVEL	ITC_BEEP_VOLUME
	An enumerator that identifies the beep volume level control. Valid range for S9C:
	`typedef enum tagBeepVolume`
	`{`
	`ITC_BEEP_VOLUME_LOW = 0,`
	`ITC_BEEP_VOLUME_MEDIUM = 2,`
	`ITC_BEEP_VOLUME_HIGH = 1 //Default`
	`}ITC_BEEP_VOLUME`
	
	Note*: Due to the hardware design on this 700 Series Computer, the volume level can be either OFF (ITC_BEEP_VOLUME_LOW) or ON (ITC_BEEP_VOLUME_MEDIUM|HIGH).*

ITC_RDRATTR_TONE_FREQUENCY	DWORD A value that identifies the tone frequency in Hz. Valid range for S9C: 1000–4095 Hz *(default: 2090).* ***Note****: The value is divided by 10 for storage. On retrieval, the scanner rounds off the value to the nearest 10 Hz, then multiplies the value by 10. For example, the value sent to the scanner is 2095. On retrieval, the value returned is 2090.*

ITC_RDRATTR_GOOD_READ_BEEPS_NUMBER	ITC_GOOD_READ_BEEPS_NUMBER
	An enumerator identifying the good read beeps number. Valid range for S9C:
	`typedef enum tagGoodReadBeepsNumber`
	`{`
	`ITC_NUM_BEEPS_NONE = 0,`
	`ITC_NUM_BEEPS_ONE = 1, // Default`
	`ITC_NUM_BEEPS_TWO = 2`
	`}ITC_GOOD_READ_BEEPS_NUMBER`

ITC_RDRATTR_GOOD_READ_BEEP_DURATION	DWORD A value identifying the good read beep duration in ms. Valid range for S9C: 0–2550 ms *(Default: 80).* ***Note****: The value is divided by 10 for storage. On retrieval, the scanner rounds the value to the nearest 10 ms, then multiplies the value by 10.*

dwAttrBufferSize [in] The size of *rgbAttrBuffer* in bytes.

Return Values:

HRESULT that indicates success or failure**.**

Remarks:

Read ahead and non-read ahead clients can change the grid. Since changing the grid changes the entire reader engine grid, use IBarCodeReaderControl::QueryAttribute to retrieve the current reader engine grid and grid changes before sending back using SetAttribute. The grid structure is *typedef struct tagBarCodeReaderGrid*.

```
{
ITC_DI_GRID stDIGrid; // Device independent grid.
ITC_DDBARCODE_GRID stDDGrid; // Reader engine dependent grid
DWORD dwDataSourceTypeMask;
} ITC_BARCODEREADER_GRID;

ITC_DI_GRID

typedef struct tagItcBarCodeGrid
{
DWORD dwSymbologyMask; // Symbologies to be received.
} ITC_DDBARCODE_GRID;
```

When the scanner is enabled, it scans when the scan button is pressed or the trigger is pulled. When the scanner is disabled, it does not respond when the scan button is pressed or the trigger is pulled.

The following attributes are not supported on the imager:

ITC_RDRATTR_TONE_ENABLE
ITC_RDRATTR_VOLUME_LEVEL
ITC_RDRATTR_TONE_FREQUENCY
ITC_RDRATTR_GOOD_READ_BEEPS_NUMBER
ITC_RDRATTR_GOOD_READ_BEEP_DURATION

See Also:

IBarCodeReaderControl::CancelReadRequest
IBarCodeReaderControl::ControlLED
IBarCodeReaderControl::Initialize
IBarCodeReaderControl::IssueBeep
IBarCodeReaderControl::QueryAttribute
IBarCodeReaderControl::Read
IBarCodeReaderControl::TriggerScanner

IBarCodeReaderControl::TriggerScanner

This function turns the scanner on and off. The client application must coordinate control of the scanner with the user.

Syntax:

HRESULT **IBarCodeReaderControl::TriggerScanner (**
BOOL *fScannerOn* **);**

Parameters:

fScannerOn [in] Set TRUE to turn the scanner on. Set FALSE to turn the scanner off.

Return Values:

HRESULT that indicates success or failure.

Remarks:

The scanner will be turned on or off independent of the actions of the users. The client application must coordinate control of the scanner with the user. When the scanner is turned on, its behavior is controlled by the trigger mode. That is, in one shot mode, the laser turns off when a label is scanned; in auto-trigger mode, the laser remains on.

See Also:

IBarCodeReaderControl::CancelReadRequest
IBarCodeReaderControl::ControlLED
IBarCodeReaderControl::Initialize
IBarCodeReaderControl::IssueBeep
IBarCodeReaderControl::QueryAttribute
IBarCodeReaderControl::Read
IBarCodeReaderControl::SetAttribute

IS9CConfig Functions

This interface provides methods to set and retrieve the 700 Series Computer bar code configuration. All supported symbologies are initialized to their defaults when the S9C firmware is loaded.

GET/SET functions use enumerations as their parameters. In most enumerations, there is an enumerator xx_NO_CHANGE (such as ITC_CODE39_NO_CHANGE), where xx refers to a particular enumeration. This enumerator can be used during a call to a SET to indicate that no change is to be made to that particular parameter. This prevents the called function from having to format the same S9C command and send down to the S9C scanner.

For all symbologies, to set a bar code length of "any length," use a value of "0" for the bar code length argument.

IS9CConfig functions are the following. IS9CCONFIG.H is the header file and ITCUUID.LIB contains the IID_IADC Interface GUID value used to obtain the interface.

IS9CConfig::GetCodabar

This function retrieves the current settings of Codabar symbology.

Syntax:

HRESULT **IS9CConfig::GetCodabar(**
ITC_CODABAR_DECODING* *peDecode*,
ITC_CODABAR_START_STOP* *peSS*, ITC_CODABAR_CLSI* *peCLSI*,
ITC_CODABAR_CHECK_DIGIT* *peCheck*,
ITC_BARCODE_LENGTH_ID* *peLengthId*, BYTE *rgbLengthBuff[]*,
DWORD* *pdwNumBytes* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_CODABAR_DECODING location to receive the decoding for Codabar symbology.
peSS	[out]	Pointer to the ITC_CODABAR_START_STOP location to receive the Start/Stop option.
peCLSI	[out]	Pointer to the ITC_CODABAR_CLSI location to receive the CLSI library system.
peCheck	[out]	Pointer to the ITC_CODABAR_CHECK_DIGIT location to receive the check digit.
peLengthId	[out]	Pointer to the ITC_BARCODE_LENGTH_ID location to receive an indicator of either ITC_BARCODE_LENGTH or ITC_BARCODE_FIXED_LENGTH.
rgbLengthBuff	[out,size_is(3)]	An array of bytes to receive 1 byte of data for ITC_BARCODE_LENGTH, or 3 bytes of data for ITC_BARCODE_FIXED_LENGTH.
pdwNumBytes	[out]	Pointer to the DWORD location to receive a number indicating number of bytes in *rbgLengthBuff[]*: 1 byte for ITC_BARCODE_LENGTH or 3 bytes for ITC_BARCODE_FIXED_LENGTH.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetCodabar

This function updates the Codabar settings with new values.

Syntax:

HRESULT **IS9CConfig::SetCodabar(**
ITC_CODABAR_DECODING *eDecode*,
ITC_CODABAR_START_STOP *eSS*, ITC_CODABAR_CLSI *eCLSI*,
ITC_CODABAR_CHECK_DIGIT *eCheck*,
ITC_BARCODE_LENGTH_ID *eLengthId*, BYTE *rgbLengthBuff[]*,
DWORD *dwNumBytes* **);**

Parameters:

eDecode	[in]	Identifies the decoding for Codabar symbology.
eSS	[in]	Identifies the Start/Stop option.
eCLSI	[in]	Identifies the CLSI library system.
eCheck	[in]	Identifies the check digit.
eLengthId	[in]	Use ITC_BARCODE_LENGTH_NO_CHANGE to indicate no change for bar code length. Use ITC_BARCODE_LENGTH for any length and minimum length, and set *rgbLengthBuff[0]* to a valid length value. Use ITC_BARCODE_FIXED_LENGTH to compose 1 or 2 or 3 fixed lengths, and set 3 bytes: *rgbLengthBuff[0]*, *rgbLengthBuff[1]*, *rgbLengthBuff[2]* with valid values.
rgbLengthBuff	[in,size_is(dwNumBytes)]	An array of bytes containing bar code lengths when *eLengthId* = ITC_BARCODE_LENGTH or ITC_BARCODE_FIXED_LENGTH.
dwNumBytes	[in]	Number of bytes in *rbgLengthBuff[]*. For S9C, this value is: 1 When *eLengthId* = ITC_BARCODE_LENGTH 3 When *eLengthId* = ITC_BARCODE_FIXED_LENGTH

Return Values:

HRESULT that indicates success or failure.

Codabar Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_CODABAR_DECODING
CLSI Library System	Not Active	ITC_CODABAR_CLSI
Start/Stop	Not Transmitted	ITC_CODABAR_START_STOP
Check Digit	Not Used	ITC_CODABAR_CHECK_DIGIT
Bar Code Length	Minimum Length = 6	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

Codabar Enumerations

```
typedef enum tagCodabarDecoding
{
ITC_CODABAR_NOTACTIVE = 0,          // Default
ITC_CODABAR_ACTIVE = 1,
ITC_CODABAR_NO_CHANGE = 255
} ITC_CODABAR_DECODING;

typedef enum tagCodabarStartStop
{
ITC_CODABAR_SS_NOTXMIT,             // Default
ITC_CODABAR_SS_LOWERABCD,           // a,b,c,d
ITC_CODABAR_SS_UPPERABCD,           // A,B,C,D
ITC_CODABAR_SS_LOWERABCDTN,         // a,b,c,d / t,n,*,e
ITC_CODABAR_SS_DC1TODC4,            // DC1,DC2,DC3,DC4
ITC_CODABAR_SS_NO_CHANGE = 255
} ITC_CODABAR_START_STOP;

typedef enum tagCodabarClsi
{
ITC_CODABAR_CLSI_NOTACTIVE = 0,     // Default
ITC_CODABAR_CLSI_ACTIVE = 1,
ITC_CODABAR_CLSI_NO_CHANGE = 255
} ITC_CODABAR_CLSI;

typedef enum tagCodabarCheckDigit
{
ITC_CODABAR_CHECK_NOTUSED,          // Default
ITC_CODABAR_CHECK_XMIT,
ITC_CODABAR_CHECK_NOTXMIT,
ITC_CODABAR_CHECK_NO_CHANGE = 255
} ITC_CODABAR_CHECK_DIGIT;

typedef enum tagBarcodeLengthId
{
ITC_BARCODE_LENGTH = 0,
ITC_BARCODE_FIXED_LENGTH,
ITC_BARCODE_LENGTH_NO_CHANGE = 255
} ITC_BARCODE_LENGTH_ID;
```

IS9CConfig::GetCode39

This function retrieves the current settings of Code 39.

Syntax:

HRESULT **IS9Cconfig::GetCode39(**
ITC_CODE39_DECODING* *peDecode*,
ITC_CODE39_FORMAT* *peFormat*, ITC_CODE39_START_STOP* *peSS*,
ITC_CODE39_SS_CHARS* *peSSChars*,
ITC_CODE39_CHECK_DIGIT* *peCheck*, DWORD* *pwLength* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_CODE39_DECODING location to receive the decoding for Code 39.
peFormat	[out]	Pointer to the ITC_CODE39_FORMAT location to receive the Code 39 format.
peSS	[out]	Pointer to the ITC_CODE39_START_STOP location to receive the Code 39 start/stop.
peSSChars	[out]	Pointer to the ITC_CODE39_SS_CHARS location to receive the Start/Stop character.
peCheck	[out]	Pointer to the ITC_CODE39_CHECK_DIGIT location to receive the check digit.
pwLength	[out]	Pointer to the DWORD location to receive the bar code length.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetCode39

This function updates the Code 39 settings with new values.

Syntax:

HRESULT **IS9CConfig::SetCode39(** ITC_CODE39_DECODING *eDecode*,
ITC_CODE39_FORMAT *eFormat*, ITC_CODE39_START_STOP *eSS*,
ITC_CODE39_SS_CHARS *eSSChars*,
ITC_CODE39_CHECK_DIGIT *eCheck*, DWORD *dwLength* **);**

Parameters:

eDecode	[in]	Identifies the decoding for Code 39.
eFormat	[in]	Identifies the Code 39 Format.
eSS	[in]	Identifies the Start/Stop option.
eSSChars	[in]	Identifies the Start/Stop character.
eCheck	[in]	Identifies the Check digit.
dwLength	[in]	Identifies the bar code length.

Return Values:

HRESULT that indicates success or failure.

Code 39 Default Settings

Parameter	Default	Valid Range
Decoding	Active	ITC_CODE39_DECODING
Format	Standard 43 Characters	ITC_CODE39_FORMAT
Start/Stop	Not Transmitted	ITC_CODE39_START_STOP
Accepted Start/Stop Characters	* only	ITC_CODE39_SS_CHARS
Check Digit	Not Used	ITC_CODE39_CHECK_DIGIT
Bar Code Length	Any Bar Code Length	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

Code 39 Enumerations

```
typedef enum tagCode39Decoding
{
ITC_CODE39_NOTACTIVE = 0,
ITC_CODE39_ACTIVE = 1,              // Default
ITC_CODE39_NO_CHANGE = 255
} ITC_CODE39_DECODING;

typedef enum tagCode39Format
{
ITC_CODE39_FORMAT_STANDARD43,       // Default
ITC_CODE39_FORMAT_FULLASCII,
ITC_CODE39_FORMAT_NO_CHANGE = 255
} ITC_CODE39_FORMAT;

typedef enum tagCode39StartStop
{
ITC_CODE39_SS_NOTXMIT,              // Default
ITC_CODE39_SS_XMIT,
ITC_CODE39_SS_NO_CHANGE = 255
} ITC_CODE39_START_STOP;

typedef enum tagCode39StartStopChars
{
ITC_CODE39_SS_CHARS_DOLLARSIGN,
ITC_CODE39_SS_CHARS_ASTERISK,       // Default
ITC_CODE39_SS_CHARS_BOTH,
ITC_CODE39_SS_CHARS_NO_CHANGE = 255
} ITC_CODE39_SS_CHARS;

typedef enum tagCode39CheckDigit
{
ITC_CODE39_CHECK_NOTUSED,           // Default
ITC_CODE39_CHECK_MOD43_XMIT,
ITC_CODE39_CHECK_MOD43_NOTXMIT,
ITC_CODE39_CHECK_FRENCH_CIP_XMIT,
ITC_CODE39_CHECK_FRENCH_CIP_NOTXMIT,
ITC_CODE39_CHECK_ITALIAN_CPI_XMIT,
ITC_CODE39_CHECK_ITALIAN_CPI_NOTXMIT,
ITC_CODE39_CHECK_NO_CHANGE = 255
} ITC_CODE39_CHECK_DIGIT;
```

IS9CConfig::GetCode93

This function retrieves the current settings of Code 93.

Syntax:

HRESULT **IS9CConfig::GetCode93(**
ITC_CODE93_DECODING* *peDecode*, DWORD* *pdwLength* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_CODE93_DECODING location to receive the decoding for Code 93 symbology.
pdwLength	[out]	Pointer to the DWORD location to receive a value for bar code length.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetCode93

This function updates the Code 93 settings with new values.

Syntax:

HRESULT **IS9CConfig::SetCode93(** ITC_CODE93_DECODING *eDecode*, DWORD *dwLength* **);**

Parameters:

eDecode	[in]	Identifies the decoding for Code93 Symbology.
dwLength	[in]	Identifies the bar code length.

Return Values:

HRESULT that indicates success or failure.

Code 93 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_CODE93_DECODING
Bar Code Length	Any Bar Code Length	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

Code 93 Enumerations

Use this when the bar code length does not require any change.

```
typedef enum tagCode93Decoding
{
ITC_CODE93_NOTACTIVE = 0,       // Default
ITC_CODE93_ACTIVE = 1,
ITC_CODE93_NO_CHANGE = 255
} ITC_CODE93_DECODING;
#define ITC_BC_LENGTH_NO_CHANGE 255.
```

IS9CConfig::GetCode128

This function retrieves the current settings of Code 128 symbology.

Syntax:

HRESULT **IS9Cconfig::GetCode128(**
ITC_CODE128_DECODING* *peDecode*,
ITC_EAN128_IDENTIFIER* *peEan128Ident*,
ITC_CODE128_CIP128 *peCip128State*, BYTE* *pbyFNC1*,
DWORD* *pdwLength* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_CODE128_DECODING location to receive the decoding for Code 128 symbology.
peEan128Ident	[out]	Pointer to the ITC_EAN128_IDENTIFIER location to receive the EAN 128 identifier.
peCip128State	[out]	Pointer to the ITC_CODE128_CIP128 location to receive the CIP 128.
pbyFNC1	[out]	Pointer to the BYTE location to receive the FNC1 separator character.
pdwLength	[out]	Pointer to the DWORD location to receive a value for bar code length.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetCode128

This function updates the Code 128 settings with new values.

Syntax:

HRESULT **IS9CConfig::SetCode128(**
ITC_CODE128_DECODING *eDecode*,
ITC_EAN128_IDENTIFIER *eEan128Ident*,
ITC_CODE128_CIP128 *eCip128State*, BYTE *byFNC1*, DWORD *dwLength*
);

Parameters:

eDecode	[in]	Identifies the decoding for Code 128 symbology.
eEan128Ident	[in]	Identifies the EAN 128 identifier.
eCip128State	[in]	Identifies the CIP 128.
byFNC1	[in]	Identifies the FNC1 separator character, usually any ASCII value.
dwLength	[in]	Identifies the bar code length.

Return Values:

HRESULT that indicates success or failure.

Code 128/EAN 128 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_CODE128_DECODING
EAN 128 Identifier	Include]C1	ITC_EAN128_IDENTIFIER
CIP 128 French Pharmaceutical Codes	Not Active	ITC_CODE128_CIP128
FNC1 Separator Character (EAN 128 norms)	GS function Char ASCII 29 or 0x1D	0x00–0xFE ITC_CODE128_FNC1_NO_CHANGE
Bar Code Length	Any Bar Code Length	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

Code 128 Enumerations

```
typedef enum tagCode128Decoding
{
ITC_CODE128_NOTACTIVE = 0, // Default
ITC_CODE128_ACTIVE = 1,
ITC_CODE128_NO_CHANGE = 255
} ITC_CODE128_DECODING;
typedef enum tagEan128Identifier
{
ITC_EAN128_ID_REMOVE,
ITC_EAN128_ID_INCLUDE, // Default
ITC_EAN128_ID_NO_CHANGE = 255
} ITC_EAN128_IDENTIFIER;
typedef enum tagCode128Cip128
{
ITC_CODE128_CIP128_NOTACTIVE = 0, // Default
ITC_CODE128_CIP128_ACTIVE = 1,
ITC_CODE128_CIP128_NO_CHANGE = 255
} ITC_CODE128_CIP128;

#define ITC_CODE128_FNC1_NO_CHANGE 255.
This definition can be used when the Code128 FNC1 does not require any change.

#define ITC_BC_LENGTH_NO_CHANGE 255. This definition can be used when the bar code length
does not require any change.
```

The table below shows what to be expected for EAN 128 labels for various symbology identifier transmit configurations and EAN 128 Identifier options.

