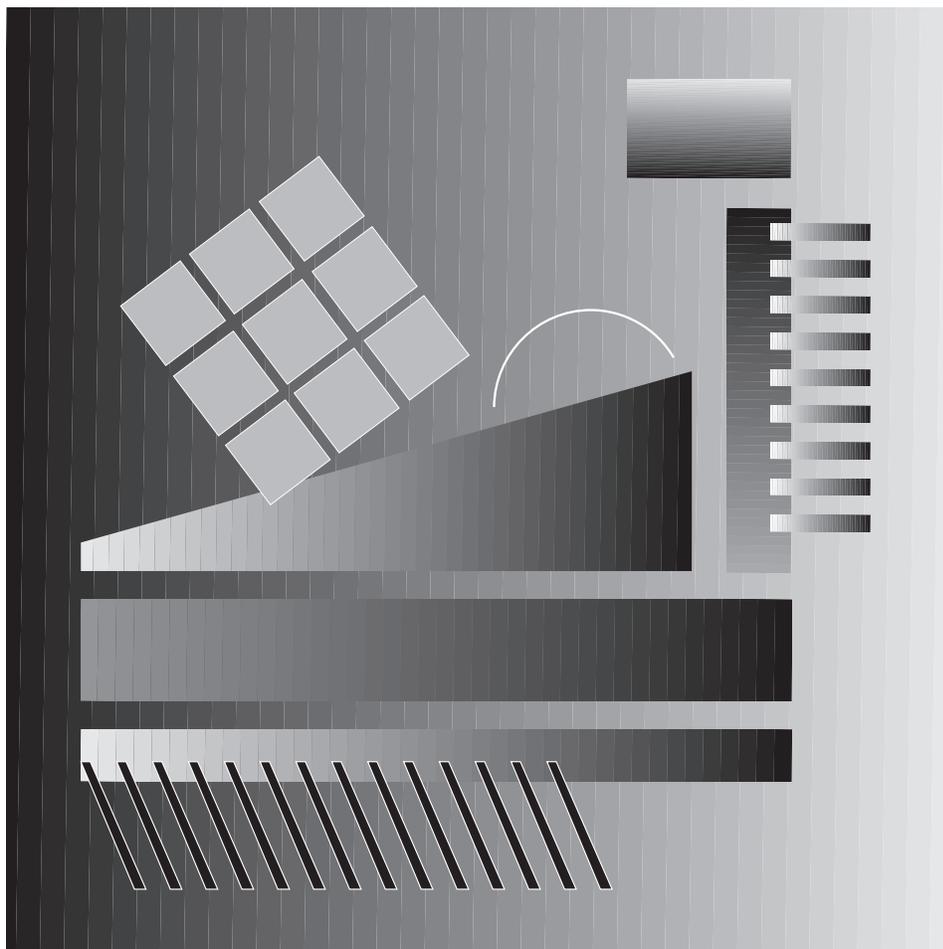


Point of Sale Subsystem

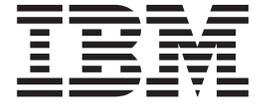


# Programming Reference and User's Guide





Point of Sale Subsystem



# Programming Reference and User's Guide

**Note**

Before using this information and the product it supports, be sure to read the general information under "Appendix K. Notices" on page K-1.

| **Twelfth Edition (September 2001)**

| This edition applies to Version 1.6 of the IBM Point of Sale Subsystem for OS/2, Version 2.3.0 of the IBM Point of Sale Subsystem for Windows, Version 1.0 of the IBM Point of Sale Subsystem for Linux and to all subsequent releases and modifications until otherwise indicated in new editions. This publication is available on the IBM Retail Solutions Electronic Support Web site.

- | 1. Go to [www.ibm.com/solutions/retail/store](http://www.ibm.com/solutions/retail/store).
- | 2. Select **Support** and then **Publications**.

| IBM welcomes your comments. A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

| Department CJMA  
| P.O. Box 12195  
| Research Triangle Park, NC 27709  
| U.S.A.

