MOTOROLA SCANNER OPOS DRIVER DEVELOPER'S GUIDE

MOTOROLA SCANNER OPOS DRIVER DEVELOPER'S GUIDE

72E-149783-01

Rev. A

May 2011

Revision History

Changes to the original guide are listed below:

Change	Date	Description
-01 Rev A	5/2011	Initial release.

TABLE OF CONTENTS

Revision History iii

About This Guide

Introduction vii
Chapter Descriptions vii
Notational Conventions viii
Service Information ix

Chapter 1: INTRODUCTION TO THE MOTOROLA SCANNER OPOS DRIVER

Overview 1-1
Motorola Scanner OPOS Driver Architecture 1-2

Chapter 2: INSTALLATION & CONFIGURATION

Overview 2-1
Configuration 2-2
Scanner Configuration Bar Codes 2-3
USB Communication Protocol 2-3
RS-232 Communication Protocol 2-3

Chapter 3: OPOS PROPERTIES, METHODS, EVENTS

Overview 3-1
Deviations from OPOS Specifications 3-2
Supported Feature Set 3-3
Properties 3-3
Methods 3-5
Events 3-6

Chapter 4: SAMPLE APPLICAITON (SCANNER OPOS TEST)

Overview 4-1

OPOS Sample Application (OPOS Test Utility) 4-1
OPOS Test Utility Window Functionality 4-2
Viewing Bar Code Data 4-5
Getting and Setting OPOS Properties 4-7
Creating a Custom OPOS Sample Application 4-8
Return Value and Result Code 4-8
Direct I/O 4-8
Statistics Methods 4-8
Modified Claim Functionality 4-9

Chapter 5: SUPPORTED SYMBOLOGY TYPES VS. SCANNER MODE
Overview 5-1
Supported Symbology Types vs. Scanner Mode 5-2

Index

Glossary

ABOUT THIS GUIDE

Introduction

This guide provides information about the Motorola OPOS Driver which enables bar code data communication between any scanner and an OPOS compliant POS application via either a USB (OPOS Hand Held and SNAPI) or RS-232 (Wincor Nixdorf Mode B) connection.

Chapter Descriptions

Topics covered in this guide are as follows:

- *Chapter 1, INTRODUCTION TO THE MOTOROLA SCANNER OPOS DRIVER* provides an overview of the Motorola OPOS Driver.
- *Chapter 2, INSTALLATION & CONFIGURATION* describes specific installation instructions and settings to configure the Motorola Scanner OPOS Driver on a host computer.
- *Chapter 3, OPOS PROPERTIES, METHODS, EVENTS* provides information about the Motorola OPOS Driver properties, methods, and events.
- *Chapter 4, SAMPLE APPLICAITON (SCANNER OPOS TEST)* provides information about the Motorola OPOS Driver properties, methods, and events.
- *Chapter 5, SUPPORTED SYMBOLOGY TYPES VS. SCANNER MODE* provides information about the sample application in the Motorola Scanner OPOS Driver suite.

Notational Conventions

The following conventions are used in this document:

- Courier New font is used for code segments.
- *Italics* are used to highlight:
  - Chapters and sections in this and related documents
  - Dialog box, window and screen names
  - Drop-down list and list box names
  - Screen field names
  - Check box and radio button names
  - File names
  - Directory names.
- **Bold** text is used to highlight:
  - Parameter and option names
  - Icons on a screen
  - Key names on a keypad
  - Button names on a screen.
- bullets (•) indicate:
  - Action items
  - Lists of alternatives
  - Lists of required steps that are not necessarily sequential
- Sequential lists (e.g., those that describe step-by-step procedures) appear as numbered lists.
- Notes, caution and warning statements appear as follows:

NOTE This symbol indicates something of special interest or importance to the reader. Failure to read the note will not result in physical harm to the reader, equipment or data.

CAUTION This symbol indicates that if this information is ignored, the possibility of data or material damage may occur.

WARNING! **This symbol indicates that if this information is ignored the possibility that serious personal injury may occur.**

Service Information

If you have a problem contact Motorola support for your region. Contact information is available at: http://supportcentral.motorola.com.

When contacting Mobility support, please have the following information available:

- Serial number of the unit
- Model number or product name
- Software type and version number.

Motorola responds to calls by E-mail, telephone or fax within the time limits set forth in support agreements.

If your problem cannot be solved by Motorola support, you may need to return your equipment for servicing and will be given specific directions. Motorola is not responsible for any damages incurred during shipment if the approved shipping container is not used. Shipping the units improperly can possibly void the warranty.

If you purchased your business product from a Motorola business partner, contact that business partner for support.

CHAPTER 1 INTRODUCTION TO THE MOTOROLA SCANNER OPOS DRIVER

Overview

The Motorola OPOS Driver enables bar code data communication between any scanner and an OPOS compliant POS application via either a USB (OPOS Hand Held and SNAPI) or RS-232 (Wincor Nixdorf Mode B) connection. This driver provides an interface for reading bar code data, receiving events, opening, claiming, and enabling/disabling the device in accordance with the UPOS committee's version 1.12 specification. The OPOS specification defines a two-layer open-driver software architecture between a POS application running on a Win32-based operating system and the physical POS hardware device.