Setup			Application's Expected Result	
EAN 128]C1 Id		**Symbology ID option**	**EAN 128 Label**	**Other Labels**
1	Include]C1	Disabled	<data>	<data>
2	Remove]C1	Disabled	<data>	<data>
3	Include]C1	AIM Id Transmitted	]C1<data>	]XY<data>
4	Remove]C1	AIM Id Transmitted	]C1<data>	]XY<data>
5	Include]C1	Custom Id Transmitted	Z]C1<data>	Z<data>
6	Remove]C1	Custom Id Transmitted	Z<data>	Z<data>

where "X" is the symbology identifier; "Y" is the modifier character; "Z" is the 1 byte symbology identifier

IS9CConfig::GetI2of5

This function retrieves the current settings of Interleaved 2 of 5.

Syntax:

HRESULT **IS9CConfig::GetI2of5(**
ITC_INTERLEAVED2OF5_DECODING* *peDecode*,
ITC_INTERLEAVED2OF5_CHECK_DIGIT* *peCheck*,
ITC_BARCODE_LENGTH_ID* *peLengthId*, BYTE *rbgLengthBuff[]*,
DWORD* *pdwNumBytes* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_INTERLEAVED2OF5_DECODING location to receive the decoding for Interleaved 2 of 5 symbology.
peCheck	[out]	Pointer to the ITC_INTERLEAVED2OF5_CHECK_DIGIT location to receive the check digit.
peLengthId	[out]	Pointer to the ITC_BARCODE_LENGTH_ID location to receive an indicator of either ITC_BARCODE_LENGTH or ITC_BARCODE_FIXED_LENGTH.
rgbLengthBuff	[out,size_is(3)]	An array of bytes to receives 1 byte of data for ITC_BARCODE_LENGTH or 3 bytes of data for ITC_BARCODE_FIXED_LENGTH.
pdwNumBytes	[out]	Pointer to the DWORD location to receive a number indicating number of bytes in *rbgLengthBuff[]*: 1 byte for ITC_BARCODE_LENGTH or 3 bytes for ITC_BARCODE_FIXED_LENGTH.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetI2of5

This function updates the Interleaved 2 of 5 settings with new values.

Syntax:

HRESULT **IS9CConfig::SetI2of5(**
ITC_INTERLEAVED2OF5_DECODING *eDecode*,
ITC_INTERLEAVED2OF5_CHECK_DIGIT *eCheck*,
ITC_BARCODE_LENGTH_ID *eLengthId*, BYTE *rgbLengthBuff[]*,
DWORD *dwNumBytes* **);**

Parameters:

eDecode [in] Identifies the decoding for Interleaved 2 of 5 symbology.

eCheck [in] Identifies the check digit.

eLengthId [in] Use ITC_BARCODE_LENGTH_NO_CHANGE to indicate no change for bar code length.
Use ITC_BARCODE_LENGTH for any length and minimum length, and set *rgbLengthBuff[0]* to a valid length value.
Use ITC_BARCODE_FIXED_LENGTH to compose 1 or 2 or 3 fixed lengths, and set 3 bytes: *rgbLengthBuff[0]*, *rgbLengthBuff[1]*, *rgbLengthBuff[2]* with valid values.

rgbLengthBuff [in,size_is(dwNumBytes)]
Contains bar code lengths when *eLengthId* =
Use ITC_BARCODE_LENGTH or
Use ITC_BARCODE_FIXED_LENGTH.

dwNumBytes [in] Number of bytes in *rbgLengthBuff[]*. For S9C, this value is:
1 When *eLengthId* = ITC_BARCODE_LENGTH,
3 When *eLengthId* = ITC_BARCODE_FIXED_LENGTH.

Return Values:

HRESULT that indicates success or failure.

Interleaved 2 of 5 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_INTERLEAVED2OF5_DECODING
Check Digit	Not Used	ITC_INTERLEAVED2OF5_CHECK_DIGIT
Bar Code Length	Minimum Length = 6	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

Interleaved 2 of 5 Enumerations

```
typedef enum tagInterleaved2of5Decoding
{
ITC_INTERLEAVED2OF5_NOTACTIVE = 0,         // Default
ITC_INTERLEAVED2OF5_ACTIVE = 1,
ITC_INTERLEAVED2OF5_NO_CHANGE = 255
} ITC_INTERLEAVED2OF5_DECODING;
typedef enum tagInterleaved2of5CheckDigit
{
ITC_INTERLEAVED2OF5_CHECK_NOTUSED,         // Default
ITC_INTERLEAVED2OF5_CHECK_MOD10_XMIT,
ITC_INTERLEAVED2OF5_CHECK_MOD10_NOTXMIT,
ITC_INTERLEAVED2OF5_CHECK_FRENCH_CIP_XMIT,
ITC_INTERLEAVED2OF5_CHECK_FRENCH_CIP_NOTXMIT,
ITC_INTERLEAVED2OF5_CHECK_NO_CHANGE = 255
} ITC_INTERLEAVED2OF5_CHECK_DIGIT;
typedef enum tagBarcodeLengthId
{
ITC_BARCODE_LENGTH = 0,
ITC_BARCODE_FIXED_LENGTH,
ITC_BARCODE_LENGTH_NO_CHANGE = 255
} ITC_BARCODE_LENGTH_ID;
```

IS9CConfig::GetMatrix2of5

This function retrieves the current settings of Matrix 2 of 5.

Syntax:

HRESULT **IS9CConfig::GetMatrix2of5(**
ITC_MATRIX2OF5_DECODING* *peDecode*, DWORD* *pdwLength* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_MATRIX2OF5_DECODING location to receive the decoding for Matrix 2 of 5 symbology.
pdwLength	[out]	Pointer to the DWORD location to receive a value for the bar code length.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetMatrix2of5

This function updates the Matrix 2 of 5 settings with new values.

Syntax:

HRESULT **IS9CConfig::SetMatrix2of5(**
ITC_MATRIX2OF5_DECODING *eDecode*, DWORD *dwLength* **);**

Parameters:

eDecode	[in]	Identifies the decoding for Matrix 2 of 5 symbology.
dwLength	[in]	Identifies the bar code length.

Return Values:

HRESULT that indicates success or failure.

Matrix 2 of 5 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_MATRIX2OF5_DECODING
Bar Code Length	Minimum Length = 6	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

Matrix 2 of 5 Enumerations

```
typedef enum tagMatrix2of5Decoding
{
ITC_MATRIX2OF5_NOTACTIVE = 0, // Default
ITC_MATRIX2OF5_ACTIVE = 1,
ITC_MATRIX2OF5_NO_CHANGE = 255
} ITC_MATRIX2OF5_DECODING;
#define ITC_BC_LENGTH_NO_CHANGE 255. This definition can be used when the bar code length
does not require any change.
```

IS9CConfig::GetMSI

This function retrieves the current MSI settings.

Syntax:

HRESULT **IS9CConfig::GetMSI(** ITC_MSI_DECODING* *peDecode*, ITC_MSI_CHECK_DIGIT* *peCheck*, DWORD* *pdwLength* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_MSI_DECODING location to receive the decoding for MSI symbology.
peCheck	[out]	Pointer to the ITC_MSI_CHECK_DIGIT location to receive the check digit.
pdwLength	[out]	Pointer to the DWORD location to receive the bar code length.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetMSI

This function updates the MSI settings with new values.

Syntax:

HRESULT **IS9CConfig::SetMSI(** ITC_MSI_DECODING *eDecode*, ITC_MSI_CHECK_DIGIT *eCheck*, DWORD *dwLength* **);**

Parameters:

eDecode	[in]	Identifies the decoding for MSI symbology.
eCheck	[in]	Identifies the check digit.
dwLength	[in]	Identifies the bar code length.

Return Values:

HRESULT that indicates success or failure.

MSI Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_MSI_DECODING
Check Digit	MOD 10 checked and transmitted	ITC_MSI_CHECK_DIGIT
Bar Code Length	Minimum Length = 6	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

MSI Enumerations

```
typedef enum tagMsiDecoding
{
ITC_MSI_NOTACTIVE = 0, // Default
ITC_MSI_ACTIVE = 1,
ITC_MSI_NO_CHANGE = 255
} ITC_MSI_DECODING;
typedef enum tagMsiCheckDigit
{
ITC_MSI_CHECK_MOD10_XMIT, // Default
ITC_MSI_CHECK_MOD10_NOTXMIT,
ITC_MSI_CHECK_DOUBLEMOD10_XMIT,
ITC_MSI_CHECK_DOUBLEMOD10_NOTXMIT,
ITC_MSI_CHECK_NO_CHANGE = 255
} ITC_MSI_CHECK_DIGIT;
#define ITC_BC_LENGTH_NO_CHANGE 255. This definition can be used when the bar code length
does not require any change.
```

IS9CConfig::GetPDF417

This function retrieves the current PDF417 settings.

Syntax:

HRESULT **IS9CConfig::GetPDF417(**
ITC_PDF417_DECODING* *pePdf417Decode*,
ITC_PDF417_MACRO_PDF* *peMacroPdf*,
ITC_PDF417_CTRL_HEADER* *pePdfControlHeader*,
ITC_PDF417_FILE_NAME* *pePdfFileName*,
ITC_PDF417_SEGMENT_COUNT* *pePdfSegmentCount*,
ITC_PDF417_TIME_STAMP* *pePdfTimeStamp*,
ITC_PDF417_SENDER* *pePdfSender*,
ITC_PDF417_ADDRESSEE* *pePdfAddressee*,
ITC_PDF417_FILE_SIZE* *pePdfFileSize*,
ITC_PDF417_CHECKSUM* *pePdfChecksum* **);**

Parameters:

pePdf417Decode	[out]	Pointer to the ITC_PDF417_DECODING location to receive the decoding for PDF417 symbology.
peMacroPdf	[out]	Pointer to the ITC_PDF417_MACRO_PDF location to receive the Macro PDF.
pePdfControlHeader	[out]	Pointer to the ITC_PDF417_CTRL_HEADER location to receive the control header.
pePdfFileName	[out]	Pointer to the ITC_PDF417_FILE_NAME location to receive the file name.
pePdfSegmentCount	[out]	Pointer to the ITC_PDF417_SEGMENT_COUNT location to receive the segment count.
pePdfTimeStamp	[out]	Pointer to the ITC_PDF417_TIME_STAMP location to receive the time stamp.
pePdfSender	[out]	Pointer to the ITC_PDF417_SENDER location to receive the sender.
pePdfAddressee	[out]	Pointer to the ITC_PDF417_ADDRESSEE location to receive the addressee.
pePdfFileSize	[out]	Pointer to the ITC_PDF417_FILE_SIZE location to receive the file size.
pePdfChecksum	[out]	Pointer to the ITC_PDF417_CHECKSUM location to receive the checksum.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetPDF417

This function updates the PDF417 settings with new values.

Syntax:

HRESULT **IS9CConfig::SetPDF417(**
ITC_PDF417_DECODING *ePdf417Decode*,
ITC_PDF417_MACRO_PDF *eMacroPdf*,
ITC_PDF417_CTRL_HEADER *ePdfControlHeader*,
ITC_PDF417_FILE_NAME *ePdfFileName*,
ITC_PDF417_SEGMENT_COUNT *ePdfSegmentCount*,
ITC_PDF417_TIME_STAMP *ePdfTimeStamp*,
ITC_PDF417_SENDER *ePdfSender*,
ITC_PDF417_ADDRESSEE *ePdfAddressee*,
ITC_PDF417_FILE_SIZE *ePdfFileSize*,
ITC_PDF417_CHECKSUM *ePdfChecksum* **);**

Parameters:

ePdf417Decode	[in]	Identifies the decoding for PDF417 symbology.
eMacroPdf	[in]	Identifies the Macro PDF.
ePdfControlHeader	[in]	Identifies the control header.
ePdfFileName	[in]	Identifies the file name.
ePdfSegmentCount	[in]	Identifies the segment count.
ePdfTimeStamp	[in]	Identifies the time stamp.
ePdfSender	[in]	Identifies the sender.
ePdfAddressee	[in]	Identifies the addressee.
ePdfFileSize	[in]	Identifies the file size.
ePdfChecksum	[in]	Identifies the checksum.

Return Values:

HRESULT that indicates success or failure.

PDF 417 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_PDF417_DECODING
Macro PDF	Macro PDF Buffered	ITC_PDF417_MACRO_PDF
Control Header	Not Transmitted	ITC_PDF417_CTRL_HEADER
*File Name	Not Transmitted	ITC_PDF417_FILE_NAME
*Segment Count	Not Transmitted	ITC_PDF417_SEGMENT_COUNT
*Time Stamp	Not Transmitted	ITC_PDF417_TIME_STAMP
*Sender	Not Transmitted	ITC_PDF417_SENDER
*Address	Not Transmitted	ITC_PDF417_ADDRESSEE
*File Size	Not Transmitted	ITC_PDF417_FILE_SIZE
*Check Sum	Not Transmitted	ITC_PDF417_CHECKSUM

** These are Macro PDF Optional Fields.*

PDF 417 Enumerations

```
typedef enum tagPdf417Decoding
{
ITC_PDF417_NOTACTIVE = 0,
ITC_PDF417_ACTIVE = 1,                  // Default
ITC_PDF417_NO_CHANGE = 255
} ITC_PDF417_DECODING;
typedef enum tagPdf417MacroPdf
{
ITC_PDF417_MACRO_UNBUFFERED = 0,
ITC_PDF417_MACRO_BUFFERED = 1,          // Default
ITC_PDF417_MACRO_NO_CHANGE = 255
} ITC_PDF417_MACRO_PDF;
typedef enum tagPdf417ControlHeader
{
ITC_PDF417_CTRL_HEADER_NOTXMIT = 0,     // Default
ITC_PDF417_CTRL_HEADER_XMIT = 1,
ITC_PDF417_CTRL_HEADER_NO_CHANGE = 255
} ITC_PDF417_CTRL_HEADER;
typedef enum tagPdf417FileName
{
ITC_PDF417_FILE_NAME_NOTXMIT = 0,       // Default
ITC_PDF417_FILE_NAME_XMIT = 1,
ITC_PDF417_FILE_NAME_NO_CHANGE = 255
} ITC_PDF417_FILE_NAME;
typedef enum tagPdf417SegmentCount
{
ITC_PDF417_SEGMENT_COUNT_NOTXMIT = 0,   // Default
ITC_PDF417_SEGMENT_COUNT_XMIT = 1,
ITC_PDF417_SEGMENT_COUNT_NO_CHANGE = 255
} ITC_PDF417_SEGMENT_COUNT;

typedef enum tagPdf417TimeStamp
{
ITC_PDF417_TIME_STAMP_NOTXMIT = 0,      // Default
ITC_PDF417_TIME_STAMP_XMIT = 1,
ITC_PDF417_TIME_STAMP_NO_CHANGE = 255
} ITC_PDF417_TIME_STAMP;
typedef enum tagPdf417Sender
{
ITC_PDF417_SENDER_NOTXMIT = 0,          // Default
ITC_PDF417_SENDER_XMIT = 1,
ITC_PDF417_SENDER_NO_CHANGE = 255
} ITC_PDF417_SENDER;
typedef enum tagPdf417Addressee
{
ITC_PDF417_ADDRESSEE_NOTXMIT = 0,       // Default
ITC_PDF417_ADDRESSEE_XMIT = 1,
ITC_PDF417_ADDRESSEE_NO_CHANGE = 255
} ITC_PDF417_ADDRESSEE;
typedef enum tagPdf417FileSize
{
ITC_PDF417_FILE_SIZE_NOTXMIT = 0,       // Default
ITC_PDF417_FILE_SIZE_XMIT = 1,
ITC_PDF417_FILE_SIZE_NO_CHANGE = 255
} ITC_PDF417_FILE_SIZE;
typedef enum tagPdf417Checksum
{
ITC_PDF417_CHECKSUM_NOTXMIT = 0,        // Default
ITC_PDF417_CHECKSUM_XMIT = 1,
ITC_PDF417_CHECKSUM_NO_CHANGE = 255
} ITC_PDF417_CHECKSUM;
```

IS9CConfig::GetPlessey

This function retrieves the current Plessey settings.

Syntax:

HRESULT *IS9CConfig::GetPlessey(*
ITC_PLESSEY_DECODING* *peDecode*,
ITC_PLESSEY_CHECK_DIGIT* *peCheck*, DWORD* *pdwLength* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_PLESSEY_DECODING location to receive the decoding for Plessey symbology.
peCheck	[out]	Pointer to the ITC_PLESSEY_CHECK_DIGIT location to receive the check digit.
pdwLength	[out]	Pointer to the DWORD location to receive the bar code length.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetPlessey

This function updates the Plessey settings with new values.

Syntax:

HRESULT **IS9CConfig::SetPlessey(**
ITC_PLESSEY_DECODING *eDecode*,
ITC_PLESSEY_CHECK_DIGIT *eCheck*, DWORD *dwLength* **);**

Parameters:

eDecode	[in]	Identifies the decoding for Plessey symbology.
eCheck	[in]	Identifies the check digit.
dwLength	[in]	Identifies the bar code length.

Return Values:

HRESULT that indicates success or failure.

Plessey Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_PLESSEY_DECODING
Check Digit	Not Transmitted	ITC_PLESSEY_CHECK_DIGIT
Bar Code Length	Any Bar Code Length	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

Plessey Enumerations

```
typedef enum tagPlesseyDecoding
{
ITC_PLESSEY_NOTACTIVE = 0,          // Default
ITC_PLESSEY_ACTIVE = 1,
ITC_PLESSEY_NO_CHANGE = 255
} ITC_PLESSEY_DECODING;
typedef enum tagPlesseyCheckDigit
{
ITC_PLESSEY_CHECK_NOTXMIT = 0,      // Default
ITC_PLESSEY_CHECK_XMIT = 1,
ITC_PLESSEY_CHECK_NO_CHANGE = 255
} ITC_PLESSEY_CHECK_DIGIT;
#define ITC_BC_LENGTH_NO_CHANGE 255. This definition can be used when the bar code length
does not require any change.
```

IS9CConfig::GetStandard2of5

This function retrieves the current Standard 2 of 5 settings.

Syntax:

HRESULT **IS9CConfig::GetStandard2of5(**
ITC_STANDARD2OF5_DECODING* *peDecode*,
ITC_STANDARD2OF5_FORMAT* *peFormat*,
ITC_STANDARD2OF5_CHECK_DIGIT* *peCheck*,
ITC_BARCODE_LENGTH_ID* *peLengthId*, BYTE *rgbLengthBuff*,
DWORD* *pdwNumBytes* **);**

Parameters:

peDecode	[out]	Pointer to the ITC_STANDARD2OF5_DECODING location to receive the decoding for Standard 2 of 5 symbology.
peFormat	[out]	Pointer to the ITC_STANDARD2OF5_FORMAT location to receive the format.
peCheck	[out]	Pointer to the ITC_STANDARD2OF5_CHECK_DIGIT location to receive Modulo 10 check digit.
peLengthId	[out]	Pointer to the ITC_BARCODE_LENGTH_ID location to receive an indicator of either ITC_BARCODE_LENGTH or ITC_BARCODE_FIXED_LENGTH.
rgbLengthBuff	[out,size_is(3)]	An array of bytes to receives 1 byte of data for ITC_BARCODE_LENGTH, or 3 bytes of data for ITC_BARCODE_FIXED_LENGTH.
pdwNumBytes	[out]	Pointer to the DWORD location to receive a number indicating number of bytes in *rbgLengthBuff[]*: 1 byte for ITC_BARCODE_LENGTH or 3 bytes for ITC_BARCODE_FIXED_LENGTH.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetStandard2of5

This function updates the Standard 2 of 5 settings with new values.