| Order publications through your IBM representative or the IBM branch office that serves your locality. Publications are not stocked at the address given below. When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1993, 2001. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Tables</b> . . . . .	xix
<b>Preface</b> . . . . .	xxi
Who Should Read this Manual . . . . .	xxi
How to Use this Manual . . . . .	xxi
<b>Related Publications</b> . . . . .	xxiii
IBM Point of Sale Subsystem Related Publications . . . . .	xxiii
OS/2 Publications . . . . .	xxiii
C/C++ for OS/2 Library . . . . .	xxiii
C related Publications . . . . .	xxiii
WorkFrame/2 Publications . . . . .	xxiii
VisualAge® Publications . . . . .	xxiv
Non-IBM related Publications . . . . .	xxiv
Store System Related Publications—Hardware . . . . .	xxiv
Scanners . . . . .	xxiv
Cabling . . . . .	xxiv
4610 SureMark® Point of Sale Printer . . . . .	xxiv
4683/4684 Point of Sale Terminals . . . . .	xxiv
4693/4694/4695 Point of Sale Terminals . . . . .	xxv
SurePOS 700 Series . . . . .	xxv
4820 SurePoint® Solution. . . . .	xxv
7497 Point of Sale Attachment Adapter. . . . .	xxv
Related Software . . . . .	xxv
<b>Summary of Changes</b> . . . . .	xxvii
September 2001 . . . . .	xxvii
June 2001 . . . . .	xxvii
September 2000 . . . . .	xxvii
May 2000 . . . . .	xxvii
March 2000 . . . . .	xxvii
January 2000. . . . .	xxvii

---

## Part 1. User's Guide

<b>Chapter 1. Introduction</b> . . . . .	1-1
Windows and OS/2 Sample Source Code. . . . .	1-1
Linux Sample Source Code . . . . .	1-2
System Requirements . . . . .	1-2
Hardware Environment. . . . .	1-2
Software Environment . . . . .	1-7
Memory and Disk Space Requirements . . . . .	1-8
Memory and Disk Space Requirements for OS/2 . . . . .	1-8
Memory and Disk Space Requirements for Microsoft Windows . . . . .	1-8
Memory and Disk Space Requirements for Linux . . . . .	1-9
<b>Chapter 2. Universal Serial Bus Architecture and IBM Point of Sale Subsystem for Windows.</b> . . . . .	2-1
USB Device Considerations . . . . .	2-1
USB Tree Traversal Order . . . . .	2-1
Device Role Assignment . . . . .	2-2
USB Hot Plug Maintenance . . . . .	2-2
USB Alphanumeric Point of Sale Keyboards . . . . .	2-2

<b>Chapter 3. Customizing the IBM Point of Sale Subsystem</b>	3-1
Configuring Your Applications	3-1
The Resource File	3-1
Using the Resource File	3-3
Configuring the Alphanumeric Point of Sale Keyboard	3-4
<b>Chapter 4. Performing Problem Determination</b>	4-1
Problem Determination on OS/2	4-1
Viewing Point of Sale Error Messages on OS/2.	4-1
Viewing Point of Sale Trace Events on OS/2.	4-1
Problem Determination on the Microsoft Windows Operating System.	4-2
Viewing Events on Microsoft Windows	4-2
Using Built-in Tracing on Microsoft Windows.	4-2
Problem Determination on the Linux Operating System.	4-2
Viewing Events on Linux	4-3
Using Tracing on Linux	4-3

---

## Part 2. Programming Guide

<b>Chapter 5. General Point of Sale Device Programming.</b>	5-1
Your Application and the IBM Point of Sale Subsystem	5-1
Initializing Your Application	5-1
Getting Input Messages	5-2
Determining which Devices are Available	5-3
Opening Your Device	5-4
Controlling Your Device	5-5
Acquiring and Releasing Your Devices	5-6
Defining User-Defined Characters.	5-8
Reading from Your Device	5-9
Writing to Your Device	5-10
Closing Your Device Connections	5-10
Terminating Your Application	5-11
Using Resources in Your Application	5-12
Argument Lists	5-12
Retrieving and Modifying Resources	5-13
Building your Application	5-14
C Language Header Files	5-14
IBM Point of Sale Subsystem Libraries	5-15
Optimizing Application Performance	5-16
Application Priority (OS/2 Only)	5-16
Presentation Manager Considerations (OS/2 Only)	5-16
Microsoft Windows Considerations	5-17
Polling Considerations	5-17
Multi-threaded Application Design	5-18
Improving the Maintainability of Your Application	5-19
<b>Chapter 6. Alarm Programming</b>	6-1
Characteristics of the Alarm	6-1
Functions Your Application Performs.	6-1
Sounding an Alarm	6-1
Silencing an Alarm	6-1
Getting Alarm Status	6-2
Related Information	6-2
Subroutines Used with Alarm	6-2
Alarm PosIOCtl() Control Requests	6-2
Alarm Resources.	6-2