- The upper layer, known as the Common Control Object (CCO), is an ActiveX Control that the POS application uses to interact indirectly with the Motorola scanner. The CCO is unique to each POS device class (e.g., scanners, scales, printers) and is provided by the UPOS committee. To simplify development and integration, the Motorola Scanner SDK includes a Monroe CCO (a vendor-independent scanner CCO provided by the UPOS committee). The \Driver folder contains the CCO.
- The lower layer, known as the Service Object (SO), is an in-process OLE Automation Server (a DLL) and is used to interact directly with the POS device, in this case the scanner. The Service Object is unique to each specific vendor's POS device. The Motorola Scanner SDK contains the Motorola specific SO.

Motorola Scanner OPOS Driver Architecture

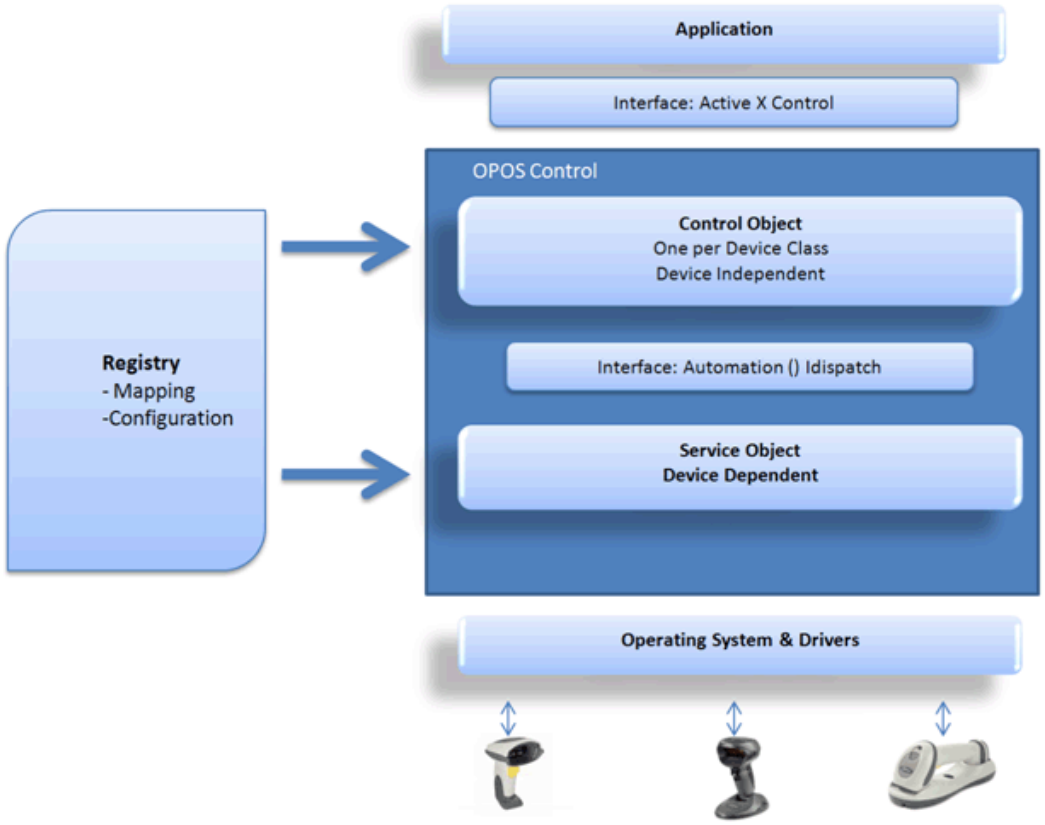


Figure 1-1 *Motorola Scanner OPOS Driver Architecture*

For more information about OPOS, OPOS architecture, terminology, and programmer's guides, refer to:

- UPOS Home Page (http://www.nrf-arts.org)
- Monroe Consulting Services, Inc. (http://www.monroecs.com/opos.htm)

CHAPTER 2 INSTALLATION & CONFIGURATION

Overview

This chapter describes installation instructions and settings to configure the Motorola Scanner OPOS Driver on a host computer.

For custom installation instructions, refer to the Scanner SDK Developer's Guide.

NOTE OPOS components are installed by default with the standard Scanner SDK installation. If a custom Scanner SDK installation is performed, the OPOS option must be selected to install the OPOS components.

Configuration

After a successful installation of the Motorola Scanner SDK with the OPOS driver, the following system registry entry is created at:

HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scanner\ SYMBOL_SCANNER

The Motorola Scanner OPOS Driver reads the registry entry above to retrieve some required configurations, for example baud rate for serial scanners) and scanner filtering rules to form the logical scanner defined by the user.

