# MOTOROLA SCANNER JPOS DRIVER DEVELOPER'S GUIDE

# MOTOROLA SCANNER JPOS DRIVER DEVELOPER'S GUIDE

72E-149781-02

Revision A

March 2012

# Revision History

Changes to the original guide are listed below:

| Change | Date | Description |
|---|---|---|
| -01 Rev A | 5/2011 | Initial release. |
| -02 Rev A | 3/2012 | Updates for 64-bit. |

# TABLE OF CONTENTS

# ABOUT THIS GUIDE

## Introduction

This guide provides information about the Motorola Scanner JPOS Driver.

## Chapter Descriptions

Topics covered in this guide are as follows:

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

## 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 using the equipment, contact your facility's technical or systems support. If there is a problem with the equipment, they will contact the Motorola Solutions Global Customer Support Center at: www.motorolasolutions.com/support.

# CHAPTER 1 INTRODUCTION TO THE MOTOROLA SCANNER JPOS DRIVER

## Overview

The POS application is either a Java application or an applet that uses one or more JavaPOS devices. An application accesses the JavaPOS device through the JavaPOS Device Interface, which is specified by Java interfaces.

Each JavaPOS device is a combination of these components:

- JavaPOS Device Control is a Java class that provides the interface between the application and the device category. It contains no graphical component and is therefore invisible at runtime and conforms to the JavaBeans API. The Device Control was designed so that all implementations of a device category's control is compatible. The Device Control can be developed independently of a Device Service for the same device category. They may even be developed by different companies.
- JavaPOS Device Service is a Java class that is called by the Device Control through the JavaPOS Device Service Interface. The Device Service is used by the Device Control to implement JavaPOS-prescribed functionality for a physical device. It can also call special event methods provided by the Device Control to deliver events to the application.

## Motorola Scanner JPOS Driver Architecture

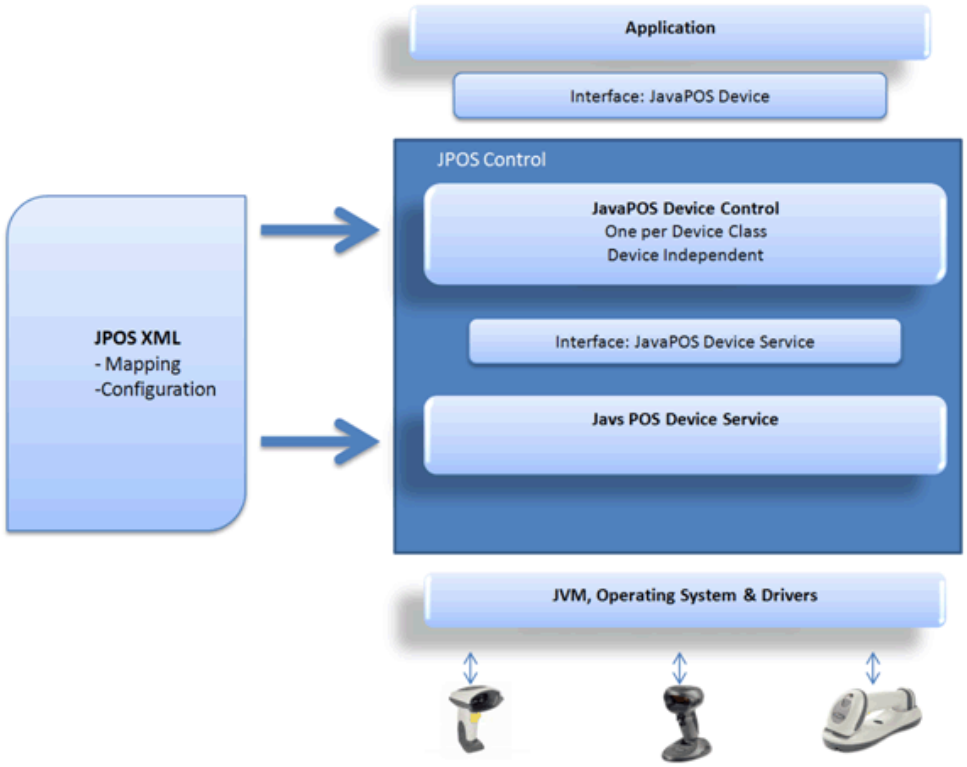



**Figure 1-1** *Motorola Scanner JPOS Driver Architecture*

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

- JavaPOS home page at http://www.javapos.com/
- UPOS home page at http://www.nrf-arts.org.

