J a v a P O S D r i v e r K i t

OPERATION MANUAL

This JavaPOS Driver Kit OPERATION MANUAL (hereinafter referred to as “the GUIDE”) describes the procedures and precautions for using the JavaPOS Driver Kit (hereinafter referred to as “the Kit”).

The GUIDE assumes that the reader is familiar with the following:
- General characteristics of POS peripheral devices
- Java terminology and architecture
- Java for Retail POS (JavaPOS for short) Programmer’s Guide

Notes:
- It is prohibited to use or duplicate a part or whole of the GUIDE without the permission of Toshiba TEC Corporation.
- The GUIDE is subject to change without prior notice.

* Windows and Windows XP, WEPOS and POSReady are registered trademarks of Microsoft Corporation in the United States and/or other countries.
The official name of Windows is the “Microsoft Windows Operating System”.

* Java and JavaPOS are trademarks of Sun Microsystems in the United States and/or other countries.

Table of Contents

Introduction

The Kit (JavaPOS Driver Kit) provides the JavaPOS Device Service to be used to develop an application for key lock (Keylock), cash drawer (Drawer), magnetic stripe reader (MSR), line display (LineDisplay), scanner (Scanner) and POS printer (POSPrinter).

Overview of the MANUAL

The GUIDE consists of the following three steps, and explains the demo program enclosed in the Kit. Please follow the steps below:

Step 1. **How to Build An Operating Environment**

This step describes the method to build an environment for using the JavaPOS Device Service.

Step 2. **How to Check Performance of the JavaPOS Device Service**

This step describes the method to test whether or not the test program for checking performance of the JavaPOS Device Service (CheckHealth.jar) successfully runs.

Step 3. **How to Use The JavaPOS Device Service**

This step describes the method to create a unique application using the JavaPOS Device Service (Keylock, Drawer, MSR, LineDisplay, POSPrinter).

Example of Creating An Application Using the JavaPOS Device Service

This chapter explains the method to operate the JavaPOS Device Service using an example. The demo program described here is a sample code which uses the JavaPOS Device Service to create the application.

1. How to Build An Operating Environment

This chapter describes the method to build an environment where the JavaPOS Device Service operates. Please take this step (Step 1) first, then go to Step 2 (Chapter 2 “How to Check Performance of the JavaPOS Device Service”) or Step 3 (Chapter 3 “How to Use the JavaPOS Device Service”).

Supported Product

- ST-B10

Operating Environment

Performance of the JavaPOS Device Service was checked under the following environment:

Operating system:	Windows XP SP3, WEPOS, POSReady2009
	SUSE Linux Enterprise Desktop 11
JavaRuntime:	JRE1.4.2
JavaPOS:	JavaPOS 1.11

Installation of Java Runtime Environment

Download the file from the following web site and implement the Java Runtime Environment.
http://java.sun.com/products/archive/j2se/1.4.2_16/

Setup for some Linux drivers

It is a required setup when using the driver of the following device.
When the following device drivers are used, execute this procedure in advance.

- iButton driver for Linux
- PS/2 POSKeyboard driver for Linux
- USB POSKeybaord driver for Linux

1. Patch application to Keyboard driver of Linux Kernel
The source file of a keyboard driver is rewritten using “tec_kbd-2.6.27.patch” file.
- /usr/src/linux/drivers/input/keyboard/atkbd.c
- /usr/src/linux/drivers/input/serio/i8042.c
- /usr/src/linux/drivers/char/keyboard.

Copy the following files to “/home/tec/tecdrv/” from “Driver” → “Linux Keyboard patch” folder in this Kit.
- tec_kbd-2.6.27.patch

Execute the following command. Then, a patch is applied and a source file can be rewritten.

```
]# cp -p /home/tec/tecdrv/tec_kbd-2.6.27.patch /usr/src/linux/
]# cd /usr/src/linux/
]# patch -p0 < tec_kbd-2.6.27.patch
```

2. Patch application to w1 driver of Linux Kernel for TTEC iButton driver
The source file of a w1 driver is rewritten using “tec_w1-2.6.27.patch” file.
- /usr/src/linux/drivers/w1/masters/ds2490.c

Copy the following files to “/home/tec/tecdrv/” from “Driver” → “Linux iButton Driver” → “patch” folder in this Kit.
- w1-patch-install.sh
- ds2490.h
- tec_w1-2.6.27.patch

Execute the following procedure, when using the iButton driver developed by TTEC.

```
]# /home/tec/tecdrv/w1-patch-install.sh
```

Rebuild a kernel, after these two procedures are completed.
Cautions : Rebuild a kernel takes several hours.

3. Rebuild the kernel

Execute the following commands in order from the top:

```
]# cd   /usr/src/linux/        ⇨ Movement to a directory with the source file of a kernel.
]# make  oldconfig             ⇨ Obtains configuration information of the kernel in operation.
]# make  clean                 ⇨ Deletes all interim files.
]# make                        ⇨ Complies the kernel and driver modules.
]# make  install               ⇨ Installs the kernel.
]# make  modules_install       ⇨ Installs the drivers.
```

Finally, restart the operating system.

4. Install the Linux Keyboard compatible Driver

If a keyboard compatible driver is installed, "setkeycodes" command can be executed even when a PS/2 keyboard has not been connected.

Note: Restarting the operating system clears this setting. The procedure must be performed whenever the operating system is restarted.

Copy the following files to "/home/tec/tecdrv/" from "Driver" → "Linux compat kbd" folder in this Kit.

- compat_keyb.ko

Execute the following command to install a Keyboard compatible driver.

```
]# insmod /home/tec/tecdrv/compat_keyb.ko
```

Installation of iButton Driver

<Windows>

Download the file from the following web site and implement the 1-Wire Drivers.
http://japan.maxim-ic.com/products/ibutton/software/tmex/index.cfm

<Linux>
Copy the following files to "/home/tec/tecdrv/" from "Driver" → "Linux iButton Driver" → "Driver" folder in this Kit.

- load_usbibutton
- usbibutton.ko

Installation of module

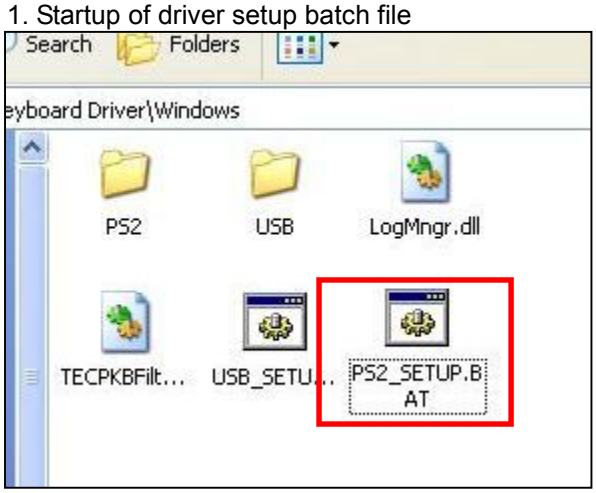
```
]# insmod /home/tec/tecdrv/compat_keyb.ko

]# cd /home/tec/tecdrv/
]# chmod 777 load_usbibutton
]# ./load_usbibutton
]# setkeycodes 0x68 93
```

Installation of PS/2 POSKeyboard Driver

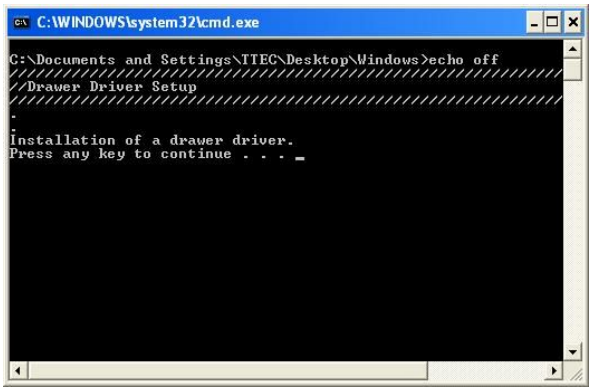
<Windows>

1. Startup of driver setup batch file



Select “Driver” → “Keyboard Driver” →”Windows”, then double-click on the “PS2_STUP.BAT” file to execute.

2. Start of installation



Press any key to start the installation.

3. Permission to add registry

A confirmation window appears. Click on the [Yes] button.

4. Result of registry addition

When the registry has been successfully added, the window shown above appears. Click on the [OK] button.

5. Permission to install the driver

When the window shown above appears, click on the [Continue Anyway] button.

6. Confirmation of installation

Start the Device Manager and check that the Keyboard Filter driver has been successfully installed. Then, open the system32 folder (C:¥WINDOWS¥system32) and make sure that the following modules have been copied.

- POSESC32.dll
- mkmgr.exe
- LogMngr.dll
- KeyMonHk.dll
- TECUSBPKBFilterJNI.dll

<Linux>

Execute the following commands in order to install the POS keyboard driver.
This procedure is necessary to operate MCR and Keylockattached to PKBST-50.

[Supported Product]
PKBST-50

Note: Restarting the operating system clears this setting. The procedure must be performed whenever the operating system is restarted.

1. Installation of module

```
]# insmod /home/tec/tecdrv/poskbd.ko
```

2. Creation of device file

```
]# mknod /dev/poskbd c 243 0
```

3. Keycode configuration
[MCR]

```
]# setkeycodes 0x61 121
```

[Keylock]

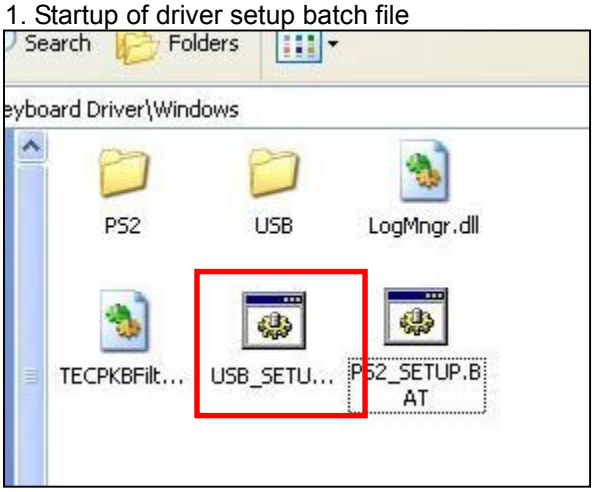
```
]# setkeycodes 0x63 123
```

Installation of USB POSKeyboard Driver

<Windows>

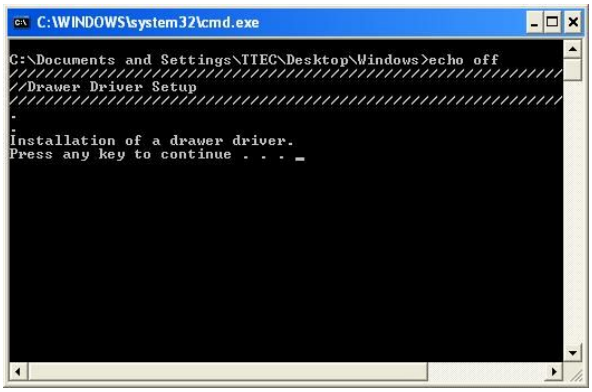
Before this installation, execute “Installation of TECUSB” procedure.