Syntax:

HRESULT **IS9CConfig::SetStandard2of5(**
ITC_STANDARD2OF5_DECODING *eDecode*,
ITC_STANDARD2OF5_FORMAT *eFormat*,
ITC_STANDARD2OF5_CHECK_DIGIT *eCheck*,
ITC_BARCODE_LENGTH_ID *eLengthId*, BYTE *rgbLengthBuff[]*,
DWORD *dwNumBytes* **);**

Parameters:

eDecode [in] Identifies the decoding for Standard 2 of 5 symbology.

eFormat [in] Identifies the format.

eCheck [in] Identifies the Modulo 10 check digit.

eLengthId [in] Use ITC_BARCODE_LENGTH_NO_CHANGE to indicate no change for bar code length.
Use ITC_BARCODE_LENGTH for any length and minimum length, and set rgbLengthBuff[0] to a valid length value.
Use ITC_BARCODE_FIXED_LENGTH to compose 1 or 2 or 3 fixed lengths, and set 3 bytes: *rgbLengthBuff[0]*, *rgbLengthBuff[1]*, *rgbLengthBuff[2]* with valid values.

rgbLengthBuff [in,size_is(dwNumBytes)]
An array of bytes containing bar code lengths when *eLengthId* = ITC_BARCODE_LENGTH or ITC_BARCODE_FIXED_LENGTH.

dwNumBytes [in] Number of bytes in *rbgLengthBuff[]*. For S9C, this value is:
1 When *eLengthId* = ITC_BARCODE_LENGTH,
3 When *eLengthId* = ITC_BARCODE_FIXED_LENGTH.

Return Values:

HRESULT that indicates success or failure.

Standard 2 of 5 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_STANDARD2OF5_DECODING
Format	Identicon (6 Start/Stop bars)	ITC_STANDARD2OF5_FORMAT
Check Digit	Not Used	ITC_STANDARD2OF5_CHECK_DIGIT
Bar Code Length	Minimum Length = 6	0x00–0xFE ITC_BC_LENGTH_NO_CHANGE

Standard 2 of 5 Enumerations

```
typedef enum tagStandard2of5Decoding
{
ITC_STANDARD2OF5_NOTACTIVE = 0, // Default
ITC_STANDARD2OF5_ACTIVE = 1,
ITC_STANDARD2OF5_NO_CHANGE = 255
} ITC_STANDARD2OF5_DECODING;
typedef enum tagStandard2of5Format
{
ITC_STANDARD2OF5_FORMAT_IDENTICON, // Default
ITC_STANDARD2OF5_FORMAT_COMPUTER_IDENTICS,
ITC_STANDARD2OF5_FORMAT_NO_CHANGE = 255
} ITC_STANDARD2OF5_FORMAT;
typedef enum tagStandard2of5CheckDigit
{
ITC_STANDARD2OF5_CHECK_NOTUSED, // Default
ITC_STANDARD2OF5_CHECK_XMIT,
ITC_STANDARD2OF5_CHECK_NOTXMIT,
ITC_STANDARD2OF5_CHECK_NO_CHANGE = 255
} ITC_STANDARD2OF5_CHECK_DIGIT;
typedef enum tagBarcodeLengthId
{
ITC_BARCODE_LENGTH = 0,
ITC_BARCODE_FIXED_LENGTH,
ITC_BARCODE_LENGTH_NO_CHANGE = 255
} ITC_BARCODE_LENGTH_ID;
```

IS9CConfig::GetTelepen

This function retrieves the current Telepen settings.

Syntax:

HRESULT **IS9CConfig::GetTelepen(**
ITC_TELEPEN_DECODING* *peDecode*,
ITC_TELEPEN_FORMAT* *peFormat* **);**

Parameters:

peDecode [out] Pointer to the ITC_TELEPEN_DECODING location to receive the decoding for TELEPEN symbology.

peFormat [out] Pointer to the ITC_TELEPEN_FORMAT location to receive the format.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetTelepen

This function updates the Telepen settings with new values.

Syntax:

HRESULT **IS9CConfig::SetTelepen(**
ITC_TELEPEN_DECODING* *eDecode*,
ITC_TELEPEN_FORMAT* *eFormat* **);**

Parameters:

eDecode [in] Identifies the decoding for Telepen symbology.

eFormat [in] Identifies the format.

Return Values:

HRESULT that indicates success or failure.

Telepen Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_TELEPEN_DECODING
Format	ASCII	ITC_TELEPEN_FORMAT

Telepen Enumerations

```
typedef enum tagTelepenDecoding
{
ITC_TELEPEN_NOTACTIVE = 0, // Default
ITC_TELEPEN_ACTIVE = 1,
ITC_TELEPEN_NO_CHANGE = 255
} ITC_TELEPEN_DECODING;
typedef enum tagTelepenDecoding
{
ITC_TELEPEN_FORMAT_ASCII, // Default
ITC_TELEPEN_FORMAT_NUMERIC,
ITC_TELEPEN_FORMAT_NO_CHANGE = 255
} ITC_TELEPEN_FORMAT;
```

IS9CConfig::GetUpcEan

This function retrieves the current UPC/EAN settings.

Syntax:

HRESULT **IS9CConfig::GetUpcEan(**
ITC_UPCEAN_DECODING* *upceanDecode*,
ITC_UPCA_SELECT* *upcASelect*, ITC_UPCE_SELECT* *upcESelect*,
ITC_EAN8_SELECT* *ean8Select*, ITC_EAN13_SELECT* *ean13Select*,
ITC_UPCEAN_ADDON_DIGITS* *upcAddOnDigits*,
ITC_UPCEAN_ADDON_TWO* *upcAddOn2*,
ITC_UPCEAN_ADDON_FIVE* *upcAddOn5*,
ITC_UPCA_CHECK_DIGIT* *upcACheck*,
ITC_UPCE_CHECK_DIGIT* *upcECheck*,
ITC_EAN8_CHECK_DIGIT* *ean8Check*,
ITC_EAN13_CHECK_DIGIT* *ean13Check*,
ITC_UPCA_NUMBER_SYSTEM* *upcANumSystem*,
ITC_UPCE_NUMBER_SYSTEM* *upcENumSystem*,
ITC_UPCA_REENCODE* *upcAReencode*,
ITC_UPCE_REENCODE* *upcEReencode*,
ITC_EAN8_REENCODE* *ean8Reencode* **);**

Parameters:

upceanDecode	[out]	Pointer to the ITC_UPCEAN_DECODING location to receive the decoding for UPC/EAN symbology.
upcASelect	[out]	Pointer to the ITC_UPCA_SELECT location to receive the UPC-A selection state.
upcESelect	[out]	Pointer to the ITC_UPCE_SELECT location to receive the UPC-E selection state.
ean8Select	[out]	Pointer to the ITC_EAN8_SELECT location to receive the EAN-8 selection state.
ean13Select	[out]	Pointer to the ITC_EAN13_SELECT location to receive the EAN-13 selection state.
upcAddOnDigits	[out]	Pointer to the ITC_UPCEAN_ADDON_DIGITS location to receive the add-on digits.
upcAddOn2	[out]	Pointer to the ITC_UPCEAN_ADDON_TWO location to receive the add-on 2 digits.
upcAddOn5	[out]	Pointer to the ITC_UPCEAN_ADDON_FIVE location to receive the add-on 5 digits.
upcACheck	[out]	Pointer to the ITC_UPCA_CHECK_DIGIT location to receive the UPC-A check digit.
upcECheck	[out]	Pointer to the ITC_UPCE_CHECK_DIGIT location to receive the UPC-E check digit.
ean8Check	[out]	Pointer to the ITC_EAN8_CHECK_DIGIT location to receive the EAN-8 check digit.
ean13Check	[out]	Pointer to the ITC_EAN13_CHECK_DIGIT location to receive the EAN-13 check digit.
upcANumSystem	[out]	Pointer to the ITC_UPCA_NUMBER_SYSTEM location to receive the UPC-A number system.

upcENumSystem	[out]	Pointer to the ITC_UPCE_NUMBER_SYSTEM location to receive the UPC-E number system.
upcAReencode	[out]	Pointer to the ITC_UPCA_REENCODE location to receive the UPC-A reencoding.
upcEReencode	[out]	Pointer to the ITC_UPCE_REENCODE location to receive the UPC-E reencoding.
ean8Reencode	[out]	Pointer to the ITC_EAN8_REENCODE location to receive the EAN-8 reencoding.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig::SetUpcEan

This function updates the UPC/EAN settings with new values.

Syntax:

HRESULT **IS9CConfig::SetUpcEan(**
ITC_UPCEAN_DECODING *upceanDecode*,
ITC_UPCA_SELECT *upcASelect*, ITC_UPCE_SELECT *upcESelect*,
ITC_EAN8_SELECT *ean8Select*, ITC_EAN13_SELECT *ean13Select*,
ITC_UPCEAN_ADDON_DIGITS *upcAddOnDigits*,
ITC_UPCEAN_ADDON_TWO *upcAddOn2*,
ITC_UPCEAN_ADDON_FIVE *upcAddOn5*,
ITC_UPCA_CHECK_DIGIT *upcACheck*,
ITC_UPCE_CHECK_DIGIT *upcECheck*,
ITC_EAN8_CHECK_DIGIT *ean8Check*,
ITC_EAN13_CHECK_DIGIT *ean13Check*,
ITC_UPCA_NUMBER_SYSTEM *upcANumSystem*,
ITC_UPCE_NUMBER_SYSTEM *upcENumSystem*,
ITC_UPCA_REENCODE *upcAReencode*,
ITC_UPCE_REENCODE *upcEReencode*,
ITC_EAN8_REENCODE *ean8Reencode* **);**

Parameters:

upceanDecode	[in]	Identifies the decoding for UPC/EAN symbology.
upcASelect	[in]	Identifies the UPC-A selection state.
upcESelect	[in]	Identifies the UPC-E selection state.
ean8Select	[in]	Identifies the EAN-8 selection state.
ean13Select	[in]	Identifies the EAN-13 selection state.
upcAddOnDigits	[in]	Identifies the Add-on digits.
upcAddOn2	[in]	Identifies the Add-on 2 digits.
upcAddOn5	[in]	Identifies the Add-on 5 digits.
upcACheck	[in]	Identifies the UPC-A check digit.
upcECheck	[in]	Identifies the UPC-E check digit.
ean8Check	[in]	Identifies the EAN-8 check digit.
ean13Check	[in]	Identifies the EAN-13 check digit.
upcANumSystem	[in]	Identifies the UPC-A number system.
upcENumSystem	[in]	Identifies the UPC-E number system.
upcAReencode	[in]	Identifies the UPC-A reencoding.
upcEReencode	[in]	Identifies the UPC-E reencoding.
ean8Reencode	[in]	Identifies the EAN-8 reencoding.

Return Values:
HRESULT that indicates success or failure.

UPC/EAN Default Settings

Parameter	Default	Valid Range
Decoding	ITC_UPCEAN_NO_CHANGE	This parameter is no longer used and must be set to this value.
UPC-A	Active	ITC_UPCA_SELECT
UPC-E	Active	ITC_UPCE_SELECT
EAN-8	Active	ITC_EAN8_SELECT
EAN-13	Active	ITC_EAN13_SELECT
Add On Digits	Not Required	ITC_UPCEAN_ADDON_DIGITS
Add On 2 Digits	Not Active	ITC_UPCEAN_ADDON_TWO
Add On 5 Digits	Not Active	ITC_UPCEAN_ADDON_FIVE
UPC-A Check Digit	Transmitted	ITC_UPCA_CHECK_DIGIT
UPC-E Check Digit	Transmitted	ITC_UPCE_CHECK_DIGIT
EAN-8 Check Digit	Transmitted	ITC_EAN8_CHECK_DIGIT
EAN-13 Check Digit	Transmitted	ITC_EAN13_CHECK_DIGIT
UPC-A Number System	Transmitted	ITC_UPCA_NUMBER_SYSTEM
UPC-E Number System	Transmitted	ITC_UPCE_NUMBER_SYSTEM
Reencode UPC-A	UPC-A transmitted as EAN-13	ITC_UPCA_REENCODE
Reencode UPC-E	UPC-E transmitted as UPC-E	ITC_UPCE_REENCODE
Reencode EAN-8	EAN-8 transmitted as EAN-8	ITC_EAN8_REENCODE

UPC/EAN Enumerations

```
typedef enum tagUpcEanDecoding
{
ITC_UPCEAN_NOTACTIVE = 0,
ITC_UPCEAN_ACTIVE = 1,                  // Default
ITC_UPCEAN_NO_CHANGE = 255
} ITC_UPCEAN_DECODING;
typedef enum tagUpcASelect
{
ITC_UPCA_DEACTIVATE,
ITC_UPCA_ACTIVATE,                      // Default
ITC_UPCA_NO_CHANGE = 255
} ITC_UPCA_SELECT;
typedef enum tagUpcESelect
{
ITC_UPCE_DEACTIVATE,
ITC_UPCE_ACTIVATE,                      // Default
ITC_UPCE_NO_CHANGE = 255
} ITC_UPCE_SELECT;
typedef enum tagEan8Select
{
ITC_EAN8_DEACTIVATE,
ITC_EAN8_ACTIVATE,                      // Default
ITC_EAN8_NO_CHANGE = 255
} ITC_EAN8_SELECT;
typedef enum tagEan13Select
{
ITC_EAN13_DEACTIVATE,
ITC_EAN13_ACTIVATE,                     // Default
ITC_EAN13_NO_CHANGE = 255
} ITC_EAN13_SELECT;
typedef enum tagUpcEanAddonDigits
{
ITC_UPCEAN_ADDON_NOT_REQUIRED,          // Default
ITC_UPCEAN_ADDON_REQUIRED,
ITC_UPCEAN_ADDON_NO_CHANGE = 255
```

```
} ITC_UPCEAN_ADDON_DIGITS;
typedef enum tagUpcEanAddonTwo
{
ITC_UPCEAN_ADDON_TWO_NOTACTIVE = 0,     // Default
ITC_UPCEAN_ADDON_TWO_ACTIVE = 1,
ITC_UPCEAN_ADDON_TWO_NO_CHANGE = 255
} ITC_UPCEAN_ADDON_TWO;
typedef enum tagUpcEanAddonFive
{
ITC_UPCEAN_ADDON_FIVE_NOTACTIVE = 0,    // Default
ITC_UPCEAN_ADDON_FIVE_ACTIVE = 1,
ITC_UPCEAN_ADDON_FIVE_NO_CHANGE = 255
} ITC_UPCEAN_ADDON_FIVE;
typedef enum tagUpcACheckDigit
{
ITC_UPCA_CHECK_NOTXMIT = 0,
ITC_UPCA_CHECK_XMIT = 1,                // Default
ITC_UPCA_CHECK_NO_CHANGE = 255
} ITC_UPCA_CHECK_DIGIT;
typedef enum tagUpcECheckDigit
{
ITC_UPCE_CHECK_NOTXMIT = 0,
ITC_UPCE_CHECK_XMIT = 1,                // Default
ITC_UPCE_CHECK_NO_CHANGE = 255
} ITC_UPCE_CHECK_DIGIT;
typedef enum tagEan8CheckDigit
{
ITC_EAN8_CHECK_NOTXMIT = 0,
ITC_EAN8_CHECK_XMIT = 1,                // Default
ITC_EAN8_CHECK_NO_CHANGE = 255
} ITC_EAN8_CHECK_DIGIT;
typedef enum tagEan13CheckDigit
{
ITC_EAN13_CHECK_NOTXMIT = 0,
ITC_EAN13_CHECK_XMIT = 1,               // Default
ITC_EAN13_CHECK_NO_CHANGE = 255
} ITC_EAN13_CHECK_DIGIT;
typedef enum tagUpcANumberSystem
{
ITC_UPCA_NUM_SYS_NOTXMIT = 0,
ITC_UPCA_NUM_SYS_XMIT = 1,              // Default
ITC_UPCA_NUM_SYS_NO_CHANGE = 255
} ITC_UPCA_NUMBER_SYSTEM;
typedef enum tagUpcENumberSystem
{
ITC_UPCE_NUM_SYS_NOTXMIT = 0,
ITC_UPCE_NUM_SYS_XMIT = 1,              // Default
ITC_UPCE_NUM_SYS_NO_CHANGE = 255
} ITC_UPCE_NUMBER_SYSTEM;
typedef enum tagUpcAReencode
{
ITC_UPCA_XMIT_AS_EAN13,                 // Default
ITC_UPCA_XMIT_AS_UPCA,
ITC_UPCA_XMIT_NO_CHANGE = 255
} ITC_UPCA_REENCODE;
typedef enum tagUpcEReencode
{
ITC_UPCE_XMIT_AS_UPCE,                  // Default
ITC_UPCE_XMIT_AS_UPCA,
ITC_UPCE_XMIT_NO_CHANGE = 255
} ITC_UPCE_REENCODE;
typedef enum tagEan8Reencode
{
ITC_EAN8_XMIT_AS_EAN8,                  //Default
ITC_EAN8_XMIT_AS_EAN13,
ITC_EAN8_XMIT_NO_CHANGE = 255
} ITC_EAN8_REENCODE;
```

IS9CConfig2 Functions

This interface is derived from the IS9CConfig interface and provides additional methods that can be used to set and retrieve the 700 Series Computer's bar code configuration. All supported symbologies are initialized to their defaults when the S9C firmware is loaded.

GET/SET functions use enumerations as their parameters. In most enumerations, there is an enumerator xx_NO_CHANGE (such as ITC_CODE39_NO_CHANGE), where xx refers to a particular enumeration. This enumerator can be used during a call to a SET to indicate that no change is to be made to that particular parameter. This prevents the called function from having to format the same S9C command and send it down to the scanner.

To specify a bar code length of "any length," use a value of "0" for the bar code length argument.

IS9CConfig2 functions are the following. IS9CCONFIG.H is the header file and ITCUUID.LIB contains the IID_IADC Interface GUID value used to obtain the interface.

IS9CConfig2::GetCode11

This function retrieves the current settings for Code 11.

Syntax:

HRESULT **GetCode11(** ITC_CODE11_DECODING* *peDecode*,
ITC_CODE11_CHECK_DIGIT* *peCheck*,
ITC_CODE11_CHECK_VERIFICATION* *peVer* **);**

Parameters:

peDecode	[out]	Pointer to ITC_CODE11_DECODING location to receive Code 11 decoding.
peCheck	[out]	Pointer to ITC_CODE11_CHECK_DIGIT location to receive the check digit option.
peVer	[out]	Pointer to ITC_CODE11_CHECK_VERIFICATION location to receive the check verification option.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig2::SetCode11

This function updates the current setting of Code 11 symbology.

Syntax:

HRESULT **SetCode11(** ITC_CODE11_DECODING *eDecode*,
ITC_CODE11_CHECK_DIGIT *eCheck*,
ITC_CODE11_CHECK_VERIFICATION *eVer* **);**

Parameters:

eDecode	[in]	An enumeration that identifies decoding option for Code 11.
eCheck	[in]	An enumeration that identifies the check digit option.
eVer	[in]	An enumeration that identifies check verification option.

Return Values:

HRESULT that indicates success or failure.