***NOTE*** The terms JavaPOS and JPOS are used interchangeably in this document.

# CHAPTER 2 INSTALLATION & CONFIGURATION

## Overview

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

For custom installation instructions on installing the Motorola scanner JPOS driver, refer to the Scanner SDK Developers Guide.

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

# Configuration

After a successful installation of the Motorola scanner SDK with the JPOS driver, an XML file named *jpos.xml* is created in the folder *C:\Program Files\Motorola Scanner\JPOS\Sample Applications\bin*.

The Motorola Scanner JPOS Driver reads *jpos.xml* to retrieve required configurations such as the baud rate for serial scanners and scanner filtering rules to form the logical scanner defined by the user. Each <*JposEntry logicalName="LogicalScannerName"*> tag defines a logical scanner. The following sample JPOSEntries represents serial, USB IBM Hand held and USB SNAPI logical devices.

## Wincor-Nixdorf Mode B RS-232 Scanners

```
<JposEntry logicalName="MotorolaScannerSerial">
  <creation factoryClass="com.symbol.jpos.SymScannerSvc112Factory"
serviceClass="com.symbol.jpos.SymScannerSvc112"/>
  <vendor name="Motorola , Inc." url="http://www.motorola.com"/>
  <jpos category="Scanner" version="1.12"/>
  <product description="Motorola Serial" name="Motorola Serial Scanner on COM1"
url="http://www.motorola.com"/>

  <!--Other non JavaPOS required properties-->
  <!--Scanner configuration-->
  <!--Comm port device name-->
  <prop name="port" value="COM1"/>
  <!--Baud rate, default=9600, valid values are: 9600, 19200, 38400, 57600, 115200-->
  <prop name="baud" value="9600"/>
  <!--Data bits, default=7, valid values are: 5, 6, 7, 8-->
  <prop name="databits" value="8"/>
  <!--Stop bits, default=1, valid values are: 1=1 stop bit, 2=2 stop bits, 3=1.5 stop
bits-->
  <prop name="stopbits" value="1"/>
  <!--Parity, default='O' (Odd), valid values are: 'N'one, 'O'dd, 'E'ven, 'M'ark,
'S'pace-->
  <prop name="parity" value="O"/>
  <!--Port mode, default='B', valid values are: 'B'=Nixdorf-B-->
  <prop name="mode" value="B"/>
  <!--UPC-A Length, default=13, indicates the number of digits generated by the scanner-->
  <!--for UPC-A labels based on the scanner's preamble and check digit settings.-->
  <!--The scanner service uses this value to determine when supplementals are present.-->
  <!--The default value of 13 assumes preamble is set to system character-->
  <!--and country code and that transmit check digit is enabled-->
  <prop name="UPCALength" value="13"/>
  <!--UPC-E Length, default=7, indicates the number of digits generated by the scanner-->
  <!--for UPC-E labels based on the scanner's preamble and check digit settings.-->
  <!--The scanner service uses this value to determine when supplementals are present.-->
  <!--The default value of 7 assumes preamble is set to system character-->
  <!--only and that transmit check digit is disabled-->
  <prop name="UPCELength" value="7"/>

  <!--Scanner type, default=0, valid values are: 7 - NixdorfUSB-->
  <prop name="ScannerType" value="7"/>
</JposEntry>
```