1. Startup of driver setup batch file



Select “Driver” → “Keyboard Driver” →”Windows”, then double-click on the “USB_SETUP.BAT” file to execute.

2. Start of installation



Press any key to start the installation.

3. Permission to add registry

A confirmation window appears. Click on the [Yes] button.

4. Result of registry addition

When the registry has been successfully added, the window shown above appears. Click on the [OK] button.

5. Confirmation of installation

Open the system32 folder (C:¥WINDOWS¥system32) and make sure that the following modules have been copied.

- POSESC32.dll
- UsbKbMgr.exe
- LogMngr.dll
- raslibc.dll
- TECUSBPKBFilterJNI.dll

<Linux>

Before this installation, execute “Installation of TECUSB” procedure.

Execute the following commands in order to install the USB POS keyboard driver.
This procedure is necessary to operate MCR and Keylockattached to PKBST-52.

[Supported Product]
PKBST-52

Note: Restarting the operating system clears this setting. The procedure must be performed whenever the operating system is restarted.

1. Installation of module

```
]# insmod /home/tec/tecdrv/usbposkbd.ko
```

2. Keycode configuration
[MCR]

```
]# setkeycodes 0x61 121
```

[Keylock]

```
]# setkeycodes 0x63 123
```

Installation of Drawer Driver

<Windows>

1. Startup of driver setup batch file

Select “Driver” → “CashDrawer Driver” → “Windows”, then double-click on the SETUP.BAT file to execute.

*1 DRVDRW.sys is a CashDrawer driver which runs on Windows. TECCashDrawerJni.dll is a library file which is used to access the Windows CashDrawer driver from Java.

2. Start of installation

Press any key to start the installation.

3. Permission to add registry

A confirmation window appears. Click on the [Yes] button.

4. Result of registry addition

When the registry has been successfully added, the window shown above appears. Click on the [OK] button.

5. Permission to install the driver

When the window shown above appears, click on the [Continue Anyway] button.

6. Confirmation of installation

Start the Device Manager and check that the CashDrawer driver has been successfully installed. Then, open the system32 folder (C:¥WINDOWS¥system32) and make sure that TECCashDrawerJni.dll has been copied.

< Linux>

1. Copy of driver module

Select “Driver” → “CashDrawer Driver” → “Linux” → “Driver”. Copy the drw.ko driver module to any desired location.

2. Installation of CashDrawer driver

```
]# insmod /home/tec/tecdrv/drw.ko port=0x448 postype=2
]# mknod /dev/drw c 242 0
```

Execute the above commands to install the driver.
(The above is an example when drw.ko has been copied to /home/tec/tecdrv.)

*The shaded value is the ST-B10 drawer port address. As the port address differs for each model, please check it with the specifications of each model.

*1 The 1st line installs the module and the 2nd line creates a device file.
*2 This setup is cleared whenever the operating system is rebooted. This command must be executed every time the operating system is rebooted

3. Confirmation of install

```
]# lsmod
```

Execute the above command to make sure the driver has been successfully installed.

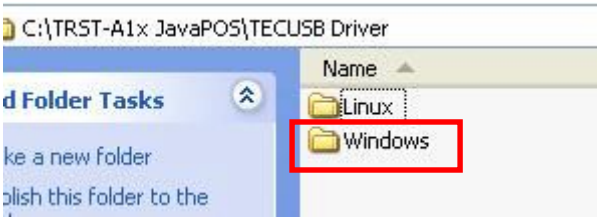
4. Generation of link file

Copy the libTECCashDrawerJni.so.0.0 to the location you want to run the program. Then generate a link file.

Installation of TECUSB Driver

<Windows>

1. Copy of driver module



Copy an “Driver” → “TECUSB Driver” →”Windows” folder in the suitable place.

2. Execute of Batch file

Carry out “TECUSB_LIBRARY_SETUP.BAT” in the folder which I stored by procedure 1.
After practice, the following file is copied by a folder of “¥Windows¥system32”.

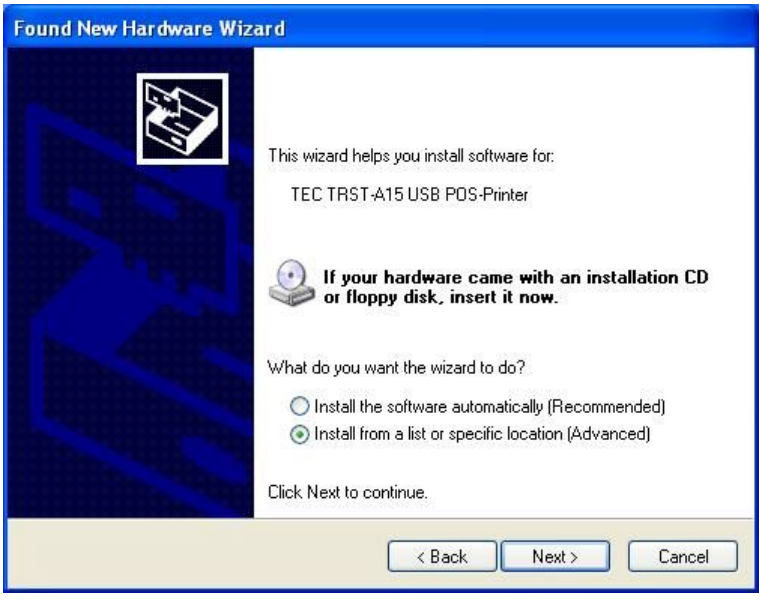
- TECUSB.dll
- LogMngr.dll
- TECUSBJNI.dll
- TECUSBPM.exe

Cautions: When it failed in a copy, copy the files to each directory manual operation.

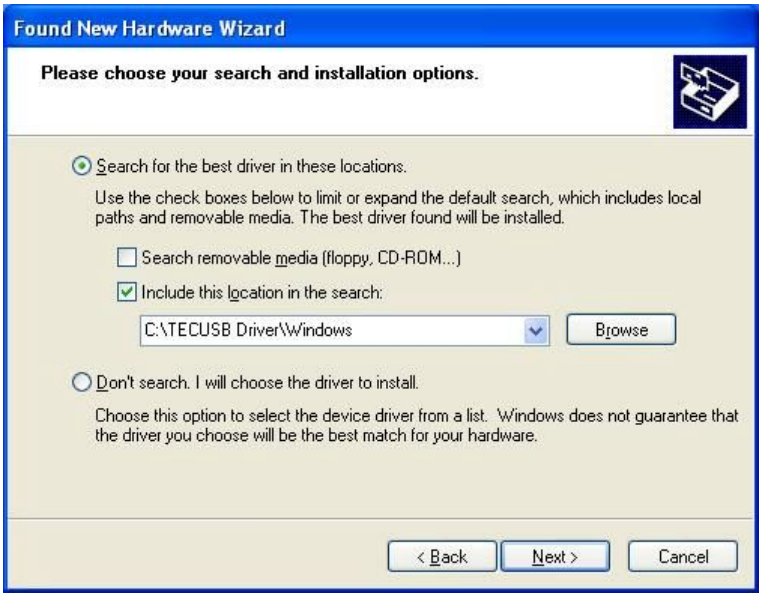
“Root directory of project
ex) A folder same as “CheckHealth.bat”
- TECUSBPM.exe

3. Installation of TRST-A1x TECUSB driver
Connect TRST-A1x by USB and turn on a power supply.
The following dialogue is displayed.

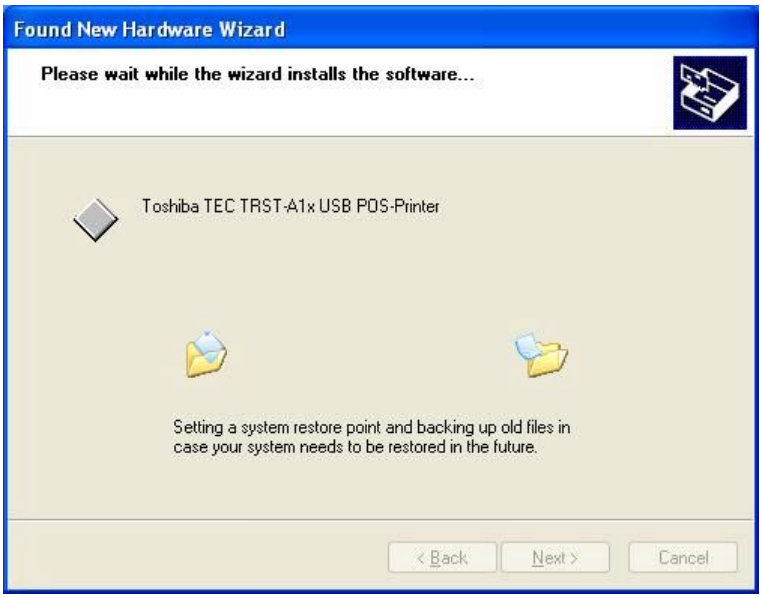
Choose “No, not this time”. Click on the “Next >” button.



Choose “Install from a list or specific location [Advanced]”. Click on the ”Next >” button.

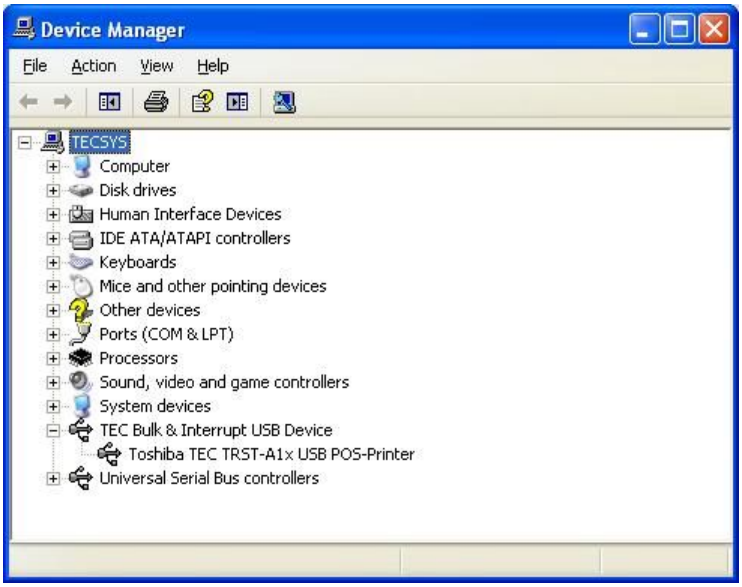


Exclude a check box of “Search removable media [floppy, CD-ROM...]”.
Choose check box of “Include this location in the search.”.
Click on the "Browse" button. And appoint a folder with the "TecUSBDEx.INF" file.
Click on the “Next >” button.