Code 11 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_CODE11_DECODING
Check Verification	1 Digit	ITC_CODE11_CHECK_VERIFICATION
Check Digit	Enable	ITC_CODE11_CHECK_DIGIT

Code 11 Enumerations

```
typedef enum tagCode11Decoding
{
ITC_CODE11_NOTACTIVE = 0,
ITC_CODE11_ACTIVE = 1, // Default
ITC_CODE11_NO_CHANGE = 255
} ITC_CODE11_DECODING;
typedef enum tagCode11CheckVerification
{
ITC_CODE11_CHK_VERIFY_ONEDIGIT = 1,
ITC_CODE11_CHK_VERIFY_TWODIGIT = 2, // Default
ITC_CODE11_CHK_VERIFY_NO_CHANGE = 255
} ITC_CODE11_CHECK_VERIFICATION;
typedef enum tagCode11CheckDigit
{
ITC_CODE11_CHECK_NOTXMIT = 0, // Default
ITC_CODE11_CHECK_XMIT = 1,
ITC_CODE11_CHECK_NO_CHANGE = 255
} ITC_CODE11_CHECK_DIGIT;
```

IS9CConfig2::GetCustomSymIds

This function retrieves all the custom symbology identifiers defined for the currently supported symbologies. *This is not supported when using an imager.*

Syntax:

HRESULT **GetCustomSymIds(**
ITC_CUST_SYM_ID_PAIR* *pStructSymIdPair*,
DWORD *dwMaxNumElement*, DWORD* *pdwNumElement* **);**

Parameters:

pStructSymIdPair	[out]	Pointer to ITC_CUST_SYM_ID_PAIR location to receive the current defined symbology identifiers for the supported symbologies. The caller must preallocate this buffer with *dwMaxNumElement* elements.
dwMaxNumElement	[in]	Maximum number of elements allocated for the *pStructSymIdPair* buffer which should always be equal to the last defined enumeration constant + 1 of the enumeration ITC_CUSTOM_ID. In this case, it is ITC_CUSTOMID_LAST_ELEMENT.
pdwNumElement	[out]	Pointer to DWORD location to receive the actual number of elements returned in the *pStructSymIdPair* buffer, which should be the same as *dwMaxNumElement*.

Return Values:

HRESULT that indicates success or failure.

See Also:

Custom Identifier Assignments *(page 6-57)*
Custom Identifier Example *(page 6-58)*
Custom Identifier Default Settings *(page 6-57)*

IS9CConfig2::SetCustomSymIds

This function updates the symbology identifiers (any ASCII values) for the currently supported symbologies. *This is not supported when using an imager.*

Syntax:

HRESULT **SetCustomSymIds(**
ITC_CUST_SYM_ID_PAIR* *pStructSymIdPair*, DWORD *dwNumElement*
);

Parameters:

pStructSymIdPair	[in]	Pointer to ITC_CUST_SYM_ID_PAIR location, containing the new symbology identifiers for any supported symbologies to update.
dwNumElement	[in]	Identifies the number of symbology identifiers to update in the *pStructSymIdPair* buffer.

Return Values:

HRESULT that indicates success or failure.

Custom Identifier Assignments

Each custom identifier is a one byte ASCII value within the range from 0x00 to 0xff. The enumerations in the ITC_CUSTOM_ID enumerator can be used as symbology identifications in the GetCustomSymIds and SetCustomSymIds functions.

```
typedef enum tagCustomId
{
ITC_CUSTOMID_CODABAR = 0        Identifies the Codabar symbology
ITC_CUSTOMID_CODE39             Identifies the Code 39 symbology
ITC_CUSTOMID_CODE93             Identifies the Code 93 symbology
ITC_CUSTOMID_CODE128_EAN_128    Identifies the Code 128 symbology
ITC_CUSTOMID_EAN8               Identifies the EAN-8 symbology
ITC_CUSTOMID_EAN13              Identifies the EAN-13 symbology
ITC_CUSTOMID_I2OF5              Identifies the Interleaved 2 of 5 symbology
ITC_CUSTOMID_MATRIX2OF5         Identifies the Matrix 2 of 5 symbology
ITC_CUSTOMID_MSI                Identifies the MSI symbology
ITC_CUSTOMID_PDF417             Identifies the PDF 417 symbology
ITC_CUSTOMID_PLESSEY            Identifies the Plessey symbology
ITC_CUSTOMID_CODE2OF5           Identifies the Standard 2 of 5 symbology
ITC_CUSTOMID_TELEPEN            Identifies the Telepen symbology
ITC_CUSTOMID_UPCA               Identifies the UPC-A symbology
ITC_CUSTOMID_UPCE               Identifies the UPC-E symbology
ITC_CUSTOMID_CODE11             Identifies the Code 11 symbology
ITC_CUSTOMID_LAST_ELEMENT       Identifies the last element.  Use to preallocate the
                                   buffer on GetCustomSymIds
}ITC_CUSTOM_ID;
typedef struct tagCustSymbIdPair
{

ITC_CUSTOM_ID eSymbology;       Identifies the symbology of interest

BYTE byteId;
 ASCII value (1 byte within the range0x00 – 0xf)

}ITC_CUST_SYM_ID_PAIR;
```

Custom Identifier Default Settings

Symbology	Default	Valid Range
Codabar	D	0x00–0xFF
Code 11	*	0x00–0xFF
Code 39	*	0x00–0xFF
Code 93	D	0x00–0xFF
Code128/EAN 128	D	0x00–0xFF
EAN-8	0xFF	0x00–0xFF
EAN-13	F	0x00–0xFF
Interleaved 2 of 5	I	0x00–0xFF
Matrix 2 of 5	D	0x00–0xFF
MSI	D	0x00–0xFF
PDF 417	*	0x00–0xFF
Plessey	D	0x00–0xFF
Standard 2 of 5	D	0x00–0xFF
Telepen	*	0x00–0xFF
UPC-A	A	0x00–0xFF
UPC-E	E	0x00–0xFF

Custom Identifier Example

The following code segment is an example of updating the UPC-E and UPC-A symbology identifiers with new values, and then retrieving the currently defined symbology identifiers for all the supported symbologies:

```
ITC_CUST_SYM_ID_PAIR oStructSymIdPair [ITC_CUSTOMID_LAST_ELEMENT];
oStructSymIdPair[0].eSymbology = ITC_CUSTOMID_UPCE;
oStructSymIdPair[0].byteId = 0x41;                          // ASCII char A
oStructSymIdPair[1].eSymbology = ITC_CUSTOMID_UPCA;
oStructSymIdPair[1].byteId = 0x42;                          // ASCII char B
HRESULT hr = pIS9CConfig2->SetCustomSymIds(&oStructSymIdPair[0], 2];
DWORD dwNum = 0;
HRESULT hr = pIS9CConfig2->GetCustomSymIds(&oStructSymIdPair[0],
ITC_CUSTOMID_LAST_ELEMENT, &dwNum);
```

IS9CConfig2::GetGlobalAmble

This function retrieves the scanner's current preamble or postamble setting.

Syntax:

HRESULT **GetGlobalAmble(** ITC_GLOBAL_AMBLE_ID *eAmbleId*, BYTE *rgbBuffer[]*, DWORD *dwBufferSize*, DWORD* *pdwBufferSize* **);**

Parameters:

eAmbleId	[in]	An enumeration of type ITC_GLOBAL_AMBLE_ID identifies whether the preamble or postamble setting is to be retrieved. Only one setting can be queried at a time.
rgbBuffer	[in]	Contains the buffer for the postamble or preamble setting to be queried.
dwBufferSize	[in]	The maximum number of bytes that rgbBuffer can store. Must be at least ITC_GLOBAL_AMBLE_MAX_CHARS bytes.
pdwBufferSize	[out]	A pointer to DWORD location to store the actual number of returned bytes in *rgbBuffer*.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig2::SetGlobalAmble

This function updates the scanner's current preamble or postamble setting depending on the input parameters.

Syntax:

HRESULT **SetGlobalAmble(** ITC_GLOBAL_AMBLE_ID *eAmbleId*, BYTE *rgbBuffer[]*, DWORD *dwBufferSize* **);**

Parameters:

eAmbleId	[in]	An enumeration of type ITC_GLOBAL_AMBLE_ID identifies whether the preamble or postamble setting is to be updated. Only one setting can be updated at a time.
rgbBuffer	[in]	Contains the buffer for the postamble or preamble setting to be updated.
dwBufferSize	[in]	Identifies number of bytes in *rgbBuffer*.

Return Values:

HRESULT that indicates success or failure.

Postamble and Preamble Defaults

Parameter	Default	Valid Range
Preamble	Null	0 to 20 ASCII characters
Postamble	Null	0 to 20 ASCII characters

IS9CConfig2::GetPDF417Ext

This function is an extended function for retrieving the PDF 417 settings not included in the IS9CConfig::GetPDF417.

Syntax:

HRESULT **GetPDF417Ext(**
ITC_MICRO_PDF417_DECODING* *peDecode*,
ITC_MICRO_PDF417_CODE128_EMULATION* *peCode128* **);**

Parameters:

peDecode [out] Pointer to ITC_MICRO_PDF417_DECODING location to receive the Micro PDF 417 decoding.

peCode128 [out] Pointer to ITC_MICRO_PDF417_CODE128_EMULATION* location to receive the Micro PDF 417 Code 128 emulation option.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig2::SetPDF417Ext

This function is an extended function for updating the additional PDF 417 settings not included in IS9CConfig::SetPDF417.

Syntax:

HRESULT **SetPDF417Ext(** ITC_MICRO_PDF417_DECODING *eDecode*,
ITC_MICRO_PDF417_CODE128_EMULATION *eCode128* **);**

Parameters:

eDecode [in] An enumeration that identifies decoding option for the Micro PDF 417.

eCode128 [in] An enumeration that identifies the Code 128 emulation option for the Micro PDF 417.

Return Values:

HRESULT that indicates success or failure.

PDF 417 Extended: Micro PDF 417 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_MICRO_PDF417_DECODING
Code 128 Emulation	Not Active	ITC_MICRO_PDF417_CODE128_EMULATION

** These are Micro PDF 417 parameters.*

IS9CConfig2::GetSymIdXmit

This function retrieves the current symbology ID transmission option as described on the next page.

Syntax:

HRESULT **GetSymIdXmit(** ITC_SYMBOLOGY_ID_XMIT* *peSymIdXmit* **);**

Parameters:

peSymIdXmit	[out]	Pointer to ITC_SYMBOLOGY_ID_XMIT location to receive the current symbology identifier transmission option.

Return Values:

HRESULT that indicates success or failure.

IS9CConfig2::SetSymIdXmit

This function updates the symbology ID transmission option as shown on the next page.

Syntax:

HRESULT **SetSymIdXmit(** ITC_SYMBOLOGY_ID_XMIT *eSymIdXmit* **);**

Parameters:

eSymIdXmit	[in]	Identifies the symbology identifier transmission option to update.

Return Values:

HRESULT that indicates success or failure.

Symbology ID Transmission Option

The symbology identifier (or code mark) concept provides a standardized way for a device receiving data from a bar code reader to differentiate between the symbologies.

The following symbology ID transmission option specifies whether or not the symbology ID should be transmitted as part of the scanned bar code label to all the connected data collection applications. Options for transmission are: do not transmit, transmit the standard AIM identifiers, or transmit the one byte custom defined identifiers. AIM and custom identifiers cannot be selected to be transmitted at the same time; only the last selected option will be active.

```
typedef enum tagSymbologyIdXmit

{
```

`ITC_ID_XMIT_DISABLE = 0`	Symbology identifier will not be transmitted as part of the label. This is the default setting.
`ITC_ID_XMIT_CUSTOM = 1`	Activate custom symbology identifier transmission for all symbologies. Example of the transmitted label: [preamble] [Custom ID] <data> [postamble]
`ITC_ID_XMIT_AIM = 2`	Activate AIM symbology identifier transmission for all symbologies. Example of the transmitted label: [preamble] [AIM symbology ID] <data> [postamble]

```
}ITC_SYMBOLOGY_ID_XMIT;
```

IS9CConfig3 Functions

The IS9CConfig3 interface provides generic methods for retrieving and setting configuration using ISCP commands.

ISCP Commands

An ISCP Command is composed of three or more bytes formatted as <SG><FID><parameters> where:

SG	Setup group.
FID	Function ID.
parameters	One or more configuration value bytes depending on the configuration.

ISCP commands include the following:

Imager Settings

This dictates the start and end column positions for the image dimension.

SG	FID	Parameter	Description
0x7B	80	Value [0..639]	Start column position.
0x7B	81	Value [0..639]	End column position.

Trigger Settings

This sets the duration of the aiming beam before acquiring images to be decoded.

SG	FID	Parameter	Description
0x70	81	Value [0..65535]	Number of milliseconds.

QRCode Symbology

This enables or disables the QRCode symbology.

SG	FID	Parameter	Description
0x55	40	0	Disable this symbology.
0x55	40	1	Enable this symbology.

Datamatrix Symbology

This enables or disables the Datamatrix symbology.

SG	FID	Parameter	Description
0x54	40	0	Disable this symbology.
0x54	40	1	Enable this symbology.

ISCP::GetConfig

This retrieves configurations using the ISCP commands format.

Syntax:

HRESULT **ISCPGetConfig(** BYTE *rgbCommandBuff[*],
DWORD *dwCommandBuffSize*, BYTE *rgbReplyBuff[],*
DWORD *dwReplyBuffMaxSize*, DWORD **pdwReplyBuffSize* **);**

Parameters:

rgbCommandBuff	[in, size_is]	Contains ISCP commands in array of bytes.
dwCommandBuffSize	[in]	Number of bytes in *rgbCommandBuff*.
rgbReplyBuff	[in, out, size_is]	Results of query in array of bytes.
dwReplyBuffMaxSize	[in]	Maximum size of *rgdReplyBuff*.
pdwReplyBuffSize	[in, out]	Number of bytes placed in *rbfReplyBuff*.

ISCP::SetConfig

This updates configurations using the ISCP commands format.

Syntax:

HRESULT **ISCPSetConfig(** BYTE *rgbCommandBuff[],*
DWORD *dwCommandBuffSize*, BYTE *rgbReplyBuff[],*
DWORD *dwReplyBuffMaxSize*, DWORD **pdwReplyBuffSize* **);**

Parameters:

rgbCommandBuff	[in, size_is]	Contains ISCP commands in array of bytes.
dwCommandBuffSize	[in]	Number of bytes in *rgbCommandBuff*.
rgbReplyBuff	[in, out, size_is]	Results of request in array of bytes.
dwReplyBuffMaxSize	[in]	Maximum size of *rgbReplyBuff*.
pdwReplyBuffSize	[in, out]	Number of bytes placed in *rgbReplyBuff*.

AIM Symbology ID Defaults

Refer to the official AIM documentation on symbology identifiers for full information on the different processing options supported.

Bar Code Symbology	ID Character	Modifier Characters
Codabar	F	0 Standard Codabar symbol. No special processing. 1 ABC Codabar (American Blood commission) concatenate/message append performed. 2 Reader has validated the check character. 4 Reader has stripped the check character before transmission.
Code 11	H	0 Single modulo 11 check character validated and transmitted. 1 Two modulo 11 check characters validated and transmitted. 3 Check characters validated but not transmitted.
Code 39	A	0 No check character validation nor full ASCII processing. All data transmitted as decoded. 1 Modulo 43 check character validated and transmitted. 3 Modulo 43 check character validated but not transmitted. 4 Full ASCII character conversion performed. No check character validation. 5 Full ASCII character conversion performed. Modulo 43 check character validated and transmitted. 7 Full ASCII character conversion performed. Modulo 43 check character validated but not transmitted.
Code 93	G	0 No options specified. Always transmit 0.
Code 128	C	0 Standard data packet. No FNC1 in first or second symbol character position after start character. 1 EAN/UCC-128 data packet. FNC1 in first symbol character position after start character. 2 FNC1 in second symbol character position after start character. 4 Concatenation according to International Society for Blood Transfusion specifications was performed. Concatenated data follows.
Interleaved 2 of 5	I	0 No check character validation. 1 Modulo 10 symbol check character validated and transmitted 3 Modulo 10 symbol check character validated but not transmitted.
Matrix 2 of 5	X	0–F For symbologies or symbology options not listed, a code character with the value 0–F may be assigned by the decoder manufacturer to identify those symbologies and options implemented in the reader.
MSI	M	0 Modulo 10 symbol check character validated and transmitted. 1 Modulo 10 symbol check character validated but not transmitted.
PDF 417/ Micro PDF 417	L	0 Reader set to conform with protocol defined in 1994 PDF 417 specifications. 1 Reader set to follow protocol of ENV 12925 for Extended Channel Interpretation (all data characters 92 doubled). 2 Reader set to follow protocol of ENV 12925 for Basic Channel Interpretation (data characters 92 are not doubled). 3 Code 128 emulation: implied FNC1 in first position. 4 Code 128 emulation: implied FNC1 after initial letter or pair of digits. 5 Code 128 emulation: no implied FNC1.

Bar Code Symbology	ID Character	Modifier Characters *(Continued)*
Plessey	P	0 No options specified. Always transmit 0.
Standard 2 of 5 (2-bar start/stop)	R	0 No check character validation. 1 Modulo 7 check character validated and transmitted. 3 Modulo 7 check character validated but not transmitted.
Standard 2 of 5 (3-bar start/stop)	S	0 No options specified. Always transmit 0.
Telepen	B	0 Full ASCII mode 1 Double density numeric only mode 2 Double density numeric followed by full ASCII 4 Full ASCII followed by double density numeric
UPC/EAN	E	Consider UPC/EAN symbols with supplements as two separate symbols. The first symbol is the main data packet, and the second symbol is the 2 or 5 digit supplement. Transmit these two symbols separately, each with its own symbology identifier. Provision is made for the option of transmitting both symbols as a single data packet. 0 Standard data packet in full EAN format (13 digits for EAN-13, UPC-A, and UPC-E; does not include add-on data). 1 Two digit add-on data only. 2 Five digit add-on data only. 3 Combined data packet comprising 13 digits from EAN-13, UPC-A, or UPC-E symbol and 2 or 5 digits from add-on symbol. 4 EAN-8 data packet

***IMPORTANT**: The "symbology_id" character letter **must** be uppercase for the above definitions.*

IImage Interface

The IImage interface gives the application the capability to acquire images. The image acquired can be either a raw image as captured by the digital camera or it can be normalized. A normalized image is presented the same as if the picture were taken at right angles to the image and at the same distance. The normalized image is commonly used for signature capture applications.