Table 2-1 *OPOS Registry*

Value Name	Sample Data	Description
SerialNumber	1. 7116000500337,7087000501981 2. * (Represents all serial numbers)	Comma separated list of scanner serial numbers for which to use with OPOS driver.
ModelNumber	1. DS6707-SR20001ZZR,DS9808* (* Represents any model number of the DS9808.)	Comma separated list of Scanner model numbers which wants to use with OPOS driver.
Connection**	ConUSB	Added for backward compatibility.
Description	Symbol scanner Service Object	Service Object description.
Type	USBIBMHID,SNAPI,NIXMODB ALL	Comma or space delimited list of scanner modes from the TypePool.
TypePool	USBIBMHID,SNAPI,NIXMODB,ALL	All the supported types only for reference. OPOS driver does not read this entry.
Port	COM1 COM1,COM2 * (Represent any COM port)	Comma or space delimited list of serial ports which needs to use with OPOS driver.
VID	0x1504	Vendor ID of Motorola Bar Code Scanners.
PID**	* (Represents all PIDs)	Product IDs of Motorola Bar Code Scanners.
Baud rate	9600 115200	Baud rate for serial port.
CompatibilityMode	0 1	Whether to registry of previous version of OPOS driver. 0 = Disable backward compatibility 1 = Enable backward compatibility

**** Added to maintain the backward compatibility and are optional if the CompatibilityMode is set to *Disable backward compatibility.***

For more details on how to use the registry, see *Modified Claim Functionality on page 4-9*.

Scanner Configuration Bar Codes

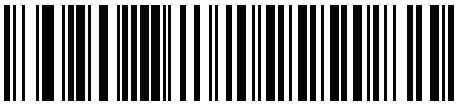
Scan the **Set All Defaults** bar code below to return all parameters to the scanner's default values. Refer to the scanner's Product Reference Guide for default values.

Set All Defaults

Scan the appropriate bar code below to configure the scanner for either USB or RS-232 communication protocols.

USB Communication Protocol

USB (OPOS Hand Held)



USB SNAPI

RS-232 Communication Protocol

Wincor-Nixdorf RS-232 Mode B

CHAPTER 3 OPOS PROPERTIES, METHODS, EVENTS

Overview

The following steps depict the behavioral model of the OPOS driver and scanner.

1. The scanner reads encoded data from a label.
2. When the Control receives input, it queues a DataEvent.
3. If the AutoDisable property is TRUE, the Control is disabled when a DataEvent is queued.
4. The Control can deliver a queued DataEvent to the application when the DataEventEnabled property is TRUE. Just before delivering this event, the Control copies the data into properties and disables further data events by setting the DataEventEnabled property to FALSE. This causes the Control to queue subsequent input data while the application processes the current input and associated properties. When the application finishes the current input and is ready for more data, it re-enables events by setting DataEventEnabled to TRUE.
5. The Control queues an ErrorEvent (or events) if it encounters an error while gathering or processing input, and delivers this to the application when the DataEventEnabled property is TRUE.
6. The DataCount property contains the number of DataEvents queued by the Control.
7. Call the ClearInput method to delete all input that the Control queued.

Scanned data is placed into the property ScanData. If the application sets the property DecodeData to TRUE, the data is decoded into ScanDataLabel and ScanDataType.

Deviations from OPOS Specifications

The Motorola Scanner OPOS Driver includes several deviations from the OPOS specification for more flexibility. For example, the Claim method and Claimed property have significant deviations. According to the OPOS model, one control accesses only one physical device. The Motorola Scanner OPOS Driver allows access to multiple scanners simultaneously. Hence, a claim succeeds with one or more scanners. Also, several applications can share one scanner.

In addition to the Motorola Scanner OPOS Driver's architectural aspects, the following special behaviors occur:

- When there is no scanner connected to a cordless base, Motorola Scanner OPOS Driver considers the cordless base a scanner. Therefore a claim succeeds with a cordless base.
- In serial mode, a claim succeeds even when no scanner is connected to the port. In this case, it indicates the success of the port opening.
- Since the Motorola Scanner OPOS Driver supports multiple scanners, it implements the following OPOS functions with certain deviations.
- The following deviations are common to the *retrieveStatistics* method call:
  - When claiming multiple scanners, scanner details appear sequentially with comma separation. The order is the same for all scanner statistics.
  - When there are multiple scanners, OPOS schema validation can fail because some date fields contain comma-separated multiple dates.
  - Non-RSM scanners, including serial scanners in Wincor-Nixdorf RS-232 Mode B, do not provide Model Number and Serial number. These values appear as empty strings.

Supported Feature Set

This section describes the supported feature set per the OPOS specification.

Properties

Table 3-1 *Common Properties*

Property	Version	Type	Access	May Use After	Comments on Motorola Scanner Support
AutoDisable	1.2	Boolean	R/W	Open	Supported
BinaryConversion	1.2	Long	R/W	Open	Supported
CapCompareFirmwareVersion	1.9	Boolean	R	Open	Not supported
CapPowerReporting	1.3	Int	R	Open	Not supported
CapStatisticsReporting	1.8	Boolean	R	Open	Supported
CapUpdateFirmware	1.9	Boolean	R	Open	Not supported
CapUpdateStatistics	1.8	Boolean	R	Open	Supported
CheckHealthText	1.0	String	R	Open	Not supported. Always returns OPOS_E_ILLEGAL
Claimed	1.0	Boolean	R/W	Open	Supported (see *Deviations from OPOS Specifications on page 3-2*)
DataCount	1.2	Long	R	Open	Supported
DataEventEnabled	1.0	Boolean	R/W	Open	Supported
DeviceEnabled	1.0	Boolean	R/W	Open & Claim	Supported
FreezeEvents	1.0	Boolean	R/W	Open	Supported
OpenResult	1.5	Long	R	N/A	Supported
PowerNotify	1.3	Long	R/W	Open	Not supported. Always returns OPOS_E_ILLEGAL
PowerState	1.3	Long	R	Open	Supported
ResultCode	1.0	Long	R	N/A	Supported
ResultCodeExtended	1.0	Long	R	Open	Supported
State	1.0	Long	R	N/A	Supported
ControlObjectDescription	1.0	String	R	N/A	Supported
ControlObjectVersion	1.0	Long	R	N/A	Supported
ServiceObjectDescription	1.0	String	R	Open	Supported