## USB IBM HAND HELD Scanners

```
<JposEntry logicalName="MotorolaScannerUSB">
  <creation factoryClass="com.symbol.jpos.SymScannerSvc112Factory"
serviceClass="com.symbol.jpos.SymScannerSvc112"/>
  <vendor name="Motorola, Inc." url="http://www.motorla.com"/>
  <jpos category="Scanner" version="1.12"/>
  <product description="Motorola USB" name="Motorola Scanner"
url="http://www.motorola.com"/>

  <!--Other non JavaPOS required properties-->
  <!--Comm port device name, must be 'USB' for USB scanner-->
  <prop name="port" value="USB"/>
  <!--Scanner type, default=0, valid values are: 6 - IBM HID-->
  <prop name="ScannerType" value="6"/>
 </JposEntry>
```

## USB SNAPI Scanners

```
<JposEntry logicalName="MotorolaScannerSNAPI">
  <creation factoryClass="com.symbol.jpos.SymScannerSvc112Factory"
serviceClass="com.symbol.jpos.SymScannerSvc112"/>
  <vendor name="Motorola, Inc." url="http://www.motorola.com"/>
  <jpos category="Scanner" version="1.12"/>
  <product description="Motorola SNAPI Scanner" name="Motorola Scanner"
url="http://www.motorola.com"/>

  <!--Other non JavaPOS required properties-->
  <!--Comm port device name, must be 'USB' for USB scanner-->
  <prop name="port" value="SNAPI"/>
  <!--Scanner type, default=0, valid values are: 2 - SNAPI-->
  <prop name="ScannerType" value="2"/>
 </JposEntry>
```

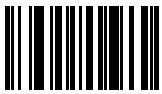
## All Scanners

```
<JposEntry logicalName="MotorolaAllScanners">
  <creation factoryClass="com.symbol.jpos.SymScannerSvc112Factory"
serviceClass="com.symbol.jpos.SymScannerSvc112"/>
  <vendor name="Motorola, Inc." url="http://www.motorola.com"/>
  <jpos category="Scanner" version="1.12"/>
  <product description="Motorola Serial/USB Scanner" name="Motorola Scanner"
url="http://www.motorola.com"/>

  <!--Scanner type, default=0, valid values are: 1- ALL Scanners-->
  <prop name="ScannerType" value="1"/>
 </JposEntry>
```

## 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.

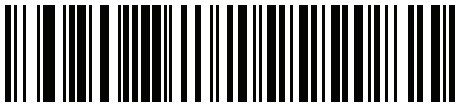


**Set All Defaults**

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

### USB Communication Protocol

**USB (IBM Hand Held)**



**USB SNAPI**

### RS-232 Communication Protocol

**Wincor-Nixdorf RS-232 Mode B**

# CHAPTER 3 JPOS PROPERTIES, METHODS, EVENTS

## Overview

The following steps depict the behavioral model of the JPOS 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 JPOS Specifications

- When there is no scanner connected to a cordless base, the Motorola Scanner JPOS 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.

# Supported Feature Set

This section describes the supported feature set per the JPOS 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 | Not 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 |
| Claimed | 1.0 | Boolean | R/W | Open | Supported (see *Deviations from JPOS Specifications on page 3-1*) |
| DataCount | 1.2 | Int32 | 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 | Not supported |
| PowerNotify | 1.3 | Long | R/W | Open | Not supported |
| PowerState | 1.3 | Int32 | R | Open | Supported |
| ResultCode | 1.0 | Long | R | N/A | Not supported |
| ResultCodeExtended | 1.0 | Long | R | Open | Not supported |
| State | 1.0 | Int32 | R | N/A | Supported |
| ControlObjectDescription | 1.0 | String | R | N/A | Supported |
| ControlObjectVersion | 1.0 | Int32 | R | N/A | Supported |
| ServiceObjectDescription | 1.0 | String | R | Open | Supported |
| ServiceObjectVersion | 1.0 | Int32 | 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 | Int32 | 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 JPOS Specifications on page 3-1*) |
| ReleaseDevice | 1.0 | Open & Claim | Supported |
| CheckHealth | 1.0 | Open, Claim & Enable | Not supported |
| ClearInput | 1.0 | Open & Claim | Supported |
| ClearInputProperties | 1.10 | Open & Claim | Supported |
| DirectIO | 1.0 | Open | Not supported |
| compareFirmwareVersion | 1.9 | Open, Claim & Enable | Not supported |
| resetStatistic | 1.8 | Open, Claim & Enable | Not supported |
| retrieveStatistics | 1.8 | Open, Claim & Enable | Supported |
| updateFirmware | 1.9 | Open, Claim & Enable | Not supported |
| updateStatistics | 1.8 | Open, Claim & Enable | Not 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 JPOS TEST)