IImage::ReadSigCapBuffer

Syntax:

HRESULT **IImage::ReadSigCapBuffer(**
ITC_SIGCAP_SPEC **pSigCapSpec*, ITC_IMAGE_SPEC **pImgBuffer*,
DWORD *nMaxBuffSize* **);**

Parameters:

pSigCapSpec [in] Pointer to the structure that identifies the signature capture region. This structure is defined as follows:

```
typedef struct tagITCSigCapSpec
{
DWORD dwStructSize;
INT  iAspectRatio;
INT  iOffsetX;
INT  iOffsetY;
UINT uiWidth;
UINT uiHeight;
INT  iResolution;
ITCFileFormat eFormat;
DWORD eDepth;
} ITC_SIGCAP_SPEC;
```

where:

dwStructSize	Size, in bytes, of this struct. This is for version control.
iAspectRatio	Ratio of the bar code height (linear bar codes) or row height (2D bar codes) to the narrow element width.
iOffsetX	Offset in X direction, relative to barcode center. Positive values are right of the bar code, negative values to the left.
iOffsetY	Offset in Y direction, relative to barcode center. Positive values are higher than the bar code, negative values lower.
uiWidth	Width of signature capture image region in intelligent bar code units.
uiHeight	Height of the signature capture image region in intelligent bar code units.
iResolution	Number of pixels per intelligent bar code unit.

eFormat Format of the image buffer returned as follows. Currently, only ITC_FILE_RAW is supported.

```
ITC_FILE_KIM =              0,     // Returns data a KIM file
ITC_FILE_TIFF_BIN =         1,     // TIFF Binary file
ITC_FILE_TIFF_BIN_GROUP4 =  2,     // TIFF Binary Group 4 compressed
ITC_FILE_TIFF_GRAY_SCALE =  3,     // TIFF Gray Scale
ITC_FILE_RAW =              4,     // Raw image
ITC_FILE_JPEG =             5,     // JPEG image
```

eDepth Number of bits per pixel. Currently, only one (monochrome) or eight (gray-scale) are supported.

pImgBuffer [out] Pointer to the buffer in which the signature capture image will be put.

```
typedef struct tagITCImageSpec
{
   DWORD  dwStructSize;
   LONG   biWidth;
   LONG   biHeight;
   WORD   biBitCount;
   ITC_FILE_FORMAT eFormat;
   DWORD  biActualImageSize;
   DWORD  biMaxImageBytes;
   BYTE   rgbImageData[1];
} ITC_IMAGE_SPEC;
```

where:

dwStructSize Size, in bytes, of this struct. This is for version control.

biWidth The width of each row in pixels.

biHeight The number of rows in the image data.

biBitCount The number of bits per pixel.

eFormat Identifies the image format.

biActualImageSize Total bytes of image data returned.

biMaxImageBytes Maximum bytes that can be stored in *rgbImageData[]*.

rgbImageData Buffer containing the actual data, for example a 640x480 uses a 307200-byte buffer. The array size of this buffer is arbitrary so do *not* use this structure directly to reserve memory. The actual dimension of the buffer is identified by *biMaxImageBytes*.

Returns:

HRESULT identifying success or error. On error, the following codes will be returned:

S_OK
Image successfully returned.

ITC_RESULT_ERR_BADREGION_E
The specified region is not in the image.

ITC_RESULT_NO_BC_DECODED_E
A bar code has not yet been decoded or the last bar code decoded was not a signature capture symbology.

ITC_IMGBUFF_TOO_SMALL_E
pImgBuffer is too small to contain the signature captured image.

ITC_INV_PARAMETER_E
One of the parameters is invalid.

S_DEVICE_NOT_OPENED_E
The device had not been opened.

Remarks:

ReadSigCapBuffer() will return the image from the last decoded label with dimensions identified by the calling parameter. This signature capture region must include the signature capture bar code. The supported bar codes for signature capture are: PDF 417, Code 128, and Code 39. The caller specifies the width, height, and center of the image to be retrieved. This image is independent of any rotation of the bar code relative to the imager. Thus, if the bar code is decoded with the code itself upside down to the imager, the retrieved image will still be right side up. However, if the specified image is outside the field of view a result code of ITC_RESULT_ERR_BADREGION_E will be returned.

This function uses the dimensions of the last decoded bar code as its coordinate system. Thus, all the parameters describing the image size and position are in units called "Intelligent Bar Code Units." An Intelligent Bar Code Unit is equivalent to the narrow element width of the bar code.

The dimensions of the resulting image can be calculated using this formula:

```
Resulting Width = Specified Width * Specified Resolution
Resulting Height = Specified Height * Specified Resolution
```

IImage::ReadSigCapFile

▶ **NOTE:** *This has not been implemented as of this publication.*

Syntax:

HRESULT **IImage::ReadSigCapFile(**
ITC_SIGCAP_SPEC **pSigCapSpec*, LPCTSTR *pszFileName* **);**

Parameters:

pSigCapSpec [in] Pointer to the structure that identifies the signature capture region. See ReadSigCapFile (page 6-65) for a description of this structure.

pszFileName [in] Name of the file in which to copy the image.

Returns:

HRESULT identifying success or error. On error, the following codes will be returned:

S_OK
Image successfully returned.

ITC_RESULT_ERR_BADREGION_E
The specified region is not in the image.

ITC_RESULT_NO_BC_DECODED_E
A bar code has not yet been decoded or the last bar code decoded was not a signature capture symbology.

ITC_FILE_OPEN_E
The file could not be opened.

ITC_INV_PARAMETER_E
One of the parameters is invalid.

S_DEVICE_NOT_OPENED_E
The device had not been opened.

Remarks:

ReadSigCapFile() will write the image from the last decoded label with dimensions identified by the calling parameter. If the file already exists, its contents will be overwritten.

This signature capture region must include the signature capture bar code. The supported bar codes for signature capture are: PDF 417, Code 128, and Code 39. The caller specifies the width, height, and center of the image to be retrieved. This image is independent of any rotation of the bar code relative to the imager. Thus, if the bar code is decoded with the code itself upside down to the imager, the retrieved image will still be right side up. However, if the specified image is outside the field of view a result code of ITC_RESULT_ERR_BADREGION_E will be returned.

This function uses the dimensions of the last decoded bar code as its coordinate system. Thus, all the parameters describing the image size and position are in units called "Intelligent Bar Code Units". An Intelligent Bar Code Unit is equivalent to the narrow element width of the bar code.

The dimensions of the resulting image can be calculated using this formula:

```
Resulting Width = Specified Width * Specified Resolution
Resulting Height = Specified Height * Specified Resolution
```

IImage::ReadImage

Syntax:

HRESULT **IImage::Read(** ITCFileFormat *eFormat*, DWORD *nDepth*, ITC_IMAGE_SPEC **pImgBuffer*, DWORD *dwTimeout* **);**

Parameters:

eFormat [in] Format of the image buffer returned as follows. Currently, only ITC_FILE_RAW is supported.

```
ITC_FILE_KIM =                0,     // Returns data a KIM file
ITC_FILE_TIFF_BIN =           1,     // TIFF Binary file
ITC_FILE_TIFF_BIN_GROUP4 =    2,     // TIFF Binary Group 4 compressed
ITC_FILE_TIFF_GRAY_SCALE =    3,     // TIFF Gray Scale
ITC_FILE_RAW =                4,     // Raw image
ITC_FILE_JPEG =               5,     // JPEG image
```

nDepth [in] Number of bits per pixel. Currently, only eight (gray-scale) is supported.

pImgBuffer [in/out] Pointer to the buffer containing the image.

dwTimeout [in] Milliseconds to wait for the image to be returned.

Returns:

HRESULT identifying success or error. On error, the following codes will be returned:

S_OK
Image successfully returned.

ITC_IMGBUFF_TOO_SMALL_E
pImgBuffer is too small to contain the signature captured image.

ITC_TIMEOUT_E
Timeout.

ITC_INV_PARAMETER_E
One of the parameters is invalid.

S_DEVICE_NOT_OPENED_E
The device had not been opened.

Remarks:

The image is returned in *pImgBuffer* in the caller specified format.

IImage::CancelReadImage

Syntax:
HRESULT **IImage::CancelReadImage( );**

Parameters:
None.

Returns:
Status code indicating success or failure as follows:

S_OK
Imager closed.

S_DEVICE_NOT_OPENED_E
The device had not been opened.

Remarks:
This function causes a pending image read of IImage::ReadImage() to return immediately with an error status. The purpose of this function is to allow the application to release a thread blocked on the ReadImage() call.

IImage::Start

Syntax:
HRESULT **IImage::Start( );**

Parameters:
None.

Returns:
Status code indicating success or failure as follows:

S_OK
Imager started.

S_DEVICE_NOT_OPENED_E
The device had not been opened.

Remarks:
This function starts the image continuously capturing images.

IImage::Stop

Syntax:
HRESULT **IImage::Stop( );**

Parameters:
None.

Returns:
Status code indicating success or failure as follows:

S_OK
Imager started.

S_IMG_NOT_PRESENT_E
The unit does not contain an imager.

S_DEVICE_NOT_OPENED_E
The device had not been opened.

Remarks:
This function stops the image continuously capturing images.

IImage::Open

Syntax:

HRESULT **IImage::Open(** BOOL *fSigCapEnable* **);**

Parameters:

fSigCapEnable [in] When TRUE, signature capture capability is enabled. When FALSE, it is disabled. In either mode, bar code labels are decoded and images (via IImage::ReadImage) the same.

Returns:

Status code indicating success or failure as follows:

S_OK
Imager opened.

S_IMG_NOT_PRESENT_E
The unit does not contain an imager.

S_DEVICE_CONTENTION_E
The device has already been opened.

Remarks:

This function exclusively allocates the imager device so that the other IImage methods can be safely called.

IImage::Close

Syntax:

HRESULT **IImage::Close( );**

Parameters:

None.

Returns:

Status code indicating success or failure as follows:

S_OK
Imager closed.

S_DEVICE_NOT_OPENED_E
The device had not been opened.

Remarks:

This function releases the imager device so that other applications can open it. An IImage::Release() will also close the imager device.

Data Collection Configuration

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 → the **Data Collection** icon. See *Appendix A*, "*Control Panel Applets*" for more information about the following parameters. *Note that these are in alphabetical order.*

- Codabar *(page A-7)*
- Code 11 *(page A-21)*
- Code 128 *(page A-10)*
  - Code 128 Options *(page A-11)*
  - Code 128 FNC1 Character *(page A-12)*
- Code 39 *(page A-5)*
- Code 93 *(page A-9)*
  - Code 93 Length *(page A-9)*
- Datamatrix *(page A-23)*
- Interleaved 2 of 5 *(page A-18)*
- Matrix 2 of 5 *(page A-19)*
- MSI *(page A-14)*
- PDF 417 *(page A-15)*
  - Macro PDF *(page A-16)*
  - Micro PDF 417 *(page A-17)*
- Plessey *(page A-13)*
- QR Code *(page A-22)*
- Standard 2 of 5 *(page A-6)*
- Telepen *(page A-20)*
- UPC/EAN *(page A-8)*

Section 7

Programming

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

- Creating CAB Files
- FTP Server *(page 7-12)*
- Full Screen *(page 7-19)*
- Kernel I/O control functions *(page 7-20)*
- Reboot Functions *(page 7-35)*
- Remapping the Keypad *(page 7-36)*

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 7-2)*.
2. *Optional* Create a SETUP.DLL file to provide custom control of the installation process *(page 7-10)*.
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 7-11)*.

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*"
Must be "$Windows NT$" as Windows CE is not available on Windows 95.

Provider: "*INF_creator*"
The company name of the application, such as "Microsoft."

CESignature: "$Windows CE$"

EXAMPLE:

```
[Version]
Signature = "$Windows NT$"
Provider = "Microsoft"
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 will be 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 will be replaced with this string value. Example: \storage_card\%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 ""<value>"" 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\Microsoft\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 SH3 CPU is 10003 and the MIPS CPU is 4000.

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**. For Windows CE Japanese language devices, set this to 2.01

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**. For Windows CE Japanese language devices, set this to 2.01

▶ **NOTE:** *Supported Windows CE operating system versions include 1.0, 1.01, 2.0, 2.01, and 2.10. When using these numbers, be sure to include all significant digits.*

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.SH3]                      ; Inherits all [CEDevice] settings
; This will create a .CAB file specific to SH3 devices.
ProcessorType = 10003               ; SH3 .cab file is valid for SH3 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 sh3 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. These can be used 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 table on the next page 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 nonzero 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\wce212\6110ie\mfc\lib\x86

[SourceDisksFiles]   ; Required section
rpm.exe      = 1,C:\Appsoft\program\wce212\WCEX86Rel6110
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\wce211\NMSD61102.11\mfc\lib\x86
rdm45wce.dll = 1,c:\rptools\rdm45wce\4_50\lib\wce212\wcex86rel
picfmt.dll   = 1,c:\rptools\picfmt\1_00\wce212\wcex86rel6110
fmtctrl.dll  = 1,c:\rptools\fmtctrl\1_00\wce212\wcex86rel6110
ugrid.dll    = 1,c:\rptools\ugrid\1_00\wce212\wcex86rel6110
simple.dll   = 1,c:\rptools\pspbm0c\1_00\wce211\wcex86rel
psink.dll  = 1,c:\rptools\psink\1_00\wce211\WCEX86RelMinDependency
pslpwce.dll  =1,c:\rptools\pslpm0c\1_00\wce211\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 7-4) in the .INF file to point to SETUP.DLL.*

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:

```
cd\"Windows CE Tools"\WCE211\"MS HPC Pro"\support\appinst\bin
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. A microprocessor tag 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 SH3 and MIPS microprocessors, assuming that the Win32 SETUP.INF file contains the SH3 and MIPS tags:

```
cabwiz.exe "c:\myfile.inf" /err myfile.err /cpu sh3 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.

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.

Synopsis:

ftpdce [*options*]

Options:

–Aaddr	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 16384.*
–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:** *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.*
–Iip	Sets the preferred 6920 Communications Server. *[optional]*
–Llog	Sets the state of logging. *Default is 0 (disabled).*
–Nsec	Specifies the number of seconds to wait before 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 directory of the drive from which the FTP Server program was executed.
–Tscript	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.

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:

ACCT *(Ignored)*
Specifies account.

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
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.

PORT
Specifies a data connection port.

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 nonstandard or operating system (OS)-specific 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 *(not supported on all platforms).*
(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*]

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 that can be used 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 the users they wish. This is in addition to the default Intermec account which can be 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 will hide this file from users. Once an access control list has been installed on the 700 Series Computer, a new one will not be accepted by the FTP Server until the previous one is removed.

Encrypted access control lists are not portable between 700 Series Computers.

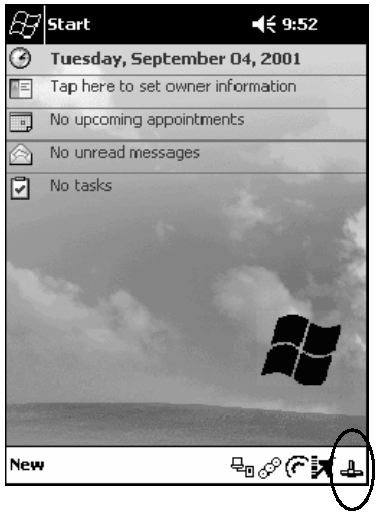
Stopping the FTP Server from your Application

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 this event, 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, 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)* to get this pop-up menu.



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**.

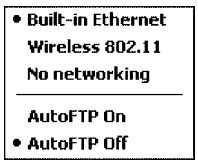
```
HKEY_LOCAL_MACHINE\Software\Intermec\Ndistray\StartupIFTP
```

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.

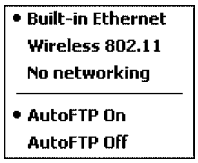
▶ NOTE: *If a network driver is unloaded using the NDISTRAY popup menu, and the FTP Server is running, the FTP Server is stopped.*

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 and 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 be running, but the FTP connection to the client should be dropped.

Full Screen

Pocket PC 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 2002 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 Section 3, "*Installing Applications and Updating System Software*," for more information on how to do this. Saving a copy of the registry will allow most system settings to be restored 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

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 The 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:

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.

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 keyboard 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.

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 will be restored upon a cold boot. FALSE indicates that the RTC will not be 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 be installed at boot time. FALSE indicates the data collection software should not be installed.

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.

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.

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.

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*” below.
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:

ITC_REGISTRY_LOCATION
This IOCTL sets the default location for where to write the registry when RegFlushKey() is called by an application. The registry may be saved to Flash, a CompactFlash storage card or a SecureDigital storage card. *lpOutBuf* must point to a buffer that contains a byte value of “1” for the CompactFlash card or “2” for the SecureDigital card to specify the location.

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.

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
This IOCTL sets the threshold for the keyboard 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
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 old-style 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`	“PXA250\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.

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;`	‘B’
`tgtcustomer`	“”
`tgtplat`	SeaRay
`tgtplatversion`	Current build version number of the bootstrap loader
`tgtcputype[8];`	“Intel\0”;
`tgtcpu`	“PXA250\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:

```
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
```

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.

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.

lpInBufSize 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];   // “PXA250\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.

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 7-32.

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 7-30.

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 7-30.

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 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 740 Keypad. Keys that are to be used in more than one shift plane must be described in each plane.

Unshifted plane:
Contains values from the keypad when not pressed with other keys, such as the following:

- **[1]** 1
- **[5]** 5
- **[9]** 9

Gold plane:
Contains values from the keypad when a key is simultaneously pressed with the **[Gold]** key, such as the following:

- **[Gold]** + **[1]** Send
- **[Gold]** + **[5]** A3
- **[Gold]** + **[9]** PageDown

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:

- **[Alpha]** + **[1]** Caps
- **[Alpha]** + **[5]** JKL
- **[Alpha]** + **[9]** WXYZ

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().

- 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`

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 table below). 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 740 Keypad identifies keys as scan codes. These scan codes are sent via the keypad microcontroller, and cannot be changed without modifying the keypad firmware.

Key/Meaning	Scancode
Reserved	0x00
I/O Button	0x01
Scanner Trigger	0x02
Scanner Left	0x03
Scanner Right	0x04
.	0x05
4	0x06
None	0x07
Left Arrow	0x08
None	0x09
Backspace	0x0A
Gold Key	0x0B
None	0x0C
ESC	0x0D
Down Arrow	0x0E
1	0x0F
7	0x10
Alpha Key	0x11
None	0x12
Up Arrow	0x13
Right Arrow	0x14
2	0x15
8	0x16
0	0x17
5	0x18
None	0x19
Action Key	0x1A
3	0x1B
9	0x1C
ENTER	0x1D
6	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

Sample View of Registry Keys

The following is a sample view of the current default key mapping. 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
```

Appendix A

Control Panel Applets

This appendix contains information about the Data Collection, SNMP, and User Information Control Panel applets that may be on your 700 Series Color Mobile Computer.

SNMP 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 Section 4, "Network Support," for more information about the radio modules or the wireless printing.*

Configuration Parameters

A configuration parameter changes the way the 700 Series Color (700C) 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.
- Access the 700 Series Computer via the Unit Manager through a web browser on your desktop PC via the SRDEVMGMT.CAB file. To use the Unit Manager, install this CAB file from the 700 Color Software Tools CD-ROM. See the *Software Tools Help* on the 700C Tools CD for assistance.
- Send parameters from an SNMP management station. See "*SNMP Configuration*" starting on page 4-37.
- 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.

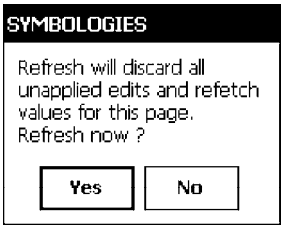
Changing a Parameter Setting

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 the following 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.

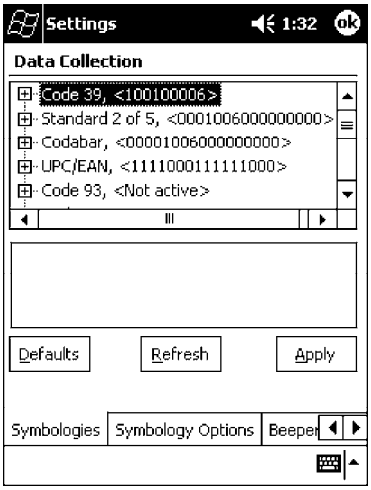
Syntax or **Options**:
Syntax lists the two-character code for the parameter, if the parameter is configurable by scanning a bar code or by sending parameters through a network. Both **Syntax** and **Options** list acceptable values for the parameter. *Default settings are noted in italic.*

Data Collection Control Panel Applet

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:** *Icons are shown to the left.*

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 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 four groups of settings or parameters:

Symbologies

Symbology Options *(starting on page A-24)*

Beeper/LED/Buttons *(starting on page A-32)*

Virtual Wedge *(starting on page A-37)*

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 C, "*Bar Code Symbologies*." *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 A-16)*, Micro PDF 417 *(page A-17)*, Matrix 2 of 5 *(page A-19)*, Telepen *(page A-20)*, and Code 11 *(page A-21)* symbologies are not supported. Likewise, when using a laser scanner, the QR Code *(page A-22)* and Datamatrix *(page A-23)* symbologies are not supported.