Table 3-1 *Common Properties (Continued)*

Property	Version	Type	Access	May Use After	Comments on Motorola Scanner Support
ServiceObjectVersion	1.0	Long	R	Open	Supported
DeviceDescription	1.0	String	R	Open	Supported
DeviceName	1.0	String	R	Open	Supported

Table 3-2 *Specific Properties*

Property	Version	Type	Access	May Use After	Comments on Motorola Scanner Support
DecodeData	1.2	Boolean	R/W	Open	Supported
ScanData	1.0	String	R	Open	Supported
ScanDataLabel	1.2	String	R	Open	Supported
ScanDataType	1.2	Long	R	Open	Supported

Methods

Table 3-3 *Common Methods*

Method	Version	May Use After	Comments on Motorola Scanner Support
Open	1.0	N/A	Supported
Close	1.0	Open	Supported
ClaimDevice	1.0	Open	Supported (see *Deviations from OPOS Specifications on page 3-2*)
ReleaseDevice	1.0	Open & Claim	Supported
CheckHealth	1.0	Open, Claim & Enable	Limited support
ClearInput	1.0	Open & Claim	Supported
ClearInputProperties	1.10	Open & Claim	Supported
DirectIO	1.0	Open	Not supported. Always returns OPOS_E_ILLEGAL
compareFirmwareVersion	1.9	Open, Claim & Enable	Not supported
resetStatistic	1.8	Open, Claim & Enable	Supported
retrieveStatistics	1.8	Open, Claim & Enable	Supported
updateFirmware	1.9	Open, Claim & Enable	Not supported
updateStatistics	1.8	Open, Claim & Enable	Supported

Events

Table 3-4 *Events*

Event	Version	May Use After	Comments on Motorola Scanner Support
DataEvent	1.0	Open, Claim & Enable	Supported
DirectIOEvent	1.0	Open & Claim	Not supported
ErrorEvent	1.0	Open, Claim & Enable	Not supported
StatusUpdateEvent	1.3	Open, Claim & Enable	Not supported

CHAPTER 4 SAMPLE APPLICAITON (SCANNER OPOS TEST)

Overview

The Motorola Scanner OPOS Driver suite ships with a sample application that demonstrates all the OPOS operations on a connected Motorola scanner.

OPOS Sample Application (OPOS Test Utility)

The OPOS Test Utility allows you to simulate an application communicating with the Motorola Scanner OPOS Driver. This utility displays scanned data received from the scanner through the Motorola Scanner SDK Driver. Motorola Scanner SDK includes source code for this VC++ test utility.

OPOS Test Utility Window Functionality

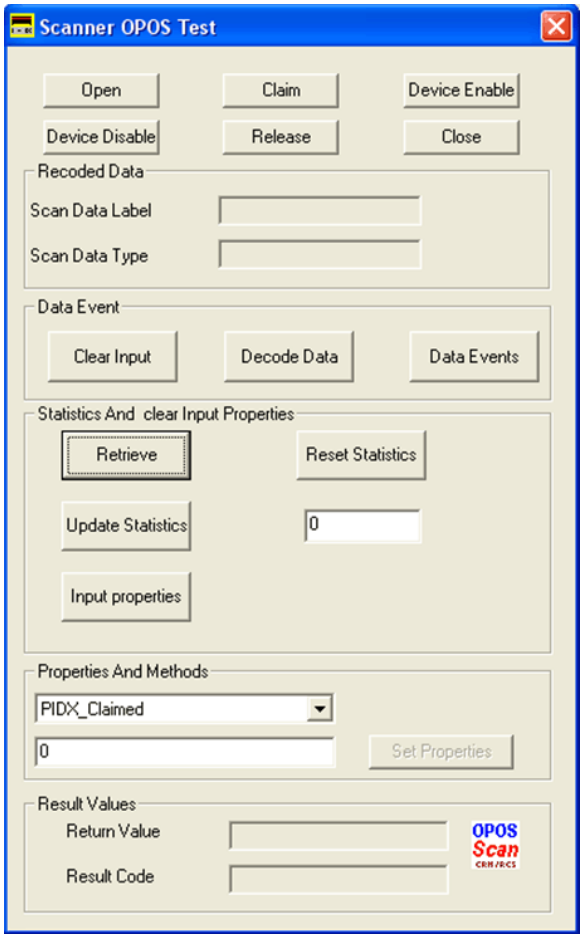


Figure 4-1 *Scanner OPOS Test Window*

Table 4-1 *Scanner OPOS Test Utility Button/Field Functionality*

Button/ Field/ Check Box	Description	Values	Code Sample
Open	Open Method	SYMBOL_SCANNER	`m_ctScanner.Open("SYMBOL_SCANNER")`
Claim	Claim the device with time out value.	-1, Any integer starting from zero	`m_ctScanner.Claim(1000)`
Device Enable	Enable the scanner. Must enable before using scanners.	N/A	`m_ctrScanner.SetDeviceEnabled(TRUE)`
Device Disable	Disable the scanner.	N/A	`m_ctrScanner.SetDeviceEnabled(FALSE)`