The installation of the TECUSB driver is started.

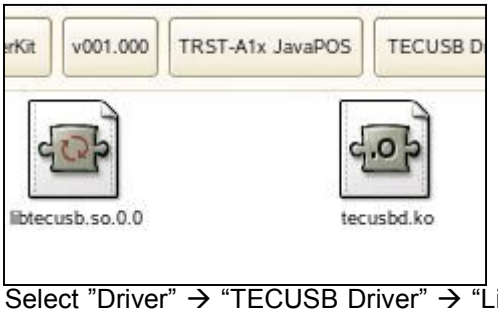
If an above screen is displayed, it is installation completion.
Click on the ”Finish” button.



Finally start device manager. And confirm that it is installed as above.

< Linux>

1. Copy of driver module



Select ”Driver” → “TECUSB Driver” → “Linux”. Copy the “tecusbd.ko” and “libtecusb.so.0.0” module to any desired location.
* tecusbd.ko is a TECUSB driver which runs on Linux.

2. Installation of driver

```
]# insmod /home/tec/tecdrv/tecusbd.ko
```

Execute the above commands to install the driver.
(The above is an example when tecusbd.ko has been copied to /home/tec/tecdrv.)

*1 This setup is cleared whenever the operating system is rebooted. This command must be executed every time the operating system is rebooted

3. Confirmation of install

```
]# lsmod
```

Execute the above command to make sure the driver has been successfully installed.

4. Installation of library

```
]# cp –p /home/tec/tecdrv/libtecusb.so.0.0 /usr/lib/
]# ldconfig –n /usr/lib/
]# ln –s /usr/lib/libtecusb.so.0.0 /usr/lib/libtecusb.so
```

Execute the above commands to install the library.
(The above is an example when libtecusb.so.0.0 has been copied to /home/tec/tecdrv.)

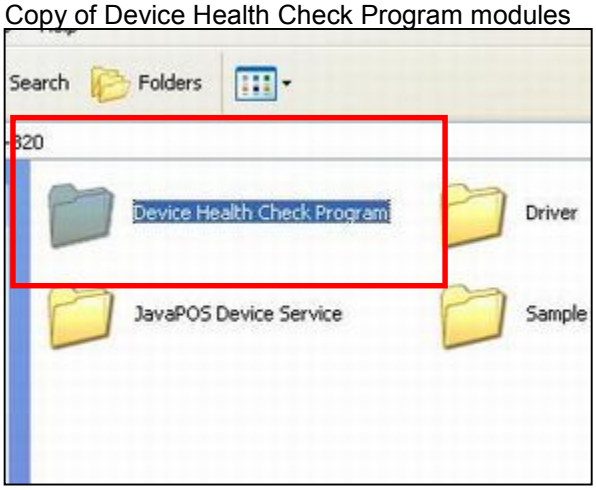
2. How to Check Performance of the JavaPOS Device Service

This chapter describes the method to check performance of the JavaPOS Device Service, assuming that the operating environment described in Chapter 1 has been built up.
Here, the setup method is referred to as “PREPARE” and the operation method of the Device Health Check Program as “OPERATION”.

In this chapter, the device health check method is explained for the following devices:
- LineDisplay
- iButton
- Keylock
- CashDrawer
- MSR
- POSPrinter

PREPARE

Copy of Device Health Check Program modules

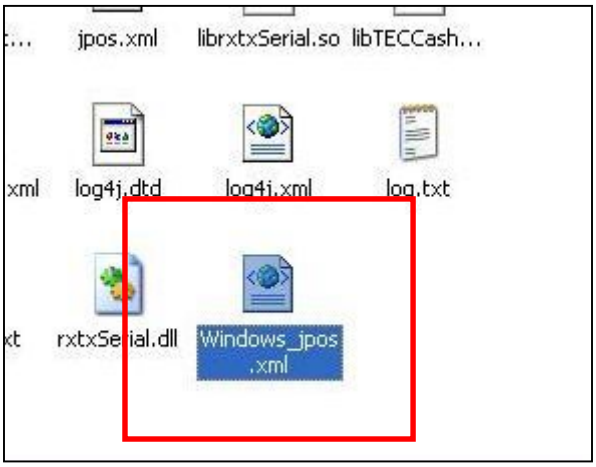


Open the Kit CD, then copy the Device Health Check Program folder to a desired location in the local computer.

* The subsequent procedures are separately explained for Windows and Linux below.

<Windows>

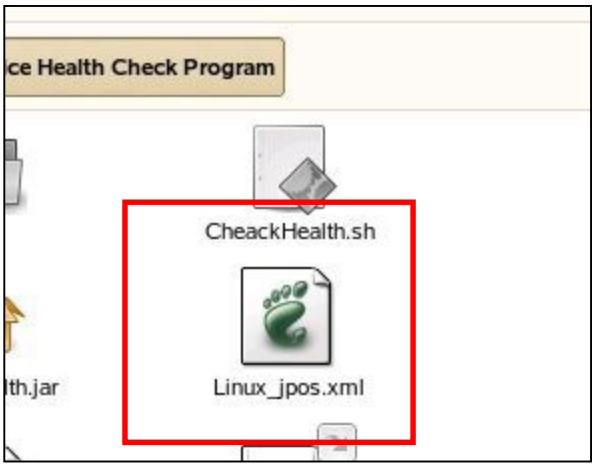
Change of setup file name



Change the name of the Windows_jpos.xml file in the Device Health Check Program folder to jpos.xml.

<Linux>

1. Change of setup file name



Change the name of the Linux_jpos.xml file in the Device Health Check Program folder to jpos.xml.

2. Grant of execute authority

```
]# chmod 775 /home/tec/Device Health Check Program/CheackHealth.sh
```

Execute the above command to grant the CheackHealth.sh file an execute authority.
(The above is an example when the Device Health Check Program folder has been copied to /home/tec.)

3. Generation of CashDrawerJni link file

```
]# ln –s /home/tec/ Device Health Check Program/libTECCashDrawerJni.so.0.0
   /home/tec/ Device Health Check Program/libTECCashDrawerJni.so
```

Execute the above command to generate link file.
The above command is needed to use drawer driver.

4. . Generation of PKBFilterJNI link file

```
]# ln –s /home/tec/ Device Health Check Program/ libTECPKBFilterJNI.so.0.0
        /home/tec/ Device Health Check Program/ libTECPKBFilterJNI.so
```

5. Generation of TECUSBJNI link file

```
]# ln –s /home/tec/ Device Health Check Program/ libTECUSBJNI.so.0.0
        /home/tec/ Device Health Check Program/ libTECUSBJNI.so
```

6. Generation of TECUSB link file

```
]# cp -p /home/tec/tecdrv/libtecusb.so.0.0 /usr/lib/
]# ldconfig -n /usr/lib/
]# ln -s /usr/lib/libtecusb.so.0.0 /usr/lib/libtecusb.so
```

Default value

Default value of major parameters is as follows. To change the default value, please refer to the chapter, “3. How to Use the JavaPOS Device Service” in the GUIDE or the setup method in the Application User Manual of each device service.

Category	LogicalName	deviceBus	portName
Keylock	iButton	USB	-
Keylock	PKBST-5x	PS2	-
Keylock	PKBST-52	USB	
MSR	MCRST	PS2	-
MSR	MCRRMS	RS232	COM5
MSR	MCRST-52	USB	-
LineDisplay	LIUST-A10	RS232	COM4
LineDisplay	WD-111	RS232	COM4
CashDrawer	DRWST	-	DRW1

Category	LogicalName	deviceBus	portName	fontSize
CashDrawer	WindowsSerialCashDrawer -TRSTA1x	RS232	COM1	1
CashDrawer	WindowsSerialCashDrawer -TRSTA00	RS232	COM1	2
POSPrinter	WindowsSerialPOSPrinter -TRSTA1x	RS232	COM1	1
POSPrinter	WindowsSerialPOSPrinter -TRSTA00	RS232	COM1	2

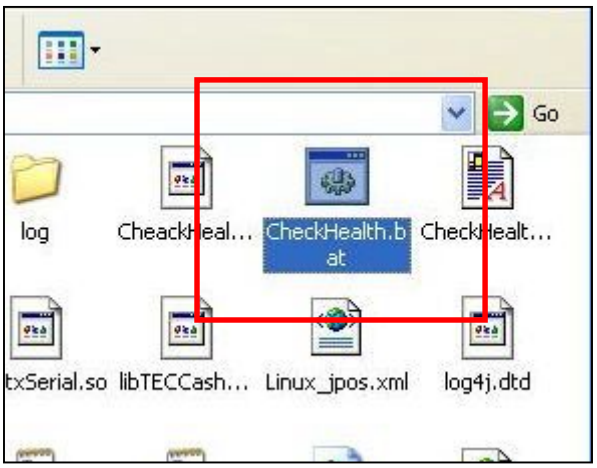
Category	LogicalName	deviceBus	productID	fontSize
CashDrawer	USBCashDrawer-TRSTA1x-QM	USB	61	1
CashDrawer	USBCashDrawer-TRSTA1x-CN	USB	70	1
CashDrawer	USBCashDrawer-TRSTA00	USB	82	2
POSPrinter	USBPOSPrinter-TRSTA1x-QM	USB	61	1
POSPrinter	USBPOSPrinter-TRSTA1x-CN	USB	70	1
POSPrinter	USBPOSPrinter-TRSTA00	USB	82	2

Category	LogicalName	deviceBus	IP Address
CashDrawer	LANCashDrawer	LAN	x.x.x.x
POSPrinter	LANPOSPrinter-1	LAN	x.x.x.x
POSPrinter	LANPOSPrinter-2	LAN	x.x.x.x
POSPrinter	LANPOSPrinter-3	LAN	x.x.x.x

OPERATION

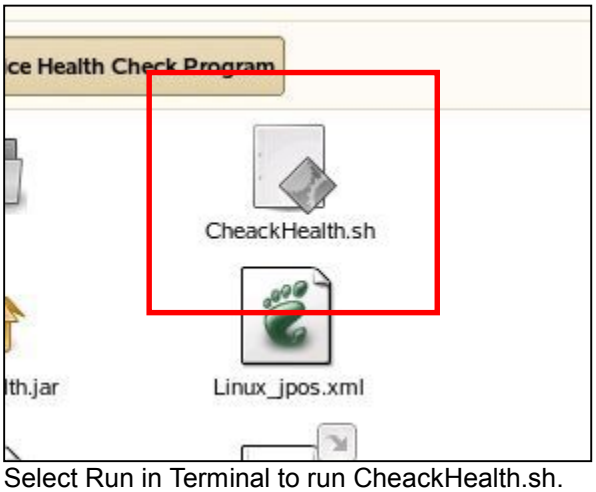
1 Startup of program

<Windows>



Execute the CheckHealth.bat file.