The following table shows which bar code symbologies are supported either by an imager or by a laser scanner.

Bar Code Symbology	Imager	Laser Scanner
Code 39	X	X
Interleaved 2 of 5	X	X
Standard 2 of 5	X	X
Matrix 2 of 5		X
Code 128	X	X
Code 93	X	X
Codabar	X	X
MSI		X
Plessey		X
UPC	X	X
EAN/EAN 128	X	X
Code 11		X
PDF 417	X	X
Micro PDF 417		X
Telepen		X
Datamatrix	X	
QR Code	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.

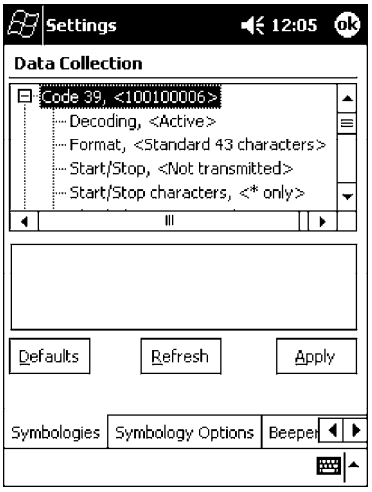
SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.3.1

Options:

Decoding	0	Not active
	1	Active *(default)*
Format	0	Standard 43 characters *(default)*
	1	Full ASCII
Start/Stop	0	Not transmitted *(default)*
	1	Transmitted
Start/Stop characters *(Not supported when using an imager)*:		
	0	$ (dollar sign) only
	1	* (asterisk) only *(default)*
	2	& and * (dollar sign and asterisk)
Check digit	0	Not used *(default)*
	1	Mod 43 transmitted
	2	Mod 43 not transmitted
	3	French CIP transmitted
	4	French CIP not transmitted
	5	Italian CPI transmitted
	6	Italian CPI not transmitted
Bar code length	0	Any length *(default)*
	1	Minimum length
Minimum length	000–254	Minimum length 1–254 *(6)*

▶ **NOTE:** *If "1" is selected for **Bar code length**, 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.

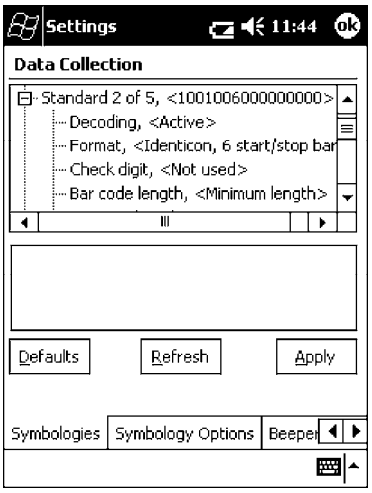
SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.4.1

Options:

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

▶ **NOTE:** *If "1" is selected for **Bar code length**, then **Minimum length** is entered. If "2" is selected for **Bar code length**, 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.

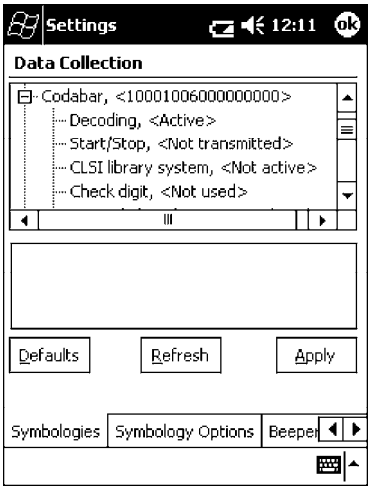
SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.5.1

Options:

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

▶ **NOTE:** *If "1" is selected for **Bar code length**, then **Minimum length** is entered. If "2" is selected for **Bar code length**, 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:

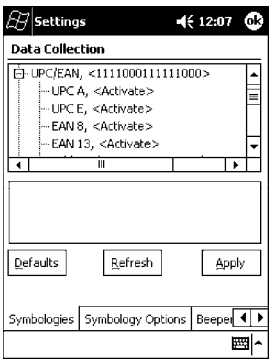
Tap (+) to expand the **UPC/EAN** parameter, select the setting to be changed, then select an option to change this setting.

SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.6.1

Options:

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

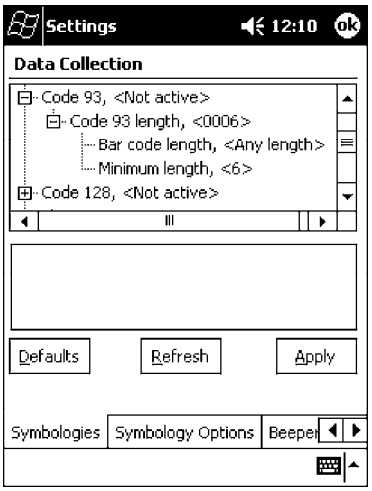
SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.19.1

Options *(active only if Code 93 is active)*:

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

► **NOTE:** *If "1" is selected for **Bar code length**, 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:

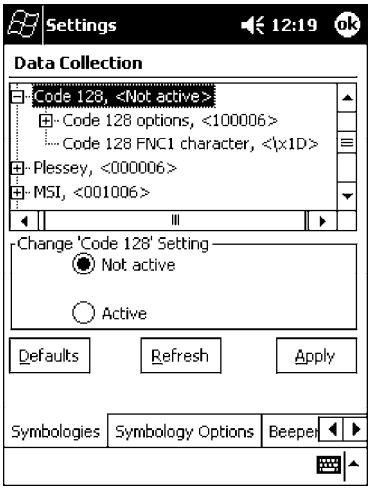
Tap the **Code 128** parameter, then select an option to change this parameter setting. Tap (+) to expand either the **Code 128 Options** or **Code 128 FNC1 character** parameters.

SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.9.1

Options:

0 Not active *(default)*
1 Active



Code 128 Options

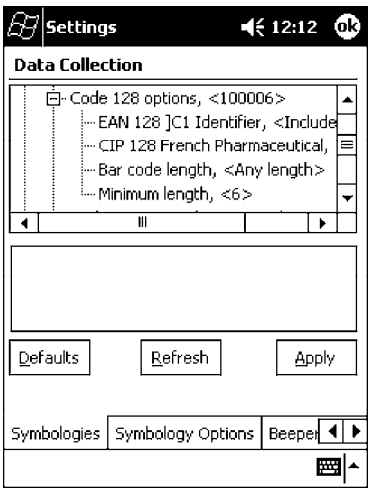
Set the following for the Code 128 parameter:

Action:

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

Options:

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



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. 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:

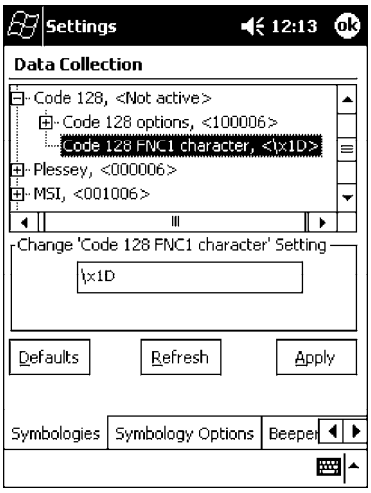
Tap (+) to expand the **Code 128** parameter, then type the ASCII characters to be set for the **Code 128 FNC1 character** parameter.

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.

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.

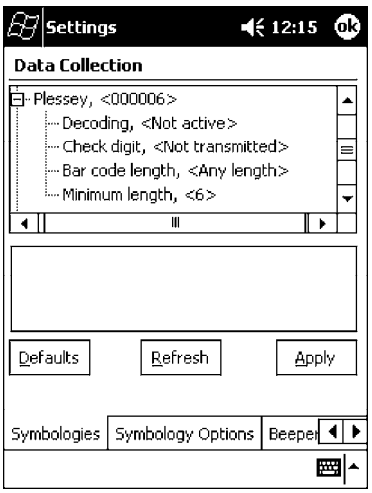
SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.10.1

Options:

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

▶ **NOTE:** *If "1" is selected for **Bar code length**, then **Minimum length** is entered.*



MSI

MSI is a symbology similar to Plessey code (page A-13) that includes a start pattern, data characters, one or two check digits, and a stop pattern.

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.

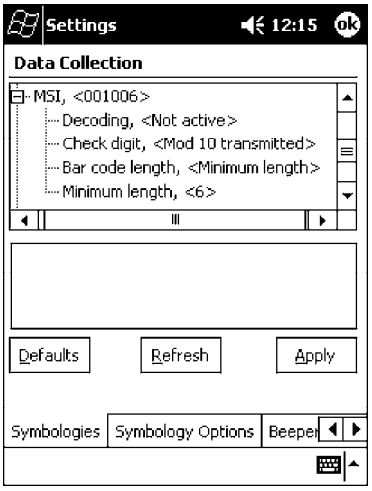
SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.15.1

Options:

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

▶ **NOTE:** *If "1" is selected for **Bar code length**, then **Minimum length** is entered.*



PDF 417

PDF 417 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 PDF 417 labels of more than 128 characters, you can develop an application that bypasses the keypad buffer.

Action:

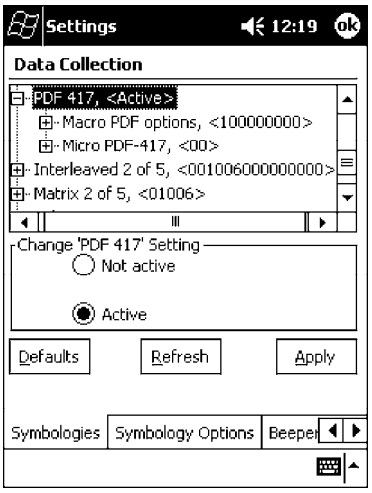
Tap the **PDF 417** parameter, then select an option to change this parameter setting. Tap (+) to access either the **Macro PDF Options** parameter or the **Micro PDF 417** parameter.

SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.17.1

Options:

0 Not active
1 Active *(default)*



Macro PDF

Macro PDF is used when a long message requires more than one PDF 417 label. *Note that this is not available when you use an imager with your 700 Series Computer.*

- Select **Buffered** to store a multi-label PDF 417 message in the Sabre buffer, thus transmitting the entire message when all labels have been read.
- Select **Unbuffered** for multi-label PDF 417 messages that are too long for the Sabre buffer (memory overflow). Each part of the PDF 417 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:

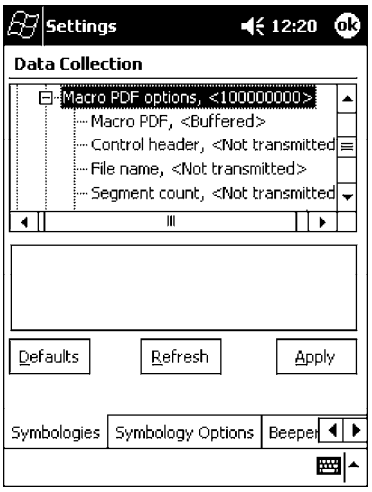
Tap (+) to expand the **PDF 417** parameter, tap (+) to expand the **Macro PDF** parameter, select a setting to be changed, then select an option to change this setting.

SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.22.1

Options:

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



Micro PDF 417

Micro PDF 417 is a multi-row symbology derived from and closely based on PDF 417 *(page A-15)*. 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:

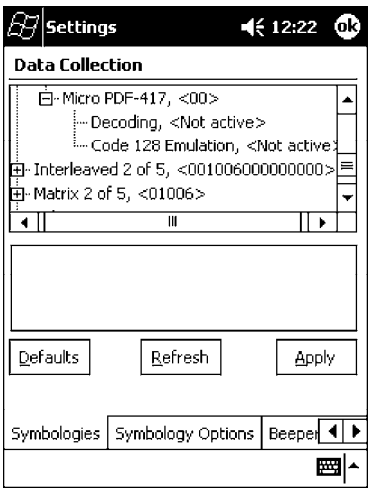
Tap (+) to expand the **PDF 417** parameter, tap (+) to expand the **Micro PDF 417** parameter, select a setting to be changed, then select an option to change this setting.

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.

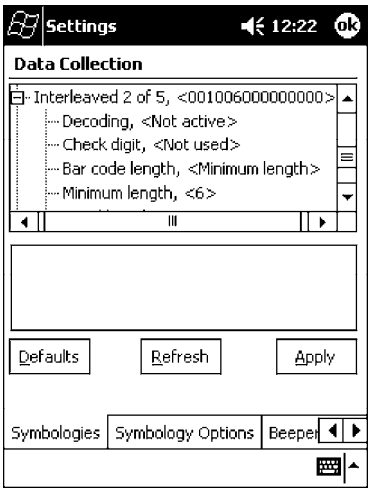
SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.23.1

Options:

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

▶ **NOTE:** *If "1" is selected for **Bar code length**, then **Minimum length** is entered.*
*If "2" is selected for **Bar code length**, 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.

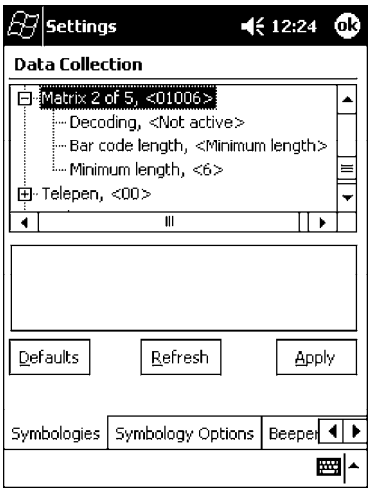
SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.24.1

Options:

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

▶ **NOTE:** *If "1" is selected for **Bar code length**, 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:

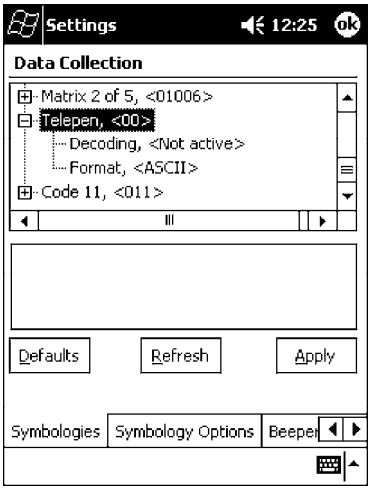
Tap (+) to expand the **Telepen** parameter, select the setting to be changed, then tap an option to change this setting.

SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.25.1

Options:

Decoding	0	Not active *(default)*
	1	Active
Format	0	ASCII *(default)*
	1	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:

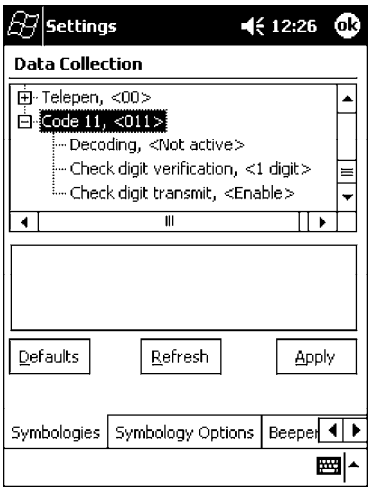
Tap (+) to expand the **Code 11** parameter, select the setting to be changed, then tap an option to change this setting.

SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.26.1

Options:

Decoding	0	Not active *(default)*
	1	Active
Check digit verification	1	1 digit *(default)*
	2	2 digits
Check digit transmit	0	Disabled *(default)*
	1	Enabled



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.*

Action:

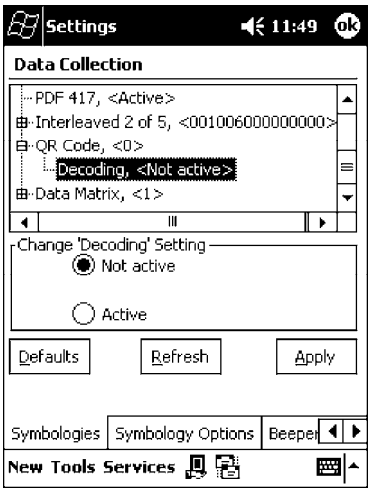
Tap (+) to expand the **QR Code** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.35.1

Options:

Decoding 0 Not active
1 Active *(default)*



Datamatrix

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.*

Action:

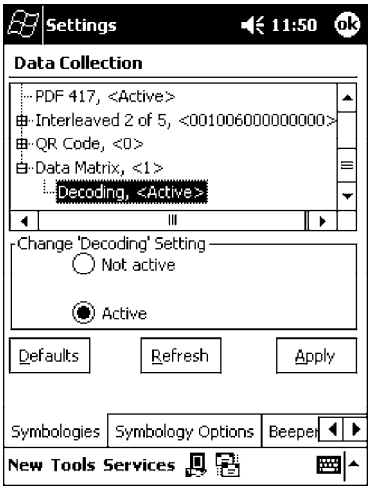
Tap (+) to expand the **Datamatrix** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

SNMP OID:

1.3.6.1.4.1.1963.15.3.3.1.1.34.1

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 has been 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:

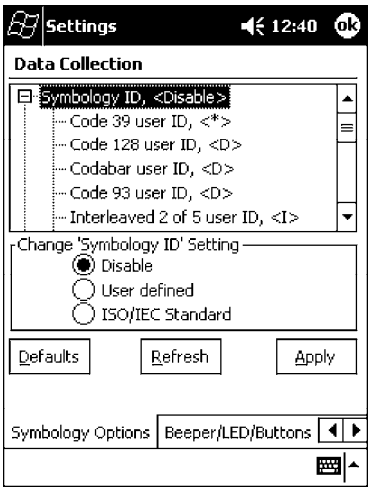
User-defined ASCII Character *(Option 1)*:
A user-defined symbology identifier is a single ASCII character. You can assign a custom identifier character to each bar code symbology.

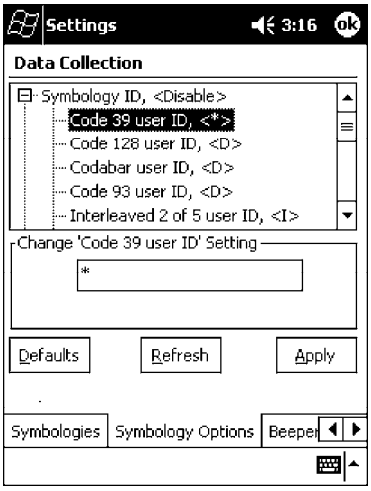
AIM ISO/IEC Standard *(Option 2 — Required to define symbology IDs)*:
The AIM Standard has a three-character structure which indicates the symbology and optional features. See the *AIM ISO/IEC Standard* for more information.