## Overview

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

## JPOS Sample Application (JPOS Test Utility)

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

## JPOS Test Utility Window Functionality

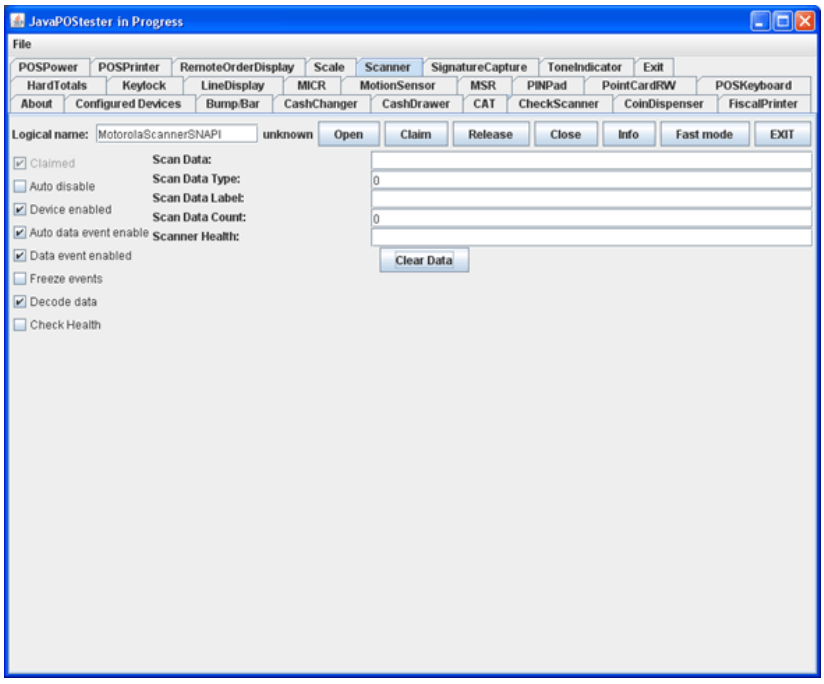



**Figure 4-1** *Scanner JPOS Test Window*

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

| Button/ Field/ Check Box | Description | Values | Code Sample |
|---|---|---|---|
| **Open** | Open Method. | MotorolaScannerSNAPI | `scanner.open(MotorolaScannerSNAPI);` |
| **Claim** | Claim the device with time out value. | -1, Any integer starting from zero | `scanner.claim(1000);` |
| *Device Enable/Disable* | Enable or Disable the scanner. Must enable before using scanners. | N/A | Enable:<br>`scanner.setDeviceEnabled(true);`<br>Disable:<br>`scanner.setDeviceEnabled(false);` |
| **Release** | Release the scanner. | N/A | `scanner.release();` |
| **Close** | Close the scanner. | N/A | `scanner.close();` |
| **Scan Data** | | | |
| Scan Data Type | Type of the scanned data. This is only a readable property. | N/A | `scanner.getScanDataType();` |
| Scan Data Label | Label of the scan data. | N/A | `scanner.getScanDataLabel();` |
| Clear Data | Clear method. Clears the input data. | N/A | |
| **Properties And Methods** | | | |
| Auto disable | Set the Auto disable property. | N/A | `scanner.setAutoDisable(false);` |
| Data Event Enabled | Set data event enabled. Must enable data event to get data. | N/A | `scanner.setDataEventEnabled(true);` |
| Freeze Events | Set the Freeze Events property. | N/A | `scanner.setFreezeEvents(false);` |
| Decode Data | Set decode data enable. | N/A | `scanner.setDecodeData(true);` |

***NOTE*** Additional documentation about the *Scanner JPOS Test Window* is available at http://postest.sourceforge.net/.