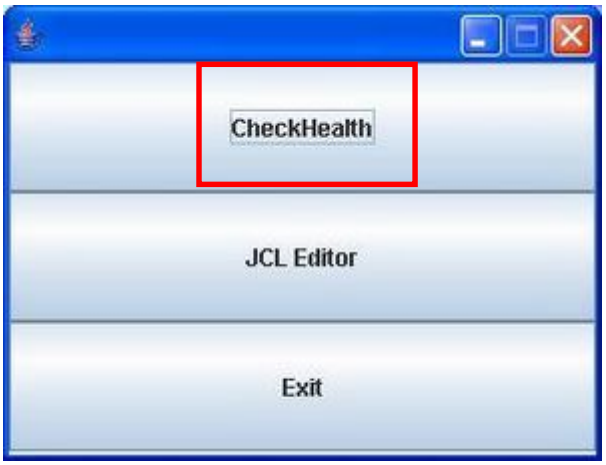
<Linux>



Select Run in Terminal to run CheackHealth.sh.

* The subsequent OPERATION applies both to Windows and Linux.

2 Execution of Device Health Check program

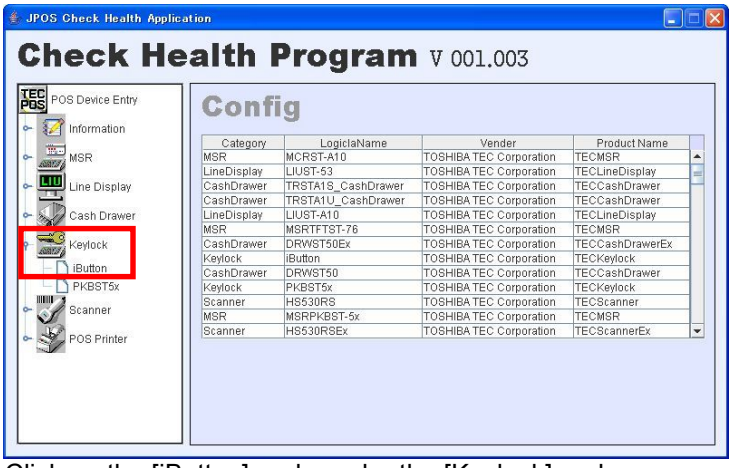


Click on the [CheckHealth] button at the top.

* Note the subsequent OPERATION differs for Keylock, Drawer, MSR, LineDisplay, and Scanner.
* The functions of the JCL Editor are not used this time.

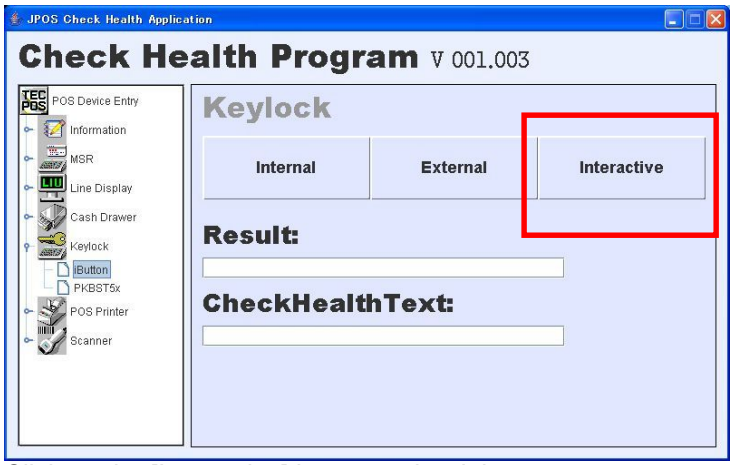
iButton

A-1 Keylock panel display



Click on the [iButton] node under the [Keylock] node.

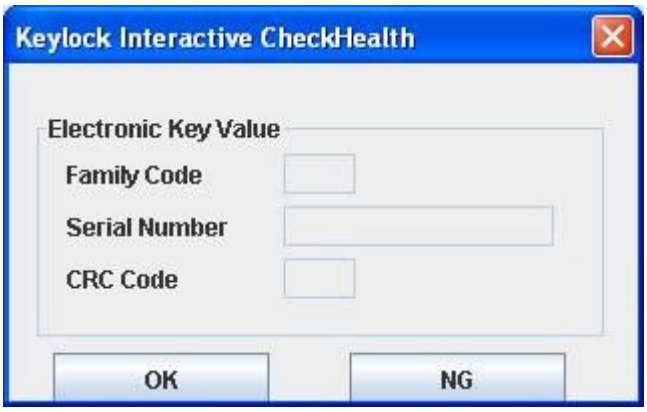
A-2 Call to the Interactive Check Health method



Click on the [Interactive] button at the right.

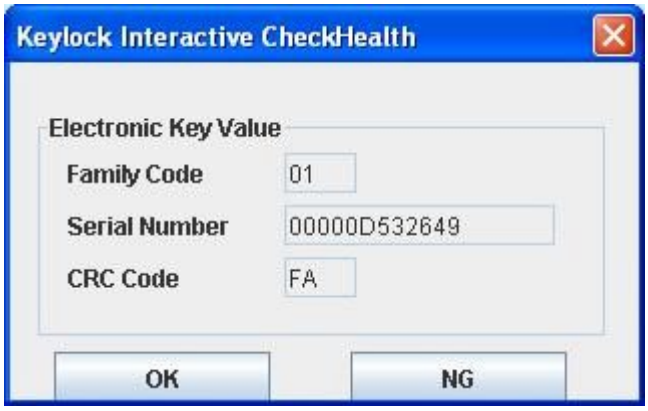
iButton

A-3 Installation of iButton



When the above window appears, make the iButton touch to the button contacts.

A-4 Display of iButton data

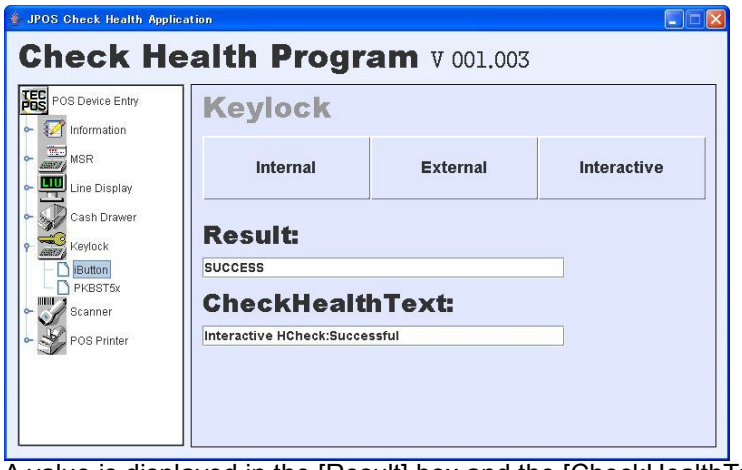


The text boxes in the window show data stored in the iButton.

* To exit, click on the [OK] or [NG] button.

iButton

A-5 Display of result



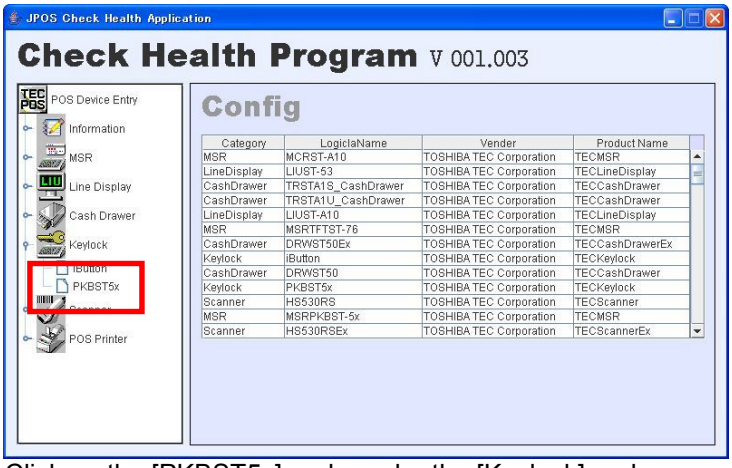
A value is displayed in the [Result] box and the [CheckHealthText] Text box.
Either of the following two value combinations will be displayed in these boxes:

- When exited with the [OK] button in Step A-4.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Successful

- When exited with the [NG] button in Step A-4.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Error

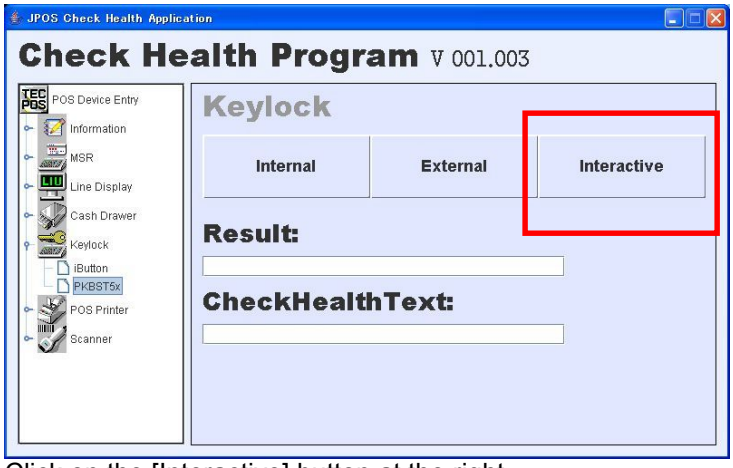
Keylock

B-1 Keylock panel display



Click on the [PKBST5x] node under the [Keylock] node.

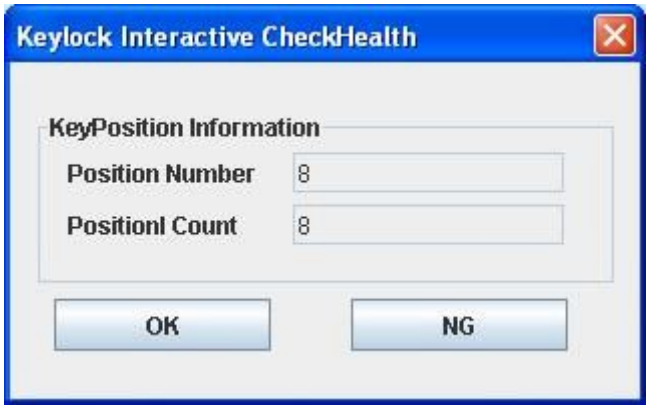
B-2 Call to the Interactive Check Health method



Click on the [Interactive] button at the right.