Table 4-1 *Scanner OPOS Test Utility Button/Field Functionality (Continued)*

Button/ Field/ Check Box	Description	Values	Code Sample
Release	Release the scanner.	N/A	`m_ctrScanner.ReleaseDevice()`
Close	Close the scanner.	N/A	`m_ctrScanner.CloseDevice()`
Recorded Data			
Scan Data Label	Label of the scan data.	N/A	`m_ctrScanner.GetScanDataLabel()`
Scan Data Type	Type of the scanned data. This is only a readable property.	N/A	`m_ctrScanner.GetScanDataType()`
Data Event			
Clear Input	Clear method. Clears the input data.	N/A	`m_ctrScanner.ClearInput()`
Decode Data	Set decode data enable.	N/A	`m_ctrScanner.SetDecodeData(1)`
Data Events	Set data event enabled. Must enable data event to get data.	N/A	`m_ctrlScanner.SetDataEventEnabled(1)`
Statistics And Clear Input Properties			
Retrieve	Retrieve statistic	"",M_,U_, GoodScanCount	`m_ctrlScanner.RetrieveStatistics(&test)`
Reset Statistics	Reset statistics	"",M_,U_, GoodScanCount	`m_ctrlScanner.ResetStatistics("GoodScanCount")`

Update Statistics	Update statistics	"",M_,U_, GoodScanCount	`CString strTemp;`
			`m_nGoodScanCount=100;`
			`strTemp.Format("GoodScanCount=%d", m_nGoodScanCount );`
			`m_ctrlScanner.UpdateStatistics(strTemp)`

Input Properties	Clear input properties	N/A	`m_ctrlScanner.ClearInputProperties()`
Properties And Methods			

Table 4-1 *Scanner OPOS Test Utility Button/Field Functionality (Continued)*

Button/ Field/ Check Box	Description	Values	Code Sample
Set Properties	Set the value of property to the given value.	N/A	`m_ctrScanner.SetFreezeEvents(0)` `m_ctrScanner.SetAutoDisable(1)`
Result Values			
Return Value	Return value of the last function call.	This is only a readable property.	
Result Code	Return value of result code.	This is only a readable property.	`m_ctrScanner.GetResultCode()`

Viewing Bar Code Data

To view bar code data using the Scanner OPOS Test Utility:

1. Scan the USB OPOS (Hand Held) bar code, SNAPI bar code or Wincor-Nixdorf RS-232 Mode B bar code *on page 2-3* to configure the scanner for the correct communication protocol.
2. Select C:\Program Files\Motorola Scanner\OPOS\Sample Applications\bin\Scanner-OPOS-Test.exe to launch the Scanner OPOS Test Utility.
3. Select **Open**. The *Open Service Object* window appears.
4. Select **Ok** to use the Motorola Scanner Service Object that the Scanner SDK Installshield setup program loaded on the PC.
5. Select **Claim**.
6. Select **Device Enable**.
7. Select **Data Events**.
8. Select **Decode Data**.
9. Scan the following sample bar code:

UPC-A

10. Select **Data Events** to view the scanned UPC-A bar code data. The bar code data that the driver processed appears in the *Scan Data Label* and *Scan Data Type* boxes:.

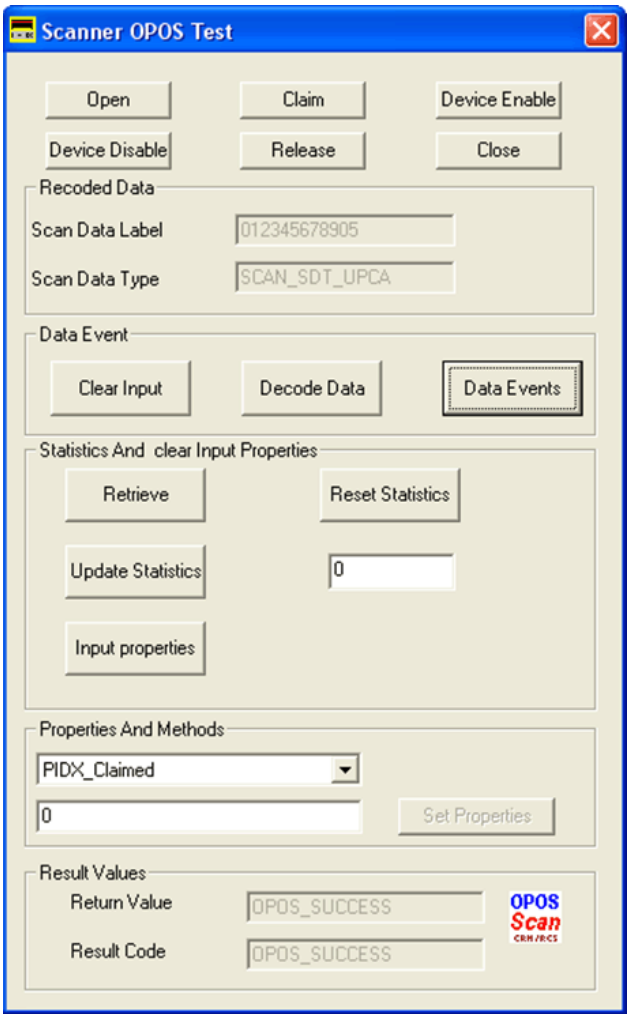


Figure 4-2 *Scanner OPOS Test Window - Scan Data*

11. Select **Clear Input** to clear the data from the *Scan Data Type* and *Scan Data Label* boxes.

12. To perform a second test, scan another bar code.

Getting and Setting OPOS Properties

This utility allows getting and setting the OPOS properties of the Motorola Scanner OPOS Driver via the *Properties And Methods* drop-down list.