## Viewing Bar Code Data

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

1. Scan the USB (IBM Hand Held) bar code, SNAPI bar code or Wincor-Nixdorf RS-232 Mode B bar code *on page 2-4* to configure the scanner for the correct communication protocol.
2. Select C:\Program Files\Motorola Scanner\Scanner SDK\JPOS\Sample Applications\bin\POSTest.bat to launch the Scanner JPOS Test Utility.
3. Select **Open** after entering the logical name.
4. Select **Claim**.
5. Select **Device Enable**.
6. Select **Data Events**.
7. Select **Decode Data**.
8. Scan the following sample bar code:


**UPC-A**

9. The bar code data the driver processed appears in the *Scan Data Type* and *Scan Data Label* boxes.

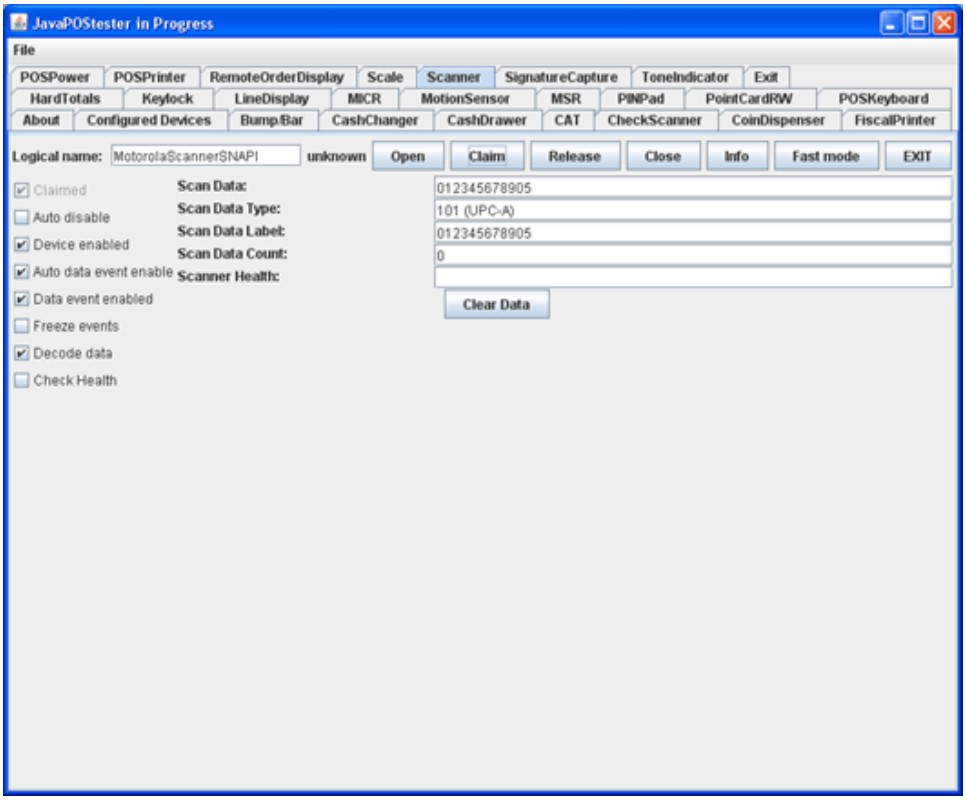



**Figure 4-2** *Scanner JPOS Test Window - Scan Data*

10. Select **Clear Data** to clear the data from the *Scan Data Type* and *Scan Data Label* boxes.
11. To perform a second test, scan another bar code.

## Return Value

When calling any method, check whether the return value is 0 (=JPOS_SUCCESS) to ensure the method is successful. Otherwise it returns an error code, which indicates the reason for the error.

## Direct I/O

The JPOS 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.

# 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 JPOS 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 JPOS 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

# 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-149781-02 Revision A - March 2012