Keylock

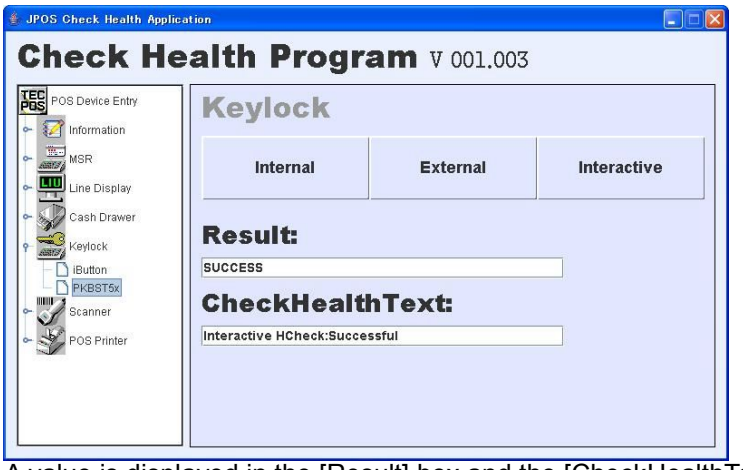
B-3 Installation of keylock



When the above window appears, change key position.

* To exit, click on the [OK] or [NG] button.

B-4 Display of result



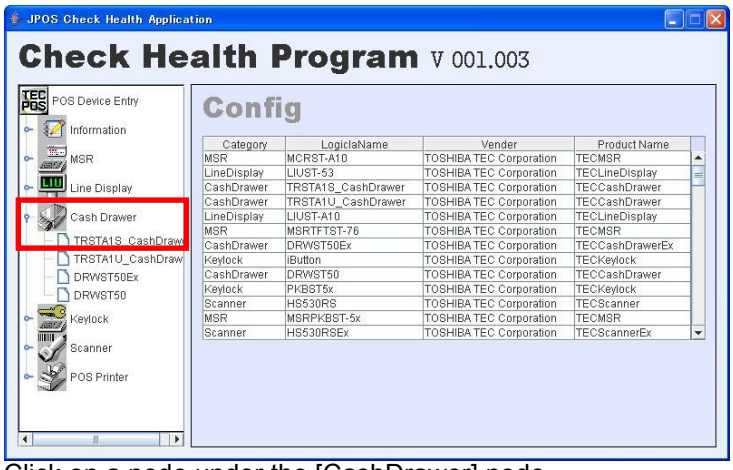
A value is displayed in the [Result] box and the [CheckHealthText] Text box.
Either of the following two value combinations will be displayed in these boxes:

- When exited with the [OK] button in Step B-4.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Successful

- When exited with the [NG] button in Step B-4.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Error

CashDrawer

C-1 Drawer panel display

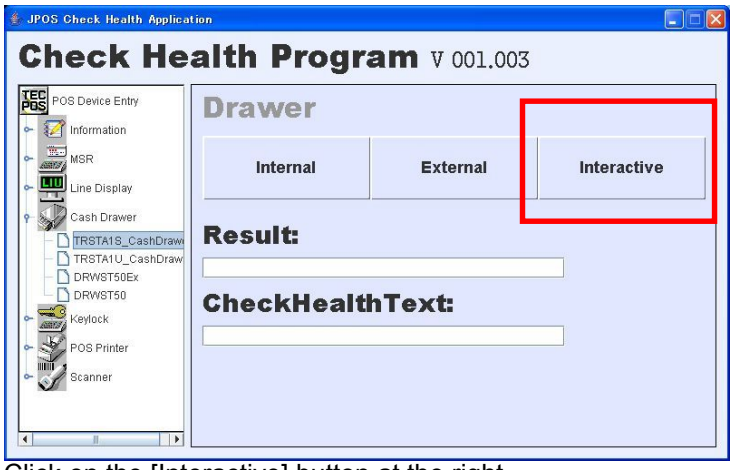


Click on a node under the [CashDrawer] node.

CashDrawer has the following three kinds.

- DRWST5x
- DRWST5xEx
- EPSON Drawer (TRSTA1xUDRW or TRSTA1xSDRW)

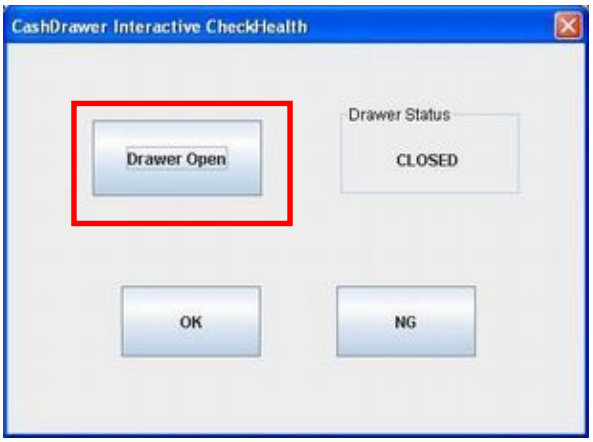
C-2 Call to the Interactive Check Health method



Click on the [Interactive] button at the right.

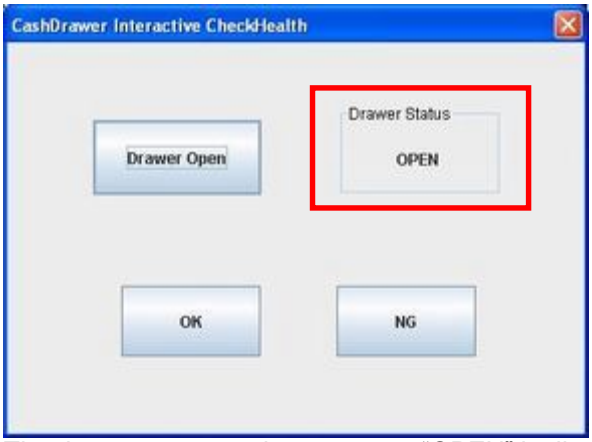
CashDrawer

C-3 Open the drawer



Click on the [Drawer Open] button.

C-4 Get status

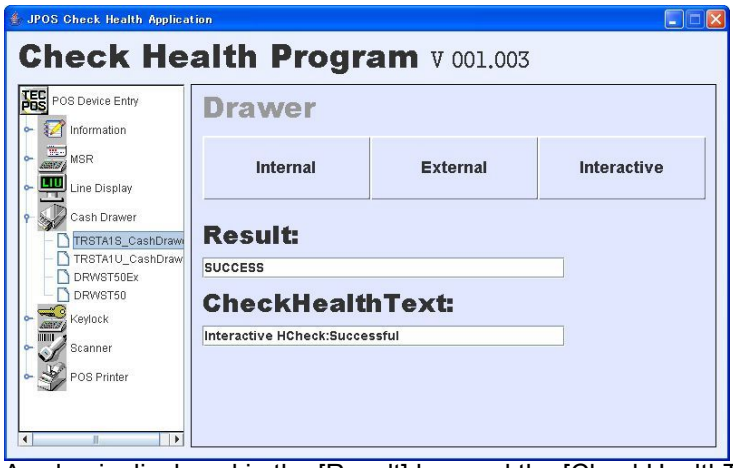


The drawer opens and a message, “OPEN” is displayed on the [Drawer Status] box at the upper right in the window on the screen.

* To exit, click on the [OK] or [NG] button.

CashDrawer

C–5 Display of result



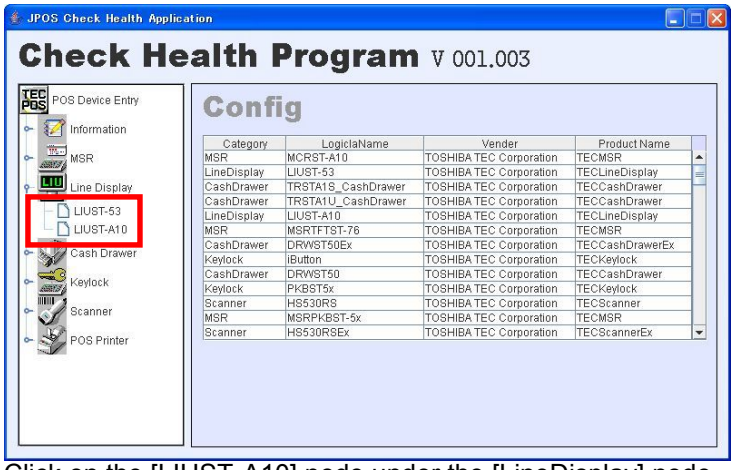
A value is displayed in the [Result] box and the [CheckHealthText] Text box.
Either of the following two value combinations will be displayed in these boxes:

- When exited with the [OK] button in Step C-4.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Successful

- When exited with the [NG] button in Step C-4.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Error

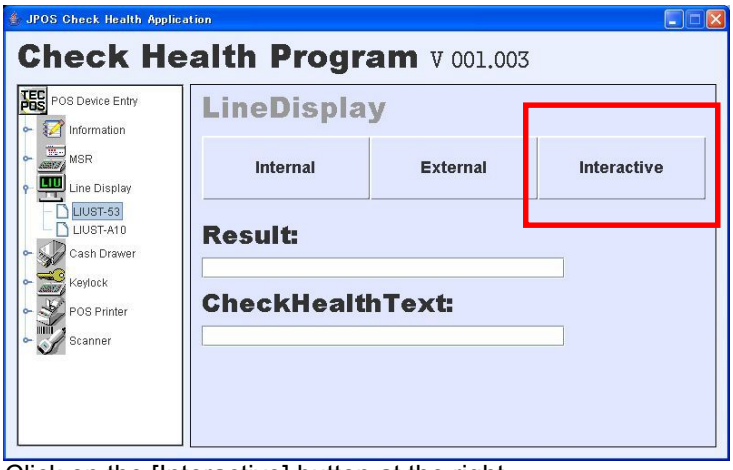
LineDisplay

D-1 LineDisplay panel display



Click on the [LIUST-A10] node under the [LineDisplay] node.

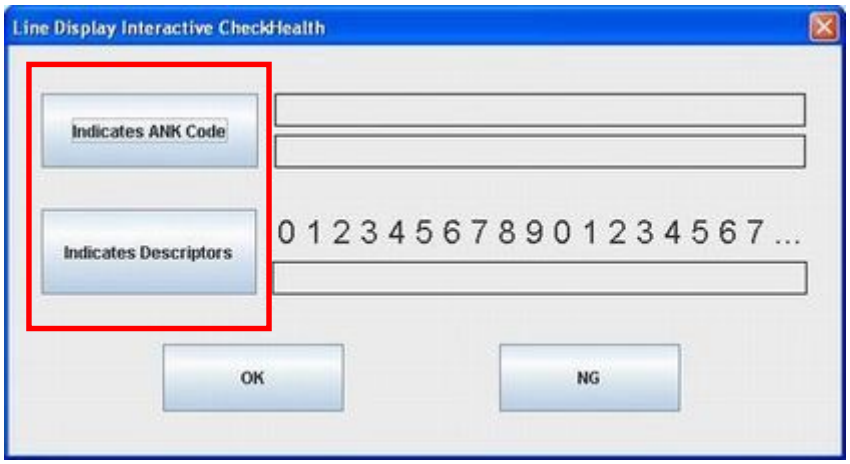
D-2 Call to the Interactive Check Health method



Click on the [Interactive] button at the right.