- Alarm Error Codes . . . . . 6-2
- Chapter 7. Cash Drawer Programming . . . . . 7-1**
  - Characteristics of the Cash Drawer . . . . . 7-1
  - Functions Your Application Performs. . . . . 7-1
    - Opening a Cash Drawer Till . . . . . 7-1
    - Getting Cash Drawer Status. . . . . 7-2
    - Setting Cash Drawer Pulse Width. . . . . 7-2
  - Related Information . . . . . 7-2
    - Subroutines Used with Cash Drawer . . . . . 7-2
    - Cash Drawer PosIOctl() Control Requests . . . . . 7-2
    - Cash Drawer Event Messages . . . . . 7-3
    - Cash Drawer Resources . . . . . 7-3
    - Cash Drawer Error Codes . . . . . 7-3
- Chapter 8. Display Programming . . . . . 8-1**
  - Characteristics of the Displays . . . . . 8-1
    - Alphanumeric Display . . . . . 8-1
    - Operator Display . . . . . 8-2
    - Shopper Display . . . . . 8-2
    - Character and Graphics Display . . . . . 8-2
    - 40-Character Liquid Crystal Display . . . . . 8-3
    - 40-Character Vacuum Fluorescent Display II and 2x20 Character VFD
      - Customer Display . . . . . 8-3
      - Two-Sided Vacuum Fluorescent Display II. . . . . 8-3
  - Functions Your Application Performs. . . . . 8-3
    - Code Page Support . . . . . 8-4
    - Writing Characters to the Display . . . . . 8-4
    - Writing Bitmaps to the Display . . . . . 8-5
    - Setting the Guidance Lights . . . . . 8-6
    - Clearing the Display Screen. . . . . 8-6
    - User-Defined Characters . . . . . 8-6
  - Related Information . . . . . 8-7
    - Subroutines Used with Displays . . . . . 8-8
    - Display PosIOctl() Control Requests . . . . . 8-8
    - Display Resources . . . . . 8-8
    - Display Error Codes. . . . . 8-8
- Chapter 9. Keyboard Programming . . . . . 9-1**
  - Characteristics of the Keyboards . . . . . 9-1
    - Keyboard Microcode Updates . . . . . 9-2
      - 50-Key Modifiable Layout Keyboard and 50-Key Modifiable Layout Keyboard and Operator Display . . . . . 9-3
      - Retail Point of Sale Keyboards. . . . . 9-3
      - Modifiable Layout Keyboard with Card Reader . . . . . 9-4
      - ANPOS Keyboard . . . . . 9-5
      - Retail Alphanumeric Point of Sale Keyboard with Card Reader . . . . . 9-6
      - PC Point of Sale Keyboard (ANKPOS Keyboard) . . . . . 9-7
      - Point of Sale Keyboard V. . . . . 9-8
      - PLU Keyboard and Display-III . . . . . 9-9
      - 4685 Point of Sale Keyboard Model K01. . . . . 9-10
      - IBM 4820 SurePoint Solution Keypad . . . . . 9-10
      - Defining Keys. . . . . 9-11
      - Restriction of the Keyboard Device Handler . . . . . 9-12
  - Functions Your Application Performs . . . . . 9-12
    - Reading Keyboard Data. . . . . 9-12