To get and set the OPOS properties of the Motorola Scanner OPOS Driver:

1. Scan the USB OPOS (Hand Held) bar code, SNAPI barcode or Wincor-Nixdorf RS-232 Mode B bar code *on page 2-3* to configure the scanner for the correct communication protocol.
2. Select C:\Program Files\Motorola Scanner\OPOS\Sample Applications\bin\Scanner-OPOS-Test.exe to launch the Scanner OPOS Test Utility.
3. Select **Open**. The *Open Service Object* window appears.
4. Select **Ok** to use the Motorola Scanner Service Object that the SDK Installshield loaded on the PC.
5. Select **Claim**.
6. Select **Device Enable**.
7. Select a configurable (setable) property in the *Properties And Methods* drop-down list. You can configure some properties (e.g., AutoDisable, FreezeEvents); other properties are read only (e.g., ServiceObjectVersion, DataCount).
8. The current value of the OPOS driver appears in the edit box below the property selected in the list box. The values 1 and 0 represent true and false, respectively.
9. To change the configurable property, change the value in the edit box and select **Set Properties**. This updates the property with the new value.

Creating a Custom OPOS Sample Application

You may use any programming language to create a custom OPOS sample application. However, Microsoft supported languages are recommended (e.g., Visual Basic, Visual C++ or C#).

To create a custom OPOS sample application:

1. Create a project in the desired development environment.
2. Select (check) OPOS Scanner Control.
3. Add OPOSScanner.ocx to the project.
4. Drag and drop to the form/Dialog window.
5. Add a variable (handle) for the scanner control added to the form/Dialog window.
6. Call Open(), Claim() methods [e.g., Open ("SYMBOL_SCANNER"); Claim (2000);].
7. SetDeviceEnabled to TRUE.
8. Set Freeze Events property to FALSE [e.g., SetFreezeEvents (FALSE)].
9. SetDataEventsEnable to TRUE to get scan data events.
10. When done, SetDeviceEnabled to FALSE, Release() and Close() the service.
11. Call Device Disable property, release and close methods to close the connection.

Return Value and Result Code

When calling any method, check whether the return value is 0 (=OPOS_SUCCESS) to ensure the method is successful. Otherwise it returns an error code, which indicates the reason for the error. After setting property values, check that the result code returns 0 (=OPOS_SUCCESS), indicating success. If unsuccessful, it returns an error code.

Direct I/O

The OPOS driver does not support any direct I/O functions to the scanner. However, an application developer can get management access to an RSM-ready scanner through the Motorola Scanner SDK. Refer to the Motorola Scanner SDK Developer's Guide for more information.

Statistics Methods

The Motorola OPOS Driver supports the retrieveStatistics, resetStatistics, and updateStatistics methods. GoodScanCount is the only defined statistic in the Motorola OPOS Driver and can be used as a parameter for these methods.

Modified Claim Functionality

Model number, serial number and the Type (Scanner Host Mode) parameters are available in the system registry as configurable entries so that user can configure them according to the business requirement. Claiming a scanner compares the scanner details provided in system registry with the attached scanner properties. The claim is successful when they match.

Enter * to include anything for the particular entry. For example, enter * for the serial number to claim scanners with any serial number. Otherwise, the claim is successful only if the provided serial number and model number matches the present scanner.

For the model number and serial number, provide the exact value, or part of the string and a star (*). Do not enter a star (*) in the middle of a string; in this case, all data after star (*) is ignored. Provide a value and star as a model number (e.g., LS4208*) to accept all scanners starting with that model number (LS4208). Provide a star (*) for the serial number to accept all scanners regardless of serial number.

It is required add the full name of the each scanner type as comma or space delimited list to get the scanners Claimed. As an example to include SNAPI scanners and IBM Hand Held mode scanners the "Type" entry should be (USBIBMHID SNAPI). To include all scanner modes (Types) the value should be (ALL)

Since non-RSM scanners, including serial scanners in Wincor-Nixdorf RS232 Mode B, do not provide the model number and serial number, to claim these scanners set a " * " to both ModelNumber and SerialNumber registry values.

CHAPTER 5 SUPPORTED SYMBOLOGY TYPES VS. SCANNER MODE

Overview

This chapter provides a matrix of scanner modes and supported symbology types in each mode.