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).*

PDF-417 User ID

If "1" was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify PDF 417 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 **PDF 417 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:

1.3.6.1.4.1.1963.15.3.3.4.1.1 1.1

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:

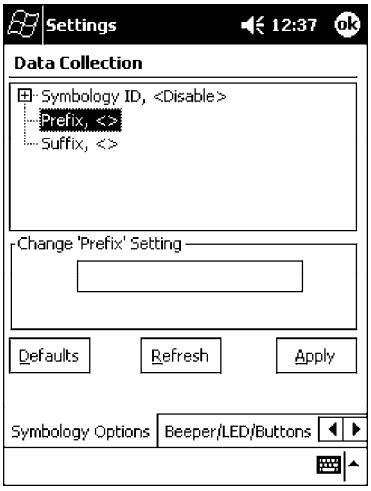
Tap the **Prefix** parameter, then enter a prefix value to change this parameter setting.

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:

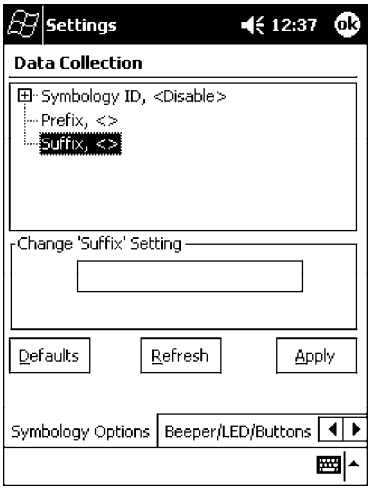
Tap the **Suffix** parameter, then enter a suffix value to change this parameter setting.

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/Buttons

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/Buttons** 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 Volume	X	X
Beeper Frequency		X
Good Read Beeps		X
Good Read Beep Duration		X
Record Button	X	X

The following are parameters for features on the700 Series Computer. *Note that these are listed in the order of their appearance.*

Beeper Volume

Sets the volume for the good read beep.

Action:

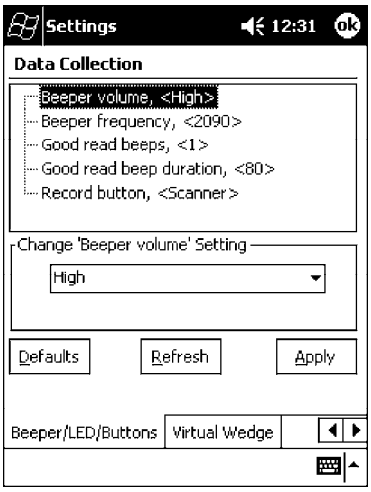
Tap the **Beeper volume** parameter, then select an option to change this parameter setting.

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 Vibrate



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:

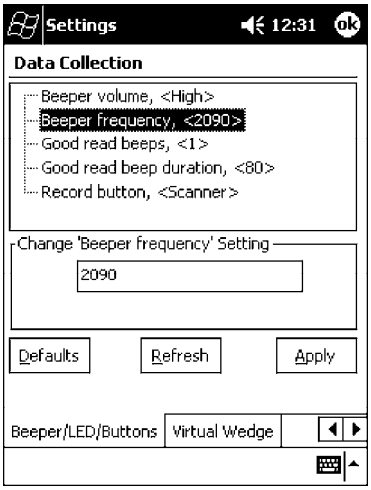
Tap the **Beeper frequency** parameter, then enter a frequency value to change this parameter setting.

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:

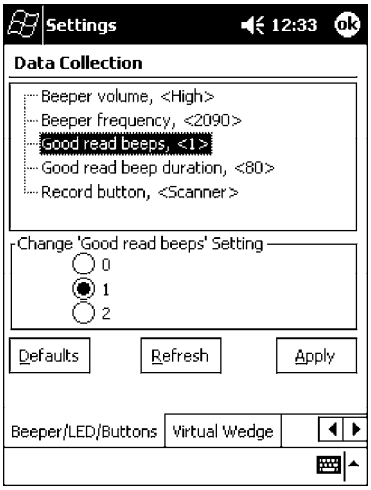
Tap the **Good read beeps** parameter, then select an option to change this parameter setting.

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:

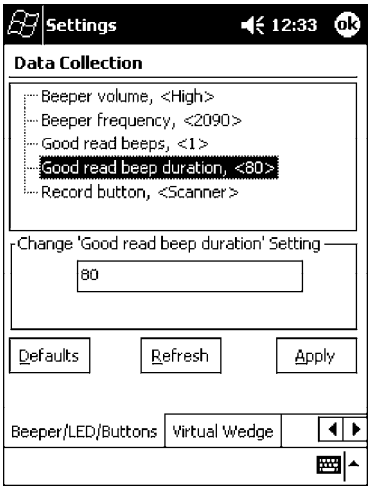
Tap the **Good read beep duration** parameter, then enter a duration value to change this parameter setting.

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)*



Record Button

This allows the configuration of the left scan button on the 700 Series Computer to either activate the scanner or the Record feature via a Pocket PC application. *See Section 2, "Pocket PC 2002," for more information about recording a message.*

Action:

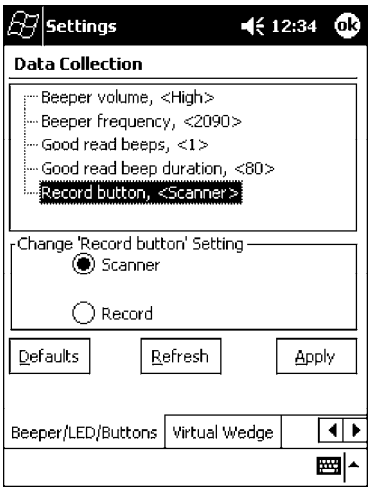
Tap the **Record button** parameter, then select either option to change this parameter setting.

SNMP OID:

1.3.6.1.4.1.1963.15.3.1.4.1.13.1

Options:

0 Scanner *(default)*
1 Record



Virtual Wedge

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 **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:

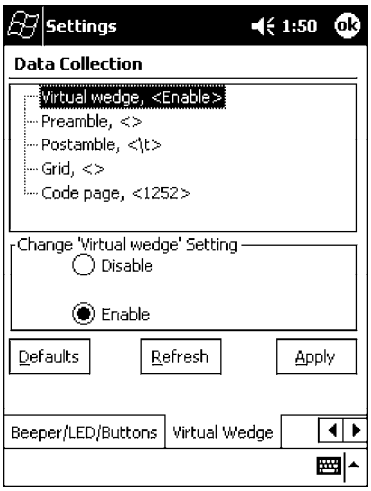
Tap the **Virtual Wedge** parameter, then tap an option to change this parameter setting.

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.

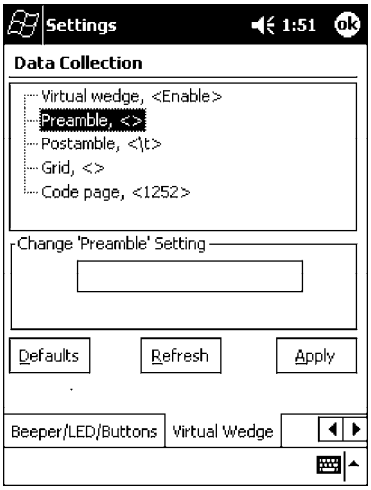
SNMP OID:

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. *Default is no characters (disabled).*



▶ **NOTE:** *When you enter the AD command without data, the preamble is disabled. If you want to use quotation marks or these combinations of characters:*
AD
AE
AF
KC
BV
EX
DF
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. See the following example:

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.

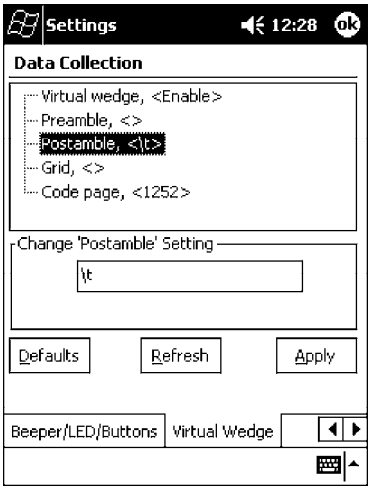
SNMP OID:

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. *Default is the tab character (\t).*



▶ **NOTE:** *When you enter the AE command without data, the postamble is disabled. If you want to use quotation marks or these combinations of characters:*
AD
AE
AF
KC
BV
EX
DF
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. See the following example:

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:

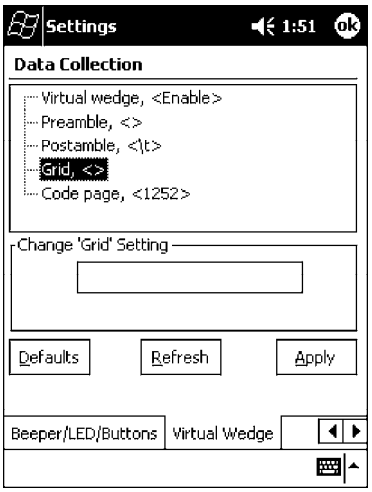
```
AF<symID> filter-expression= > editing-expression
```

where:

<symID>
The AIM symbology ID.

filter-expression
Any character string that includes valid filter expression values, and editing-expression is any character string that includes valid editing expression values.

<width> is any positive integer or NULL. A NULL width means that the field type (defined next) applies all the way to the end of the data string. A non-NULL width means that the field applies to that many characters of data. The grid can be up to 240 characters in length. *Default is NULL.*



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:

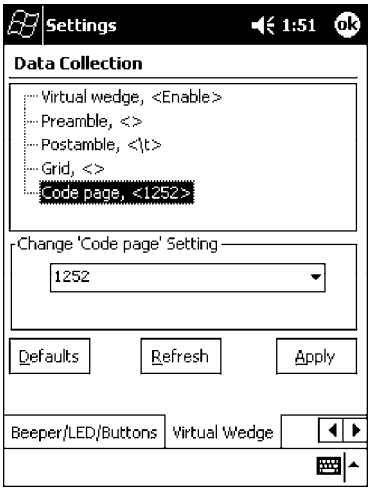
Tap the **Code Page** parameter, then select an option to change this parameter setting.

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.



SNMP Control Panel Applet

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

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **SNMP** icon 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 A-48)*

Identification *(starting on page A-50)*

Security

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **SNMP** icon → 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:

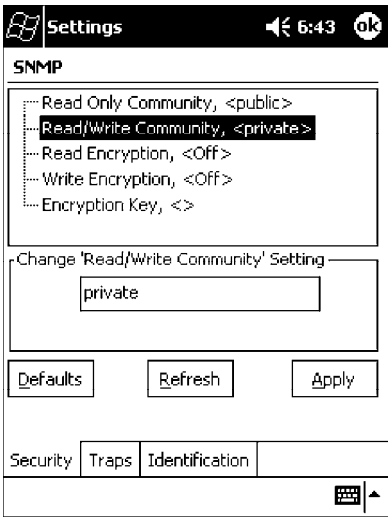
Tap the **Read/Write Community** parameter, then enter a community string to change this parameter setting.

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 A-47).*

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 A-47).*

Action:

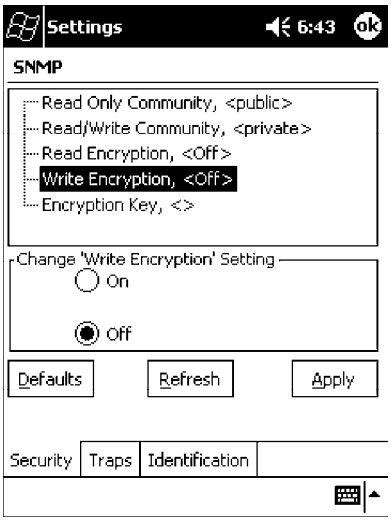
Tap the **Write Encryption** parameter, then select an option to change this parameter setting.

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 A-45) or **Write Encryption** (page A-46) 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 → the **SNMP** icon → 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:

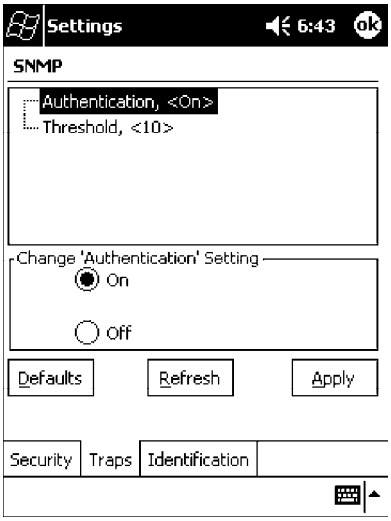
Tap the **Authentication** parameter, then select an option to change this parameter setting.

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:

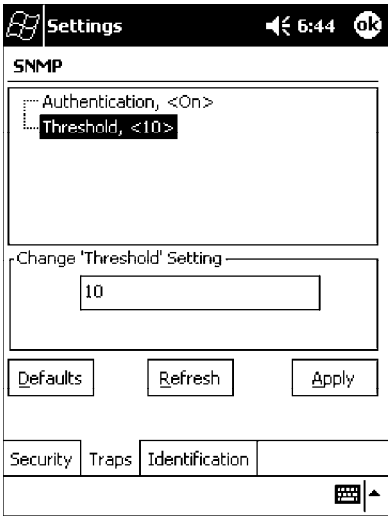
Tap the **Threshold** parameter, then enter a threshold value to change this parameter setting.

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 → the **SNMP** icon → 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:

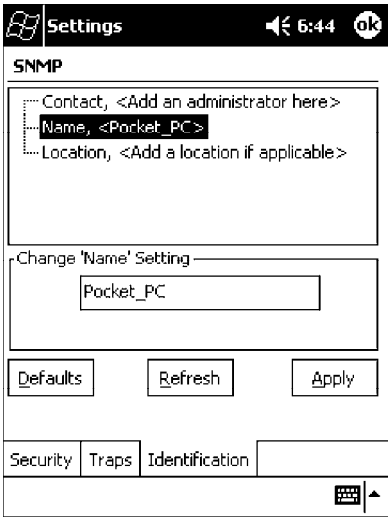
Tap the **Name** parameter, then enter the name of your 700 Series Computer to change this parameter setting.

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:

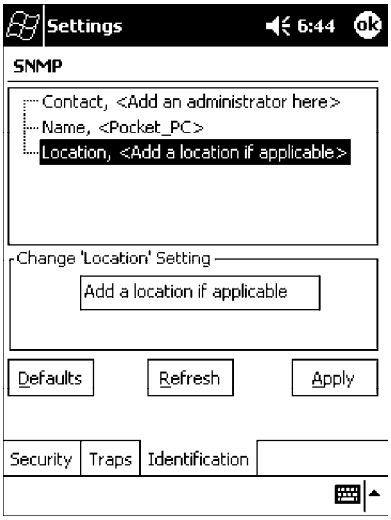
Tap the **Location** parameter, then enter the location of where your 700 Series Computer to change this parameter setting.

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.*



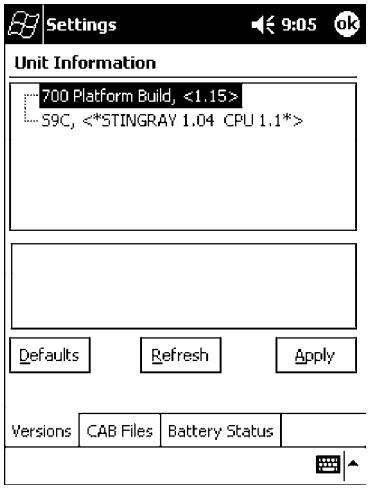
Unit Information Control Panel Applet

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.

This control panel applet is only available in the 700 Series Computer if Intermec Content is enabled, the Plus region is enabled and installed, and a laser scanner is installed.


Unit Information

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



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

Versions *(starting on the next page)*

CAB Files *(starting on page A-55)*

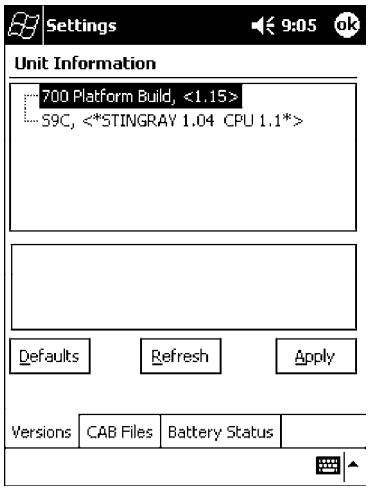
Battery Status *(starting on page A-57)*

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 → the **Unit Information** icon → the **Versions** tab to view the latest software build version. Tap **ok** to exit this information.



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.

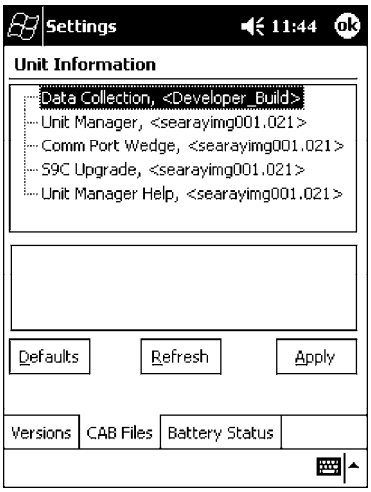
S9C:
Provides the name and version of the scanner file built into this 700 Series Computer, along with the current CPU version.

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 Help* for more information about these files.


Unit Information

To access the information from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Unit Information** icon → 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 Section 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 700C Tools CD.*

Data Collection:
Installs all Data Collection components, including functions to scan bar codes and configure Intermec-specific options on the 700 Series Computer. *More information about Data Collection parameters are provided earlier in this appendix.*

NPCPTest:
This installs a Norand® Portable Communications Protocol (NPCP) Printing test application which will print to an Intermec® 4815, 4820, or 6820 Printer. *See Section 5, "Printer Support," for more information.*

PDWPM0C:
This is the installer for the Wireless Printing Demo application. To run this demonstration, tap **Start** → **Programs** → the **Wireless Printing Demo** icon. *Press Help in the demo application for more information.*

S9C Upgrade:
Installs the files needed to upgrade the S9C scanner firmware. *See Section 3, "Installing Applications and Updating System Software," 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.*

Unit Manager:
Installs the Unit Manager application which provides tools for remotely managing the 700 Series Computer.

Unit Manager Help:
Installs the online help for the Unit Manager application.

WinCfg:
Configures the NRINET.INI file, launches the NRINet client, and loads and unloads the LAN and WLAN device drivers. *See the Windows 95 and Windows CE Configuration Utilities Reference Manual (P/N: 978-054-010) for more information.*

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 700C Tools 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.

AXSystemPower:
The System Information control that retrieves the battery charge status.

AXVWedge:
The virtual wedge control that retrieves scanned ADC data and sends it to the keyboard driver to interpret data as keyboard input.

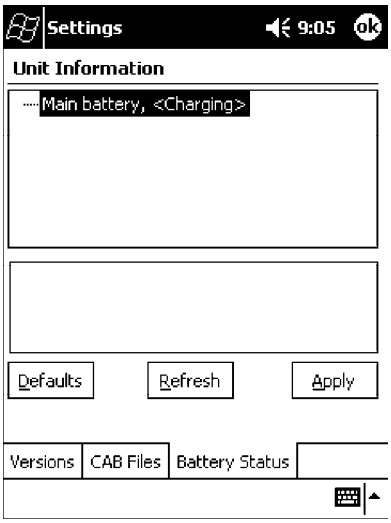
BarCodeMFCClient:
The bar code reader control that collect bar code information from the 700 Series Computer, configures the scanner, and provides audio or visual feedback when data arrives.

ITCAXUtilityEX:
Utility functions that performs a warm-boot or a cold-boot on the 700 Series Computer and assumes control over the suspend/resume function.

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 → the **Unit Information** icon → the **Battery Status** tab to view the current status. Tap **ok** to exit this information.