Using the Manager Keyboard Lock . . . . .	9-13
Using the Keyboard Tone . . . . .	9-13
Using the Keyboard Point of Sale Lights . . . . .	9-13
Controlling the Keyboard Click . . . . .	9-14
Controlling the Num Lock Key . . . . .	9-14
Controlling the Scroll Lock key . . . . .	9-14
Controlling the Point of Sale-Unique Keys . . . . .	9-14
Controlling the System Hot Keys . . . . .	9-15
Controlling the Keyboard Typematic Function . . . . .	9-15
Specifying the Numeric Keypad Style . . . . .	9-15
Specifying the Numeric Keypad Location . . . . .	9-15
Related Information . . . . .	9-15
Subroutines Used with Keyboard . . . . .	9-15
Keyboard PosIOctl() Control Requests . . . . .	9-15
Keyboard Event Messages . . . . .	9-16
Keyboard Resources . . . . .	9-16
Keyboard Error Codes . . . . .	9-16
<b>Chapter 10. Magnetic Stripe Reader Programming . . . . .</b>	<b>10-1</b>
Characteristics of the MSR. . . . .	10-1
One-Track Magnetic Stripe Reader. . . . .	10-1
Dual-Track Magnetic Stripe Reader . . . . .	10-2
Three-Track Magnetic Stripe Reader . . . . .	10-2
Two-Head/Two-Sided Magnetic Stripe Reader. . . . .	10-3
Restriction of the MSR Device Handler . . . . .	10-3
Functions Your Application Performs . . . . .	10-3
Unlocking the MSR . . . . .	10-3
Reading MSR Data . . . . .	10-4
Locking the MSR . . . . .	10-5
Related Information . . . . .	10-5
Subroutines Used with MSR . . . . .	10-6
MSR PosIOctl() Control Requests . . . . .	10-6
MSR Event Messages . . . . .	10-6
MSR Error Codes . . . . .	10-6
<b>Chapter 11. Non-Volatile Random Access Memory Programming . . . . .</b>	<b>11-1</b>
Characteristics of the NVRAM Device . . . . .	11-1
Functions Your Application Performs . . . . .	11-2
Available NVRAM Application Address Space . . . . .	11-2
The Cursor Position . . . . .	11-2
Opening NVRAM in Direct Mode or Sequential Mode . . . . .	11-3
Reading Data in Direct Mode or Sequential Mode . . . . .	11-3
Writing Data in Direct Mode or Sequential Mode . . . . .	11-3
Related Information . . . . .	11-4
Subroutines Used with NVRAM . . . . .	11-4
NVRAM PosIOctl() Control Requests . . . . .	11-4
NVRAM Resources . . . . .	11-4
NVRAM Error Codes . . . . .	11-4
<b>Chapter 12. Printer Programming . . . . .</b>	<b>12-1</b>
Characteristics of the Printers. . . . .	12-1
IBM Model 2 Printer . . . . .	12-1
IBM Model 3 and IBM Model 4 Printers . . . . .	12-2
IBM Model 3R and Model 4R Printers. . . . .	12-4
IBM Model 3F Fiscal Printer . . . . .	12-4
IBM Model 4A Printer. . . . .	12-5

IBM 4689-001 and IBM 4689-002 Printer . . . . .	12-6
IBM 4689 Point of Sale Printer Model 301, 3G1, 3M1, and TD5 . . . . .	12-7
IBM 4610 SureMark Point of Sale Printer . . . . .	12-8
Functions Your Application Performs . . . . .	12-11
Code Page Support . . . . .	12-11
Reading Data from the Printer . . . . .	12-12
Writing Data to the Printer . . . . .	12-14
Writing Data in Normal Mode . . . . .	12-14
Writing Data in Logo Mode . . . . .	12-18
Writing Data in Download Message Mode (4610 SureMark Printers Only) . . . . .	12-19
Writing Data in Download Logo Mode . . . . .	12-20
Writing Data in Fiscal Mode (Fiscal Printer Only) . . . . .	12-20
Control Characters . . . . .	12-21
Escape Character Sequences . . . . .	12-22
Printer Input/Output Control Requests (IOCtl) . . . . .	12-33
Printer Resources . . . . .	12-35
Printer Event Messages . . . . .	12-35
Determining the Printer Status . . . . .	12-35
Printer Queues. . . . .	12-35
Document Insert Station . . . . .	12-36
Receipt Paper Cutter . . . . .	12-42
Printing Checks . . . . .	12-43
MICR Reader . . . . .	12-43
Fiscal Printing . . . . .	12-44
User-Defined Characters . . . . .	12-44
IBM 4689-00x in 25 CPL Mode. . . . .	12-44
IBM 4689-00x in 30 CPL Mode. . . . .	12-45
IBM 4689-301, 3G1, 3M1, and TD5 . . . . .	12-46
Performance Considerations. . . . .	12-47
Related Information . . . . .	12-48
Subroutines Used with the Printer. . . . .	12-48
Printer PosIOCtl Control Requests . . . . .	12-48
Printer Event Messages . . . . .	12-48
Printer Resources . . . . .	12-48
Printer Error Codes . . . . .	12-49
<b>Chapter 13. Programmable Power Programming.</b> . . . . .	<b>13-1</b>
Characteristics of the Programmable Power Device. . . . .	13-1
Functions Your Application Performs . . . . .	13-1
Turning Power Off to a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 . . . . .	13-2
Turning Power On and Off to the 4693-2x2. . . . .	13-2
Querying the Time that Power Is to Be Turned On . . . . .	13-3
Related Information . . . . .	13-3
Subroutines Used with Programmable Power Subsystem . . . . .	13-3
Programmable Power Subsystem PosIOCtl() Control Requests . . . . .	13-3
Programmable Power Resources . . . . .	13-4
Programmable Power Device Error