LineDisplay

D-3 Display of LineDisplay



There are the following two CheckHealth functions for LineDisplay.

Indicates ANK Code

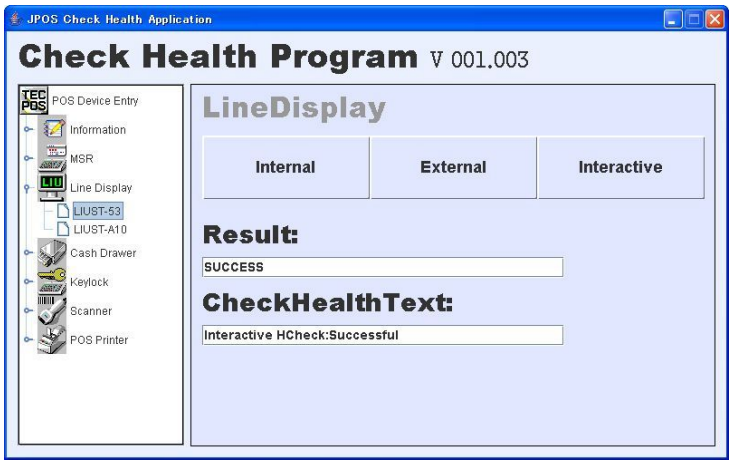
Click on the [Indicates ANK Code] button at the upper left, and the same content, displayed in the two boxes on the right side of the button, is also displayed on the line display device.

Indicates Descriptors

Click on the [Indicates Descriptors] button at the centre left, and a descriptor is displayed in the line display device at a location indicated by the value in the box on the right side of the button.

LineDisplay

D–4 Display of result



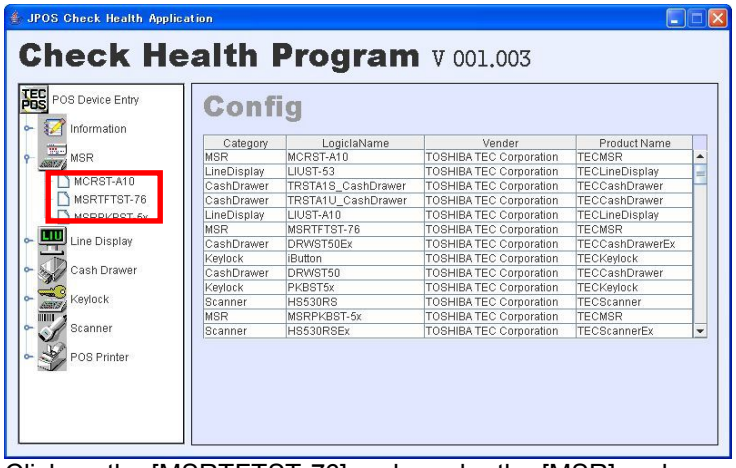
A value is displayed in the [Result] box and the [CheckHealthText] Text box.
Either of the following two value combinations will be displayed in these boxes:

- When exited with the [OK] button in Step D-3.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Successful

- When exited with the [NG] button in Step D-3.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Error

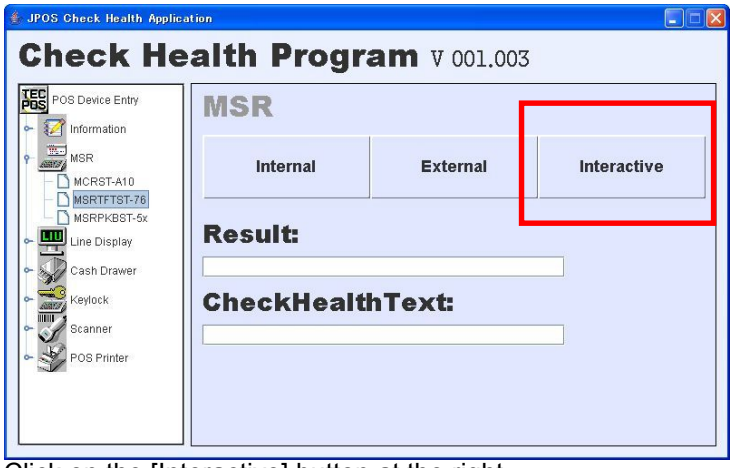
MSR

E-1 MSR panel display



Click on the [MSRTFTST-76] node under the [MSR] node.

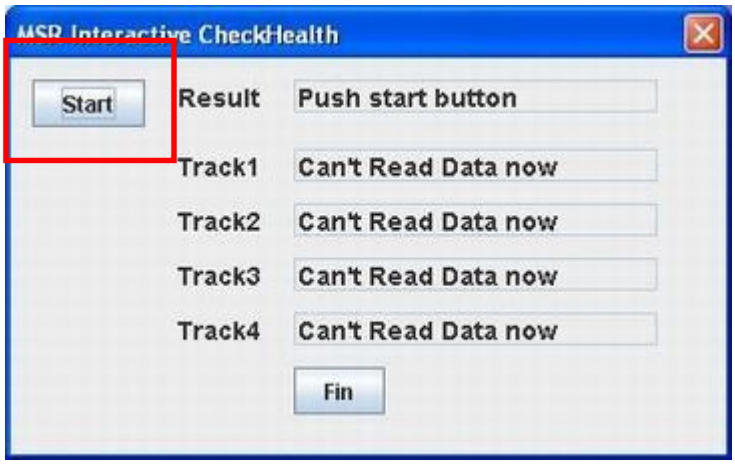
E-2 Call to the Interactive Check Health method



Click on the [Interactive] button at the right.

MSR

E-3 Startup of card data read mode



Click on the [Start] button.

E-4 Read of card data

A message "Waiting" is displayed in the text boxes. Swipe a card.

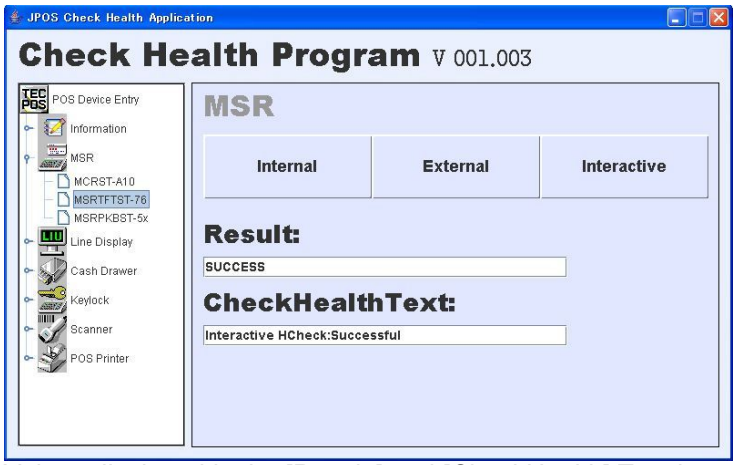
MSR

<Reading succeeded.>

<Reading failed.>

* To exit, click on the [Fin] button.

E-5 Display of result



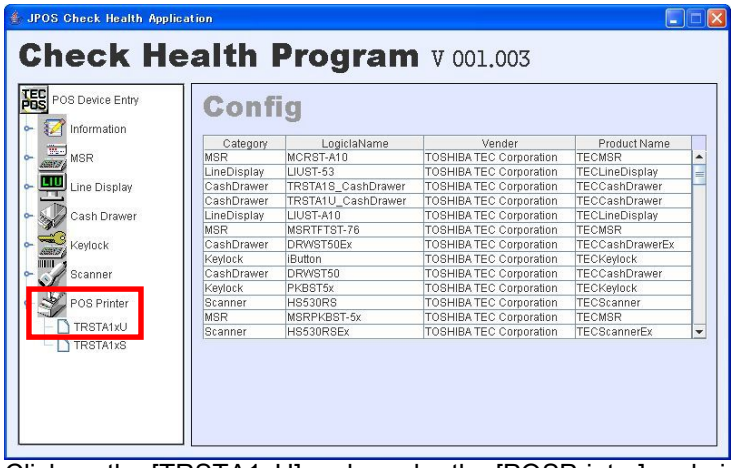
Values displayed in the [Result:] and [CheckHealth] Text boxes differ depending on the reading result. There are the following two value combinations.

- When a reading operation in Step E-4 did not fail even once.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Successful

- When a reading operation in Step E-4 failed at least once.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Error

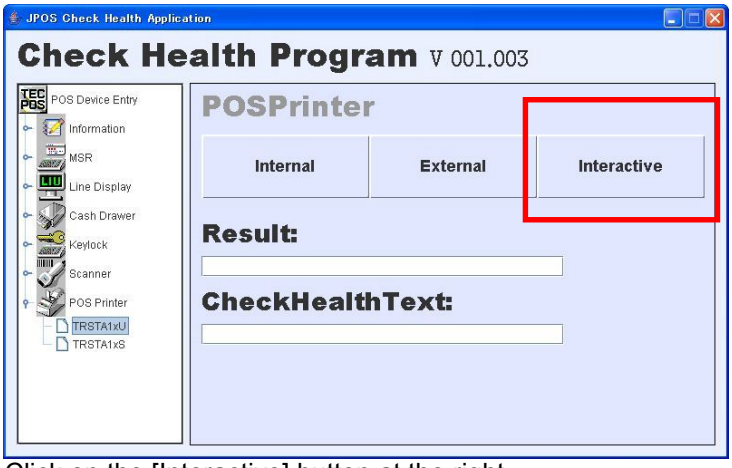
POSPrinter

F-1 POSPrinter panel display



Click on the [TRSTA1xU] node under the [POSPrinter] node in the case of using USBPOSPrinter.
Click on the [TRSTA1xS] node under the [POSPrinter] node in the case of using SerialPOSPrinter.

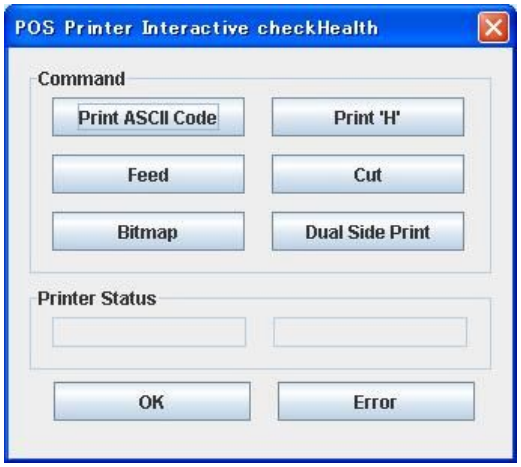
F-2 Call to the Interactive Check Health method



Click on the [Interactive] button at the right.