Appendix B

Unit Manager

NOTE: *Parameter information, such as SNMP OID and options, is detailed in Appendix A, "Control Panel Applets."*

Configuration parameters are also configurable using a Unit Manager application which accesses the 700 Series Computer through a web browser on your desktop PC via the SRDEVMGMT.CAB file.

To use the Unit Manager, install this CAB file from the 700C Software Tools CD-ROM. See the *Software Tools Help* on the 700C Tools CD for assistance. See the *Unit Manager Help* on the 700C Tools CD for information on using the Unit Manager application.

Data Collection

Data Collection

Within the Unit Manager, click **Configuration** from the left navigation bar, then click the **Data Collection** icon to access any of these tabs: Symbologies, Symbology ID, Beeper/LED/Buttons, or Virtual Wedge.

Symbologies

Within the Unit Manager, select **Configuration Management → Data Collection**, then click the **Symbologies** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Codabar *(page A-7)*
- Code 11 *(page A-21)*
- Code 128 *(page A-10)*
  - Code 128 Options *(page A-11)*
  - Code 128 FNC1 Character *(page A-12)*
- Code 39 *(page A-5)*
- Code 93 *(page A-9)*
  - Code 93 Length *(page A-9)*
- Datamatrix *(page A-23)*
- Interleaved 2 of 5 *(page A-18)*
- Matrix 2 of 5 *(page A-19)*
- MSI *(page A-14)*

- PDF 417 *(page A-15)*
  - Macro PDF *(page A-16)*
  - Micro PDF 417 *(page A-17)*
- Plessey *(page A-13)*
- QR Code *(page A-22)*
- Standard 2 of 5 *(page A-6)*
- Telepen *(page A-20)*
- UPC/EAN *(page A-8)*

Symbology ID

Within the Unit Manager, select **Configuration Management → Data Collection**, then click the **Symbology ID** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Prefix *(page A-30)*
- Suffix *(page A-31)*
- Symbology ID *(page A-24)*
  - Codabar user ID *(page A-25)*
  - Code 11 user ID *(page A-29)*
  - Code 128 user ID *(page A-25)*
  - Code 39 user ID *(page A-25)*
  - Code 93 user ID *(page A-26)*
  - EAN-13 user ID *(page A-28)*
  - EAN-8 user ID *(page A-28)*
  - Interleaved 2 of 5 user ID *(page A-26)*
  - Matrix 2 of 5 user ID *(page A-28)*
  - MSI user ID *(page A-26)*
  - PDF 417 user ID *(page A-26)*
  - Plessey user ID *(page A-27)*
  - Standard 2 of 5 user ID *(page A-27)*
  - Telepen user ID *(page A-28)*
  - UPC-A user ID *(page A-27)*
  - UPC-E user ID *(page A-27)*

Beeper/LED/Button

Within the Unit Manager, select **Configuration Management → Data Collection**, then click the **Beeper/LED/Button** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Beeper Frequency *(page A-33)*
- Beeper Volume *(page A-32)*
- Good Read Beep Duration *(page A-35)*
- Good Read Beeps *(page A-34)*
- Record Button *(page A-36)*

Virtual Wedge

Within the Unit Manager, select **Configuration Management → Data Collection**, then click the **Virtual Wedge** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Code Page *(page A-41)*
- Grid *(page A-40)*
- Postamble *(page A-39)*
- Preamble *(page A-38)*
- Virtual Wedge *(page A-37)*

SNMP


SNMP

Within the Unit Manager, click **Configuration** from the left navigation bar, then click the **SNMP** icon to access any of these tabs: Security, Traps, or Identification.

Security

Within the Unit Manager, select **Configuration Management → SNMP**, then click the **Security** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Encryption Key *(page A-47)*
- Read Encryption *(page A-45)*
- Read Only Community *(page A-43)*
- Read/Write Community *(page A-44)*
- Write Encryption *(page A-46)*

Traps

Within the Unit Manager, select **Configuration Management → SNMP**, then click the **Traps** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Authentication *(page A-48)*
- Threshold *(page A-49)*

Identification

Within the Unit Manager, select **Configuration Management → SNMP**, then click the **Identification** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Contact *(page A-50)*
- Location *(page A-52)*
- Name *(page A-51)*

Unit


Unit

Within the Unit Manager, click **Configuration** from the left navigation bar, then click the **Unit** icon to access any of these tabs: Date/Time, Display, Keypad, Power Management, or Speaker.

Date/Time

Sets the current date and time.

Action:

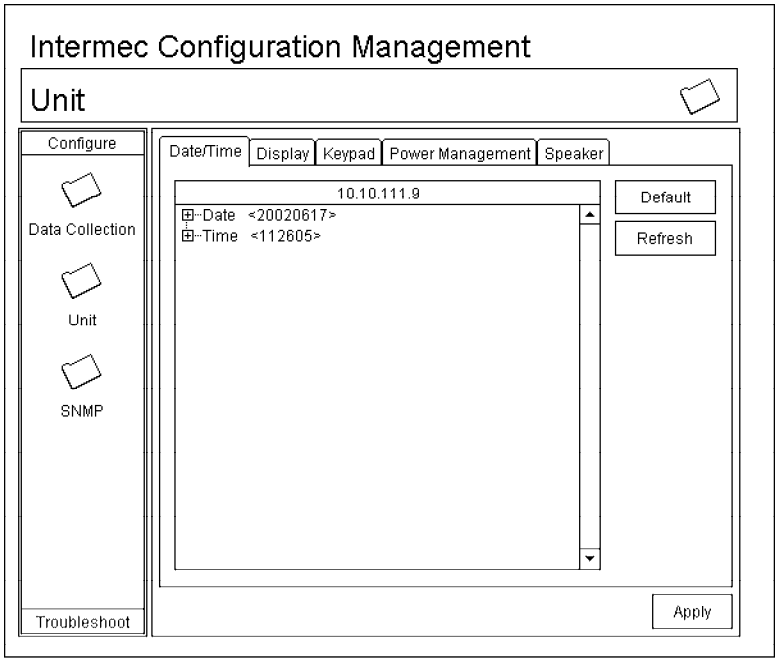
Click the **Date/Time** tab, then select **Date** or **Time** and make changes in the entry field, or tap (+) to expand either the Date or Time parameter, select the setting to be changed, then select a value from the drop-down list or enter a new value to change this setting.

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:

Date	Year	0000–999 *(1999)*
	Month	1–12 *(6)*
	Day	1–31 *(1)*
Time	Hour	0–23 *(0)*
	Minute	0–59 *(00)*
	Second	0–59 *(00)*



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.

Action:

Click the **Display** tab, then select an option from the **Backlight timeout** drop-down list.

SNMP OID:

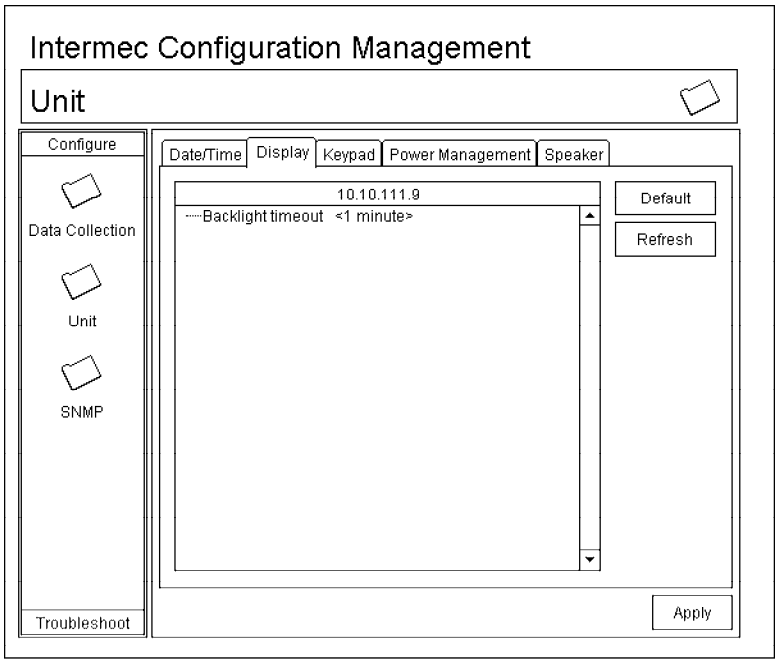
1.3.6.1.4.1.1963.15.13.1.0

Syntax:

`DFdata`

where *data* is any of the following:

10 10 seconds
30 30 seconds
60 1 minute *(default)*
120 2 minutes
180 3 minutes
240 4 minutes
300 5 minutes



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.

Action:

Click the **Keypad** tab, then select an option from the **Key clicks** drop-down list.

SNMP OID:

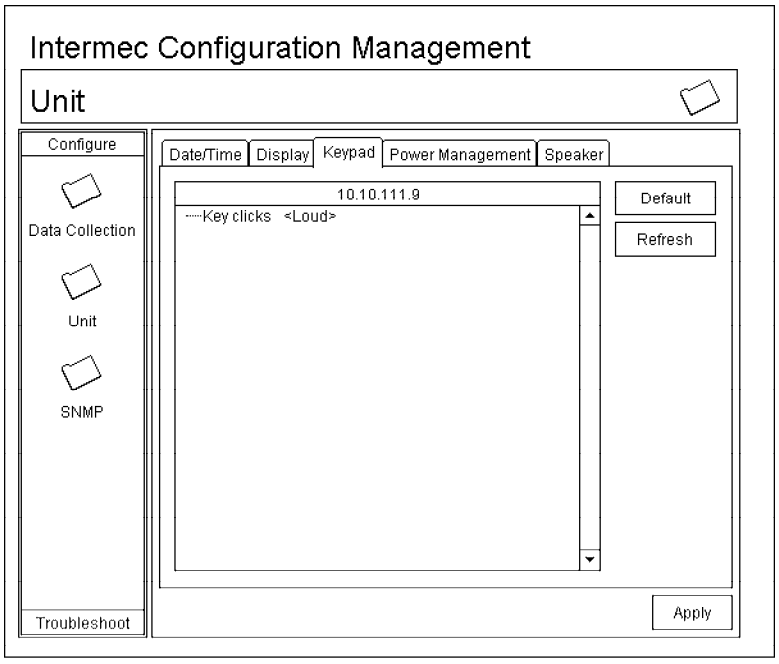
1.3.6.1.4.1.1963.15.12.1.0

Syntax:

`KCdata`

where *data* is any of the following:

0 Disable clicks
1 Enable soft key clicks
2 Enable loud key clicks *(default)*



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.

Action:

Click the **Power Management** tab, then select an option from the **Automatic shutoff** drop-down list.

SNMP OID:

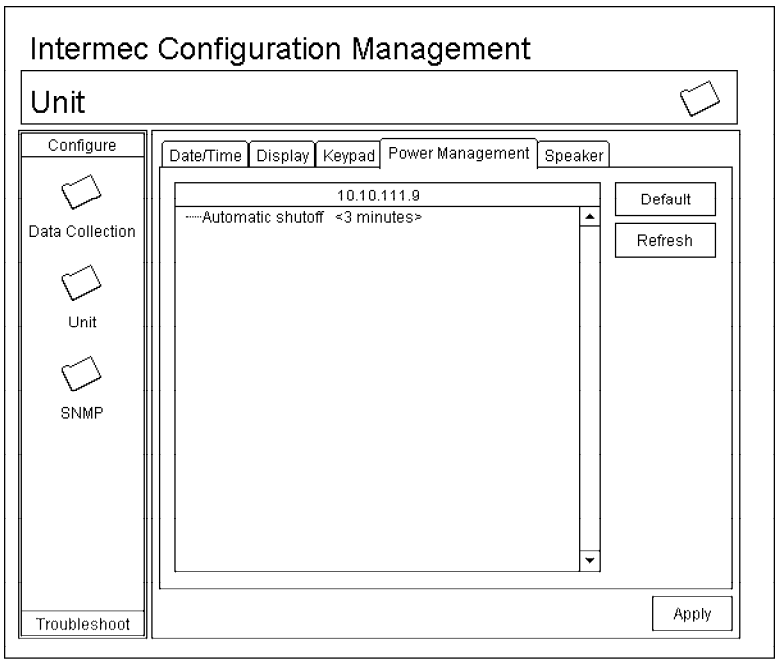
1.3.6.1.4.1.1963.15.1 1.3.0

Syntax:

`EZdata`

where *data* is any of the following:

1	1 minute
2	2 minutes
3	3 minutes *(default)*
4	4 minutes
5	5 minutes



Volume

Changes the volume of all audio signals.

Action:

Click the **Speaker** tab, then select an option from the **Volume** drop-down list.

SNMP OID:

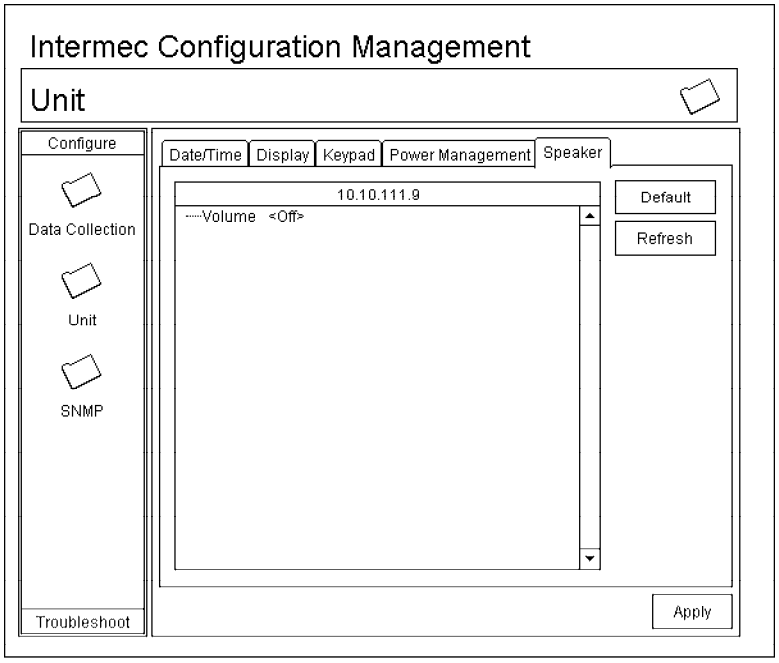
1.3.6.1.4.1.1963.15.3.1.3.0

Syntax:

`BVdata`

where *data* is any of the following:

0 Off
1 Very quiet
2 Quiet
3 Normal *(default)*
4 Loud
5 Very loud



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 Appendix A, "Control Panel Applets" 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:

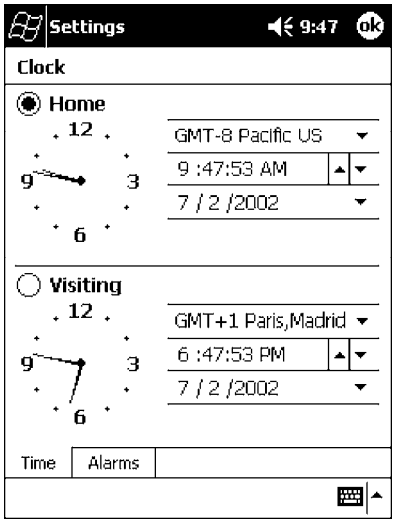
```
/+ yyyymmddhhmmss
```

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.



Appendix C

Bar Codes

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. It also includes several bar code labels that can be scanned into your 700 Series Computer.

Bar Code Symbologies

Specific bar code algorithms can be enabled using the setup menus or the host computer. Once the computer correctly decodes a bar code, the computer encodes data with descriptive information about the symbol. Response time is improved by limiting the computer to the bar codes being used.

Table C-1
Bar Code Data String Formats

Data Bar Code Type	Data Format	Data Length
UPC short (UPC-E)	ndddddc	8
EAN short (EAN-8)	fnddddc	8
UPC long (UPC-A)	nddddddddddc	12
EAN long (EAN-13)	fnddddddddddc	13
UPC short add-on 2	nddddddcaa	10
EAN short add-on 2	fnddddcaa	10
UPC long add-on 2	nddddddddddcaa	14
EAN long add-on 2	fnddddddddddcaa	15
UPC short add-on 5	nddddddcaaaaa	13
EAN short add-on 5	fnddddcaaaaa	13
UPC long add-on 5	nddddddddddcaaaaa	17
EAN long add-on 5	fnddddddddddcaaaaa	18
Interleaved 2 of 5	d.....d	1 to 31
Standard 2 of 5	d.....d	1 to 31
Plessey	d.....dc	2 to 31
Codabar	sd....ds	3 to 31
Code 11	d.....d	1 to 31
Code 39	d.....d	1 to 31
Extended Code 39	d.....d	1 to 31
Code 93	d.....d	1 to 31
Code 128	d.....d	1 to 31

▶ **NOTE:** *These bar code data definitions apply to the Data Format column in Table C-1:*

- *a Add-on code digits*
- *c Check digits*
- *d Bar code digits*
- *f EAN flag 1 characters*
- *n Number system digits*
- *s Start and stop digits*

If MOD 10 or MOD 11 check digits are enabled, the digit falls at the end of a bar code data string. Each check digit enabled extends the bar code data string length by one character.

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.

- UPC (Universal Product Code) with/without add-ons
- EAN (European Article Numbering Code) with/without add-ons
- Codabar
- C11 (Code 11)
- C39 (Code 39)
- C93 (Code 93)
- C128 (Code 128)
- I 2 of 5 (Interleaved 2 of 5 Code)
- S 2 of 5 (Standard 2 of 5)
- Plessey
- MSI (a variant of Plessey)

UPC

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.

EAN

EAN (European Article Numbering) 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 throughout most of Europe in the retail environment. Although similar to UPC symbology, the two are not interchangeable.

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 700 Series Computer handles symbology that is 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 C-8 to scan several Code 39 bar code labels available to change settings on your 700 Series 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. This is done using two character sequences made up of one of the symbols (“$”,“.”,“%”,“/”) followed by one of the 26 letters.

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.

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.

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.

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 are not self-checking and employ a variety of check characters. Plessey employs 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)

In addition to Plessey characteristics, the MSI Code employs a Modulus 10 Check. 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.

Bar Code Labels

You can change some settings on your 700 Series Computer by scanning the following Code 39 bar code labels.

You can use the Unit Manager application to set the Automatic Shutoff, Volume, Backlight Timer, or Key Clicks parameters *(starting on page B-4)*.

You can use the Unit Manager application or the Data Collection control panel to set the three Virtual Wedge parameters *(starting on page A-37)*.

▶ **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 that parameter information is on page B-8.

Turn Audio Off

$+BV0

Set Audio Volume to very quiet

$+BV1

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 that parameter information is on page B-7.

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 that parameter information is on page B-5.

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 that parameter information is on page B-6.

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 A-40.*

*$+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 A-38.*

*$+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 A-39.*

*$+AE

Classes and Functions

NOTE:

This index covers classes and functions for the 700 Series Color Mobile Computer. The general index follows this index.

K

O

P

Q

R

S

T

U

V

W

General Index

NOTE:

This index covers parameters and general topics. Parameter names are italicized. The index of classes and functions precedes this general index.

NUMBERS

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Files Index

NOTE:

This index section is provided to assist you in locating descriptions for device drivers, applications, utilities, batch files, or other files within this publication.

NUMBERS

A

B

C

D

E

F

I

L

M

N

O

P

R

S

T

W

Corporate Headquarters
6001 36th Avenue West
Everett, Washington 98203
tel 425.348.2600
fax 425.355.9551
www.intermec.com

700 Series Color Mobile Computer User's Manual - September 2002

961054031 REV A