Supported Symbology Types vs. Scanner Mode

Table 5-1 *Supported Symbology Types vs. Scanner Modes*

Symbology		Scanner Mode		
Type	**Value**	**IBM HID**	**SNAPI**	**Nixdorf Mode B**
UPC-A	SCAN_SDT_UPCA	X	X	X
UPC-A with supplemental bar code	SCAN_SDT_UPCA_S	X	X	X
UPC-E	SCAN_SDT_UPCE	X	X	X
UPC-E with supplemental bar code	SCAN_SDT_UPCE_S	X	X	X
UPC-D1	SCAN_SDT_UPCD1	X	X	X
UPC-D2	SCAN_SDT_UPCD2	X	X	X
UPC-D3	SCAN_SDT_UPCD3	X	X	X
UPC-D4	SCAN_SDT_UPCD4	X	X	X
UPC-D5	SCAN_SDT_UPCD5	X	X	X
EAN 8 (=JAN 8)	SCAN_SDT_EAN8	X	X	X
JAN 8 (= EAN 8)	SCAN_SDT_JAN8	X	X	X
EAN 8 with supplemental barcode	SCAN_SDT_EAN8_S	X	X	X
EAN 13 (= JAN 13)	SCAN_SDT_EAN13	X	X	X
JAN 13 (= EAN 13)	SCAN_SDT_JAN13	X	X	X
EAN 13 with supplemental barcode	SCAN_SDT_EAN13_S	X	X	X
EAN-128	SCAN_SDT_EAN128	X	X	X
Standard (or discrete) 2 of 5	SCAN_SDT_TF	X	X	X
Interleaved 2 of 5	SCAN_SDT_ITF	X	X	X
Codabar	SCAN_SDT_Codabar	X	X	X
Code 39	SCAN_SDT_Code39	X	X	X
Code 128	SCAN_SDT_Code128	X	X	X
OCR "A"	SCAN_SDT_OCRA	X	X	-
OCR "B"	SCAN_SDT_OCRB	X	X	-
GS1 DataBar Omnidirectional (normal or stacked)	SCAN_SDT_GS1_DATABAR	X	X	-
GS1 DataBar Expanded (normal or stacked)	SCAN_SDT_GS1_DATABAR_E	X	X	-

Table 5-1 *Supported Symbology Types vs. Scanner Modes (Continued)*

Symbology		Scanner Mode		
Type	**Value**	**IBM HID**	**SNAPI**	**Nixdorf Mode B**
Composite Component A	SCAN_SDT_CCA	-	X	-
Composite Component B	SCAN_SDT_CCB	-	X	-
Composite Component C	SCAN_SDT_CCC	-	X	-
PDF 417	SCAN_SDT_PDF417	X	X	-
MAXICODE	SCAN_SDT_MAXICODE	X	X	-
Data Matrix	SCAN_SDT_DATAMATRIX	-	X	-
QR Code	SCAN_SDT_QRCODE	-	X	-
Micro QR Code	SCAN_SDT_UQRCODE	-	X	-
Aztec	SCAN_SDT_AZTEC	-	X	-
Micro PDF 417	SCAN_SDT_UPDF417	-	X	-

When the scanner is in Wincor-Nixdorf RS232 Mode B, the Motorola OPOS return value for the ScanDataType property differs from the expected value for the bar code types listed in *Table 5-2*.

Table 5-2 *Bar Code Types Not Accurately Identified in Wincor-Nixdorf RS232 Mode B*

Symbology Type	Expected Value	Motorola RSM OPOS Return Value	Comments
UPC-A with supplemental bar code	SCAN_SDT_UPCA_S	SCAN_SDT_UPCA	Nixdorf Mode B cannot distinguish UPCA since it identifies bar code types UPCA, UPCA_S, EAN13, EAN13_S, and BOOKLAND as one type.
UPC-E with supplemental bar code	SCAN_SDT_UPCE_S	SCAN_SDT_UPCE	Nixdorf Mode B identifies both bar code types UPCE and UPCE_S as UPCE.
EAN 8 with supplemental bar code	SCAN_SDT_EAN8_S	SCAN_SDT_EAN8	Nixdorf Mode B identifies both EAN8 and EAN8_S bar code types as EAN8.
EAN 13	SCAN_SDT_EAN13	SCAN_SDT_UPCA	Nixdorf Mode B cannot distinguish EAN 13 since it identifies bar code types UPCA, UPCA_S, EAN13, EAN13_S, and BOOKLAND as one type.
EAN 13 with supplemental bar code	SCAN_SDT_EAN13_S	SCAN_SDT_UPCA	Nixdorf Mode B cannot distinguish EAN 13_S since it identifies bar code types UPCA, UPCA_S, EAN13, EAN13_S, and BOOKLAND as one type.

INDEX

A

architecture 1-2
AutoDisable 3-3

B

BinaryConversion 3-3
bold text use in guide viii
bullets use in guide viii

C

CapCompareFrmwareVersion 3-3
CapPowerReporting 3-3
CapStatisticsReporting 3-3
CapUpdateFirmware 3-3
CapUpdateStatistics 3-3
CheckHealth 3-5
CheckHealthText 3-3
ClaimDevice 3-5
Claimed 3-3
ClearInput 3-5
ClearInputProperties 3-5
Close 3-5
common methods 3-5
common properties 3-3
compareFirmwareVersion 3-5
ControlObjectDescription 3-3
ControlObjectVersion 3-3
conventions
    notational viii