POSPrinter

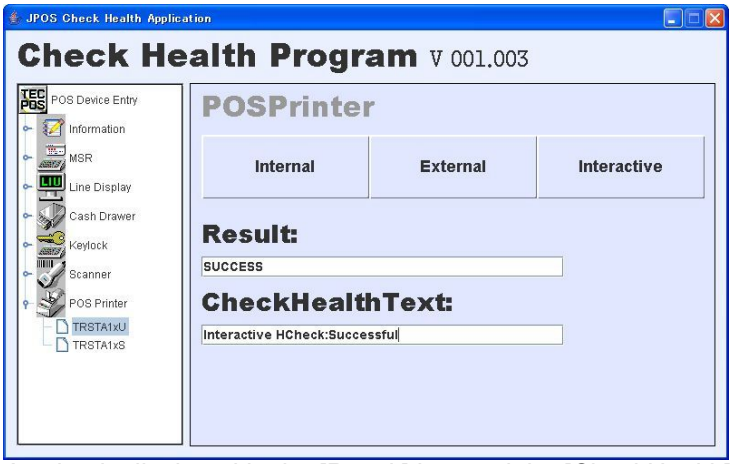
F-3 Execute of Print



When the above window appears, execute each function.

* To exit, click on the [OK] or [NG] button.

F-4 Display of result



A value is displayed in the [Result] box and the [CheckHealth] Text box.
Either of the following two value combinations will be displayed in these boxes:

- When exited with the [OK] button in Step G-3.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Successful

- When exited with the [NG] button in Step G-3.
  Result : SUCCESS
  CheckHealthText : Interactive Hcheck:Error

3. How to Use the JavaPOS Device Service

This chapter describes the setups required to use the JavaPOS Device Service, assuming that the operating environment described in Chapter 1 has been built up.

Required files

In order to use the JavaPOS Device Service provided by the Kit, the following files are required besides the library file.

- jpos111.jar
- jpos.xml
- log4j.xml
- log4j.dtd
- log4j-1.2.12.jar
- commons-logging.jar
- RXTXcomm.jar
- swing-layout-1.0.3.jar
- xercesImpl.jar
- xml-apis.jar
- jcl_editor.jar
- rxtxSerial.dll(*1)
- librxtxSerial.so(*2)
- rxtxParallel.dll(*1)
- librxtxParallel.so(*2)
- TECCashDrawerJni.dll(*3)
- libTECCashDrawerJni.so(*4)
- libTECiButtonJni.so(*5)
- JimiProClasses.jar
- TECUSB.dll(*6)
- libtecusb.so.0.0(*7)
- TECUSBJNI.dll(*6)
- libTECUSBJNI.so.0.0(*7)
- TECUSBPM.exe(*3)
- libTECPKBFilterJNI.so.0.0(*8)
- libtecusb.so.0.0(*9)
- libTECUSBJNI.so.0.0(*10)

*1 Required only for Windows
*2 Required only for Linux
*3 Required only for using CashDrawer Device Service under Windows
*4 Required only for using CashDrawer Device Service under Linux
*5 Required only for using iButton Device Service under Linux
*6 Required only for using POSPrinter USB Device Service under Windows
*7 Required only for using POSPrinter USB Device Service under Linux
*8 Required only for using POSKeyboard under Linux
*9 Required only for using TECUSB Device under Linux
*10 Required only for using TECUSB Device under Linux

Description of Files

A destination to save a file may differ depending on the development environment. The following explanation is based on the development using the NetBeans5.5.

Jpos111.jar

(Destination to save):	Any location
(Description):	JavaPOS Device Control. To be imported when creating an application.
(Available from):	JavaPOS-1.11.0-Dist.zip on the web site, http://www.javapos.com/samplecode.html, or http://www.javapos.com/index.html

jpos.xml

(Destination to save): Root directory of project

(Description): A device setup file required to operate each Device Service. The following focuses on the major setup items descried in the file. This file is required for operating each Device Service.

*1. Creation of jpos.xml file

An xml file is provided for each device in the JavaPOS folder. When using the xml files, compile all xml files into one file and name it “jpos.xml”.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE JposEntries PUBLIC "-//JavaPOS//DTD//EN" "jpos/res/jcl.dtd">

    <JposEntry logicalName="DefaultDisplay">
        <creation factoryClass="jpos.toshibatec.linedisplay.loader.JavaPOSServiceFactory"
          serviceClass="jpos.toshibatec.linedisplay.services.LineDisplayService"/>
        <vendor name="TOSHIBA TEC Corporation" url="http://www.toshibatec.co.jp"/>
        <jpos category="LineDisplay" version="1.11"/>
        <product description="TEC LUIST-51 Serial Line Display"
          name="TECLineDisplay" url="http://www.toshibatec.co.jp"/>

        <!--Other non JavaPOS required property (mostly vendor properties and bus specific properties i.e.
RS232 )-->
        <prop name="portName" type="String" value="COM4"/>
        <prop name="countryCode" type="String" value="3"/>
        <prop name="dataBits" type="String" value="8"/>
        <prop name="parity" type="String" value="None"/>
        <prop name="modelName" type="String" value="LIUST-51"/>
        <prop name="flowControl" type="String" value="Xon/Xoff"/>
        <prop name="stopBits" type="String" value="1"/>
        <prop name="deviceBus" type="String" value="RS232"/>
        <prop name="baudRate" type="String" value="9600"/>
  </JposEntry>

  <JposEntries>
        //Descriptions for other devices
  </JposEntry>

</JposEntries>
```

The following describes the major setup items. For details of the setup method, please refer to the Application User Manual of each JavaPOS Device Service.

- <JposEntry logicalName="DefaultDisplay" value="0">
  A description to set a logical device name. Change the shaded area.
- <prop name="portName" type="String" value="COM4"/>
  A description to set COM ports of a device. Change the shaded area.
- <prop name="baudRate" type="String" value="9600"/>
  A description to set baud rate of a device. Change the shaded area.

* 2Defference in descriptions between Windows and Linux
As for portName, "COMX" is used for Windows and "/dev/ttySX" is used for Linux. (X: serial port no.)
Note that COMX starts from 1 while /dev/ttySX starts from 0.

[Windows]
value=”COM1”
value=”COM2”
value=…

[Linux]
value=”dev/ttyS0”
value=”dev/ttyS1”
value=…

log4j.xml
(Destination to save): Root directory of project
(Description): A setup file for a log to be output. To be copied in the directory where the execution file exists. The following focuses on the major setup items descried in the file. Please create your own file.

- <param name="file" value="log/ST-A10.log" />
  A description to set a file name of log to be output.
- <priority value="info" />
  A description to set a log level.

Fatal:	Fatal error	error:	Error
warn:	Warning	info:	Information
debug:	Debug	trace:	Trace

Log4j.dtd
(Destination to save): Root directory of project
(Description): A file to define XML tags. To be copied in the directory where the execution file exists. Please create your own file.

log4j-1.2.12.jar

(Destination to save): Any location

(Description): A library file to output a log. As with JavaPOS DeviceService, this file must be imported in a project.

(Available from): logging-log4j-1.2.12.zip on the web site, http://archive.apache.org/dist/logging/log4j/1.2.12/, or http://logging.apache.org/

commons-logging.jar

(Destination to save): Any location

(Description): A library file to output a log. To be imported when creating an application.

(Available from): commons-logging-1.0.4.zip on the web site, http://archive.apache.org/dist/commons/logging/binaries/, or http://commons.apache.org/logging/

RXTXcomm.jar

(Destination to save): Any location

(Description): A library file to access a Device which uses a COM (component object model). To be imported when creating an application.

(Available from): rxtx-2.1-7-bins-r2.zip on the web site, http://rxtx.qbang.org/pub/rxtx/, or http://users.frii.com/jarvi/rxtx/download.html

swing-layout-1.0.3.jar

(Destination to save): Any location

(Description): A library file to use swing. To be imported when creating an application.

(Available from): swing-layout-1.0.3.jar on the web site, http://java.sun.com/products/archive/jfc/1.0.3/index.html, or http://www.sun.com/

xercesImpl.jar

(Destination to save): Any location

(Description): A library file to convert into text or other XML format. To be imported when creating an application.

(Available from): Xerces-J-bin.2.9.0.zip on the web site, http://apache.adcserver.com.ar/xml/xerces-j/, or http://xerces.apache.org/

xml-apis.jar
(Destination to save): Any location
(Description): A library file to convert into text or other XML format. To be imported when creating an application.
(Available from): Xerces-J-bin.2.9.0.zip on the web site, http://apache.adcserver.com.ar/xml/xerces-j/, or http://xerces.apache.org/

JposEntryEditor.jar
(Destination to save): Any location
(Description): A library file to access an XML file. To be imported when creating an application.
(Available from): jcl2.2.0.zip on the web site, http://Availablefromforge.net/project/showfiles.php?group_id=128804&package_id=141062&release_id=306139, or http://jposloader.Available fromforge.net/downloads/?S=A

JimiProClasses.jar
(Destination to save): Any location
(Description): A library file to access an image file. To be imported when creating an application.
(Available from): jimi1_0.zip on the web site, http://java.sun.com/products/jimi/

rxtxSerial.dll
(Destination to save): Root directory of project
(Description): A library file to access a serial port under Windows.
(Available from): rxtx-2.1-7-bins-r2.zip on the web site, http://rxtx.qbang.org/pub/rxtx/, or http://users.frii.com/jarvi/rxtx/download.html

librxtxSerial.so
(Destination to save): Root directory of project
(Description): A library file used to access a serial port under Linux.
(Available from): rxtx-2.1-7-bins-r2.zip on the web site, http://rxtx.qbang.org/pub/rxtx/, or http://users.frii.com/jarvi/rxtx/download.html

rxtxParallel.dll

(Destination to save): Root directory of project
(Description): A library file used to access a parallel port under Windows.
This is a RXTX parallel library customized by TTEC.
It is based on rxtx-2.1-7(LGPL). “rxtxSerial.dll” file is necessary to use this library.
(Available from): This Kit. Click on “TEC RXTX Parallel Library Source”.
It is bundled with rxtxParallel.zip.

librxtxParallel.so

(Destination to save): Root directory of project
(Description): A library file used to access a parallel port under Linux.
This is a RXTX parallel library customized by TTEC.
It is based on rxtx-2.1-7(LGPL). “librxtxSerial.so” file is necessary to use this library.
(Available from): This Kit. Click on “TEC RXTX Parallel Library Source”.
It is bundled with rxtxParallel.zip.

TECCashDrawerJni.dll

(Destination to save): C:¥Windows¥system32 or C:¥WINNT¥system32
(Description): An application programming interface (API) to be used to access the Windows CashDrawer driver from Java.
(Available from): This Kit. Click on “Driver” → “Cash Drawer Driver” → “Windows”.

libTECCashDrawerJni.so.0.0

(Destination to save): Root directory of project
(Description): An application programming interface (API) to be used to access the Linux.
Make a link file called libTECCashDrawerJni.so and use it.
Ex :]# ln –s libTECCashDrawerJni.so.0.0 libTECCashDrawerJni.so
(Available from): This Kit. Click on “Driver” → “Cash Drawer Driver” → “Linux”.

libTECiButtonJni.so

(Destination to save): Root directory of project
(Description): An application programming interface (API) to be used to access the Linux iButton driver from Java.
(Available from): This Kit. Click on “Driver” → “Linux iButton Driver”

TECUSB.dll
(Destination to save): C:¥Windows¥system32 or C:¥WINNT¥system32
(Description): Library of TECUSB driver for Windows.
(Available from): This Kit. Click on “Driver” → “TECUSB Driver” → “Windows”.

LogMngr.dll
(Destination to save): C:¥Windows¥system32 or C:¥WINNT¥system32
(Description): Library of TECUSB driver for Windows.
(Available from): This Kit. Click on “Driver” → “TECUSB Driver” → “Windows”.

libtecusb.so.0.0
(Destination to save): Root directory of project
(Description): Library of TECUSB driver for Linux.
Make a link file called libtecusb.so and use it.
Ex :]# ln –s libtecusb.so.0.0 libtecusb.so
(Available from): This Kit. Click on “Driver” → “TECUSB Driver” → “Linux”.

TECUSBPM.exe
(Destination to save): C:¥Windows¥system32 or C:¥WINNT¥system32
In case of Vista, root directory of project
(Description): An USB power management process for Windows.
It is nesessary to use TRST-A1x-U on Windows
(Available from): This Kit. Click on”Driver” → “TECUSB Driver” → “Windows”.

TECUSBJNI.dll
(Destination to save): C:¥Windows¥system32 or C:¥WINNT¥system32
(Description): An application programming interface (API) to be used to access the Windows TECUSB driver from Java.
(Available from): This Kit. Click on “Driver” → “TECUSB Driver” → “Windows”.

libTECUSBJNI.so.0.0
(Destination to save): Root directory of project
(Description): An application programming interface (API) to be used to access the Linux.
Make a link file called libTECUSBJNI.so and use it.
Ex :]# ln –s libTECUSBJNI.so.0.0 libTECUSBJNI.so
(Available from): This Kit. Click on”Driver” → “TECUSB Driver” → “Linux”.

libTECPKBFilterJNI.so.0.0
(Destination to save): Root directory of project
(Description): An application programming interface (API) to be used to access the Linux.
Make a link file called libTECPKBFilterJNI.so and use it.
Ex :]# ln –s libTECUSBJNI.so.0.0 libTECUSBJNI.so
(Available from): This Kit. Click on “Driver” →”PKBFilter Driver” →“Linux”.

Example of Creating An Application Using the JavaPOS Device Service

This chapter describes the method to create an application using the JavaPOS Device Service. For this purpose, the demo program enclosed in the Kit is used.

Coding Process

There are the following processes to create an application using the JavaPOS Device Service.

1. Create a device class object.
2. Enable a device.
3. Call to a device-specific method and get a property.
4. Disable a device.

The subsequent sections explain each process.

1. Create a device class object.

```
import javax.swing.DefaultListModel;
import jpos.*;
import jpos.events.*;
import java.util.*;


public class DrawerPanel extends javax.swing.JPanel implements StatusUpdateListener, DirectIOListener{
    private CashDrawer    drawer;


    /** Creates new form DrawerPanel */
    public DrawerPanel() {
        initComponents();
        drawer = new CashDrawer();
    }
```

- Import Jpos.
- Specify a variable for CashDrawer type (for example, drawer).
- By specifying “new”, create a Device Control object.

* To receive events, implement events which are defined by each device.
(For details of implementation, refer to “Supplemental Explanation 2 Receipt of events”.)

2. Enable a device.

```
drawer.open(LOGICALNAME);
drawer.claim(100);
drawer.setDeviceEnabled(true);
```

- Open a device by specifying a device logical name.
- Set a timeout and perform an exclusive process (*).
- Set the DeviceEnabled property to TRUE.

*1 Specify a logical device name for LOGICALNAME.
*2 For devices which do not perform an exclusive processing, no claim handlings are required.
*3 For the claim handling, specify a timeout period in milliseconds in the round brackets as argument.

3. Call to a device-specific method and get a property.

```
Bool status;
drawer.openDrawer();
status = drawer.getOpened();
```

- Define a variable to get a status.
- By calling to the CashDrawer-specific method, a cash drawer opens.
- A status is obtained.

*1 Device-specific methods differ for each device. For details, please refer to the Application User Manual of each Device Service.
*2 Get method and Set method differ for each property. The example below explains the method to get YYY property in XXX object and set ZZZ value to the property.

```
Get:    a=XXX.getYYY();
Set:    XXX.setYYY (ZZZ);
```

4. Disable a device.

```
drawer.setDeviceEnabled(flase);
drawer.release();
drawer.close();
```

- Set the DeviceEnabled property to FALSE.
- Release the exclusive processing.
- Close the device.

*1 For the devices which do not perform an exclusive processing, the release processing is not required.
*2 When the device is closed, an operation speed increases. Please close the device when exiting from the application, as much as possible. To suspend the device, set the DeviceEnabled property to FALSE.

Supplemental Explanation 1. Method to set an exception

```
try
{
        //A process which uses JavaPOS Device Control
        drawer.open(LOGICALNAME);
        drawer.claim(100);
        drawer.setDeviceEnabled(true);

catch(JposException e)
{
        //Describe an exception here.
}
```

Perform an exception for all cases where the JavaPOS Device Control is used, for example, device open, claim, call to device-specific method, property handling.

In detail, describe a code which uses JavaPOS Device Control in braces “{}” and a handling when an exception occurs in braces of catch(JposException){}. JposException is an exception which is thrown when an exception occurs with the JavaPOS Device Control.

Supplemental Explanation 2. Receipt of events

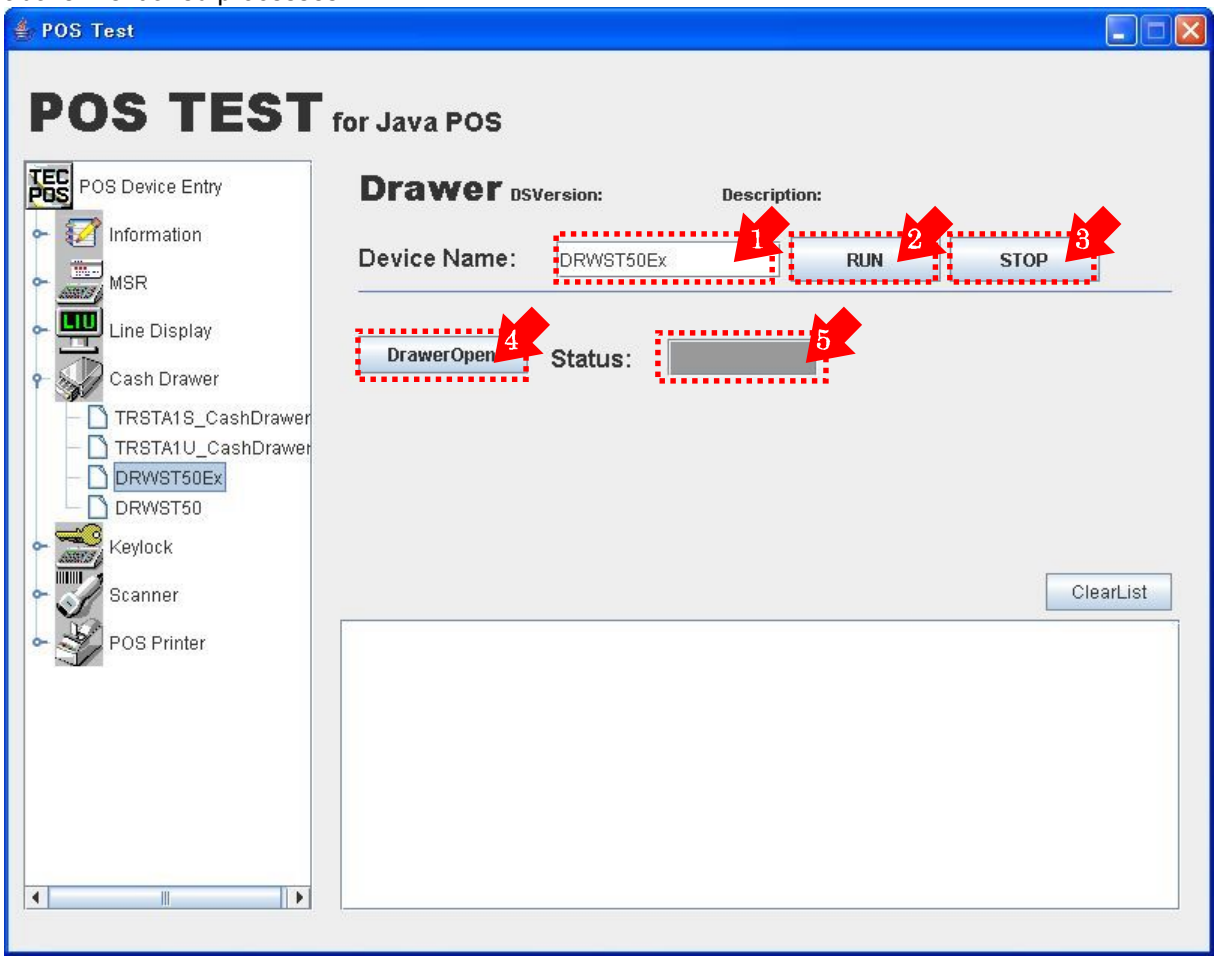
```
public void statusUpdateOccurred(StatusUpdateEvent e){
        //Describe a handling here.
}
public void directIOOccurred(DirectIOEvent e){
        //Describe a handling here.
}
```

- Implement an event interface in a main class. (Please refer to “1. Create a device class object”.)
- Implement functions which should be called when an event occurs.
- Add an appropriate description in the shaded areas.

*1 Event types differ for each device. For details, please refer to the Application User Manual of each device.

Creation of Window

The figure below shows an example of window created when an application is created following the above-mentioned processes.



1. Device Name (Logical Name) text box
A text box which is used to set a logical device name of a device to open the device.

2. Device enable button
Performs a process required to enable the device. Specifically, performs Open and Claim, then set the DeviceEnabled property to TRUE.

3. Device disable button
Performs a process required to disable the device. Specifically, set the DeviceEnabled property to FALSE, then performs Release and Close.

4. Calls to a method to operate a device
Calls to a device-specific device to operate a device. This example calls a drawerOpened property to explain the case of CashDrawer.

5. Property status label
A label to display property information. This example displays the drawerOpened property to explain the case of CashDrawer.