D

DataCount 3-3
DataEvent 3-6
DataEventEnabled 3-3
DecodeData 3-4
DeviceDescription 3-4
DeviceEnabled 3-3
DeviceName 3-4
DirectIO 3-5
DirectIOEvent 3-6
driver architecture 1-2

E

ErrorEvent 3-6
events 3-6
    DataEvent 3-6
    DirectIOEvent 3-6
    ErrorEvent 3-6
    StatusUpdateEvent 3-6

F

font use in guide viii
FreezeEvents 3-3

I

information, service ix
italics use in guide viii

M

methods
    CheckHealth 3-5
    ClaimDevice 3-5
    ClearInput 3-5
    ClearInputProperties 3-5
    Close 3-5
    compareFirmwareVersion 3-5

DirectIO 3-5
Open 3-5
ReleaseDevice 3-5
resetStatistic 3-5
retrieveStatistics 3-5
updateFirmware 3-5
updateStatistics 3-5

N

notational conventions viii

O

Open 3-5
OpenResult 3-3
OPOS
driver architecture 1-2

P

PowerNotify 3-3
PowerState 3-3
properties
AutoDisable 3-3
BinaryConversion 3-3
CapCompareFrmwareVersion 3-3
CapPowerReporting 3-3
CapStatisticsReporting 3-3
CapUpdateFirmware 3-3
CapUpdateStatistics 3-3
CheckHealthText 3-3
Claimed 3-3
ControlObjectDescription 3-3
ControlObjectVersion 3-3
DataCount 3-3
DataEventEnabled 3-3
DeviceDescription 3-4
DeviceEnabled 3-3
DeviceName 3-4
FreezeEvents 3-3
OpenResult 3-3
PowerNotify 3-3
PowerState 3-3
ResultCode 3-3
ResultCodeExtended 3-3
ServiceObjectDescription 3-3
ServiceObjectVersion 3-4
specific
DecodeData 3-4
ScanData 3-4
ScanDataLabel 3-4
ScanDataType 3-4
State 3-3

R

ReleaseDevice 3-5
resetStatistic 3-5
ResultCode 3-3
ResultCodeExtended 3-3
retrieveStatistics 3-5

S

ScanData 3-4
ScanDataLabel 3-4
ScanDataType 3-4
scanner mode 2-3, 5-2
service information ix
ServiceObjectDescription 3-3
ServiceObjectVersion 3-4
specific properties 3-4
State 3-3
StatusUpdateEvent 3-6
symbology types 5-2
symbology values 5-2

U

updateFirmware 3-5
updateStatistics 3-5

GLOSSARY

A

API. An interface by means of which one software component communicates with or controls another. Usually used to refer to services provided by one software component to another, usually via software interrupts or function calls

Aperture. The opening in an optical system defined by a lens or baffle that establishes the field of view.

Application Programming Interface. See **API**.

ANSI Terminal. A display terminal that follows commands in the ANSI standard terminal language. For example, it uses escape sequences to control the cursor, clear the screen and set colors. Communications programs support the ANSI terminal mode and often default to this terminal emulation for dial-up connections to online services.

ASCII. American Standard Code for Information Interchange. A 7 bit-plus-parity code representing 128 letters, numerals, punctuation marks and control characters. It is a standard data transmission code in the U.S.

Autodiscrimination. The ability of an interface controller to determine the code type of a scanned bar code. After this determination is made, the information content is decoded.

B

Bar. The dark element in a printed bar code symbol.

Bar Code. A pattern of variable-width bars and spaces which represents numeric or alphanumeric data in machine-readable form. The general format of a bar code symbol consists of a leading margin, start character, data or message character, check character (if any), stop character, and trailing margin. Within this framework, each recognizable symbology uses its own unique format. See **Symbology**.

Bar Code Density. The number of characters represented per unit of measurement (e.g., characters per inch).

Bar Height. The dimension of a bar measured perpendicular to the bar width.

Bar Width. Thickness of a bar measured from the edge closest to the symbol start character to the trailing edge of the same bar.

BIOS. Basic Input Output System. A collection of ROM-based code with a standard API used to interface with standard PC hardware.

Bit. Binary digit. One bit is the basic unit of binary information. Generally, eight consecutive bits compose one byte of data. The pattern of 0 and 1 values within the byte determines its meaning.

Bits per Second (bps). Bits transmitted or received.

Bit. Binary digit. One bit is the basic unit of binary information. Generally, eight consecutive bits compose one byte of data. The pattern of 0 and 1 values within the byte determines its meaning.

bps. See **Bits Per Second**.

Byte. On an addressable boundary, eight adjacent binary digits (0 and 1) combined in a pattern to represent a specific character or numeric value. Bits are numbered from the right, 0 through 7, with bit 0 the low-order bit. One byte in memory is used to store one ASCII character.

BOOTP. A protocol for remote booting of diskless devices. Assigns an IP address to a machine and may specify a boot file. The client sends a bootp request as a broadcast to the bootp server port (67) and the bootp server responds using the bootp client port (68). The bootp server must have a table of all devices, associated MAC addresses and IP addresses.

boot or boot-up. The process a computer goes through when it starts. During boot-up, the computer can run self-diagnostic tests and configure hardware and software.

Motorola Solutions, Inc.
One Motorola Plaza
Holtsville, New York 11742, USA
http://www.motorolasolutions.com

72E-149783-01 Revision A - May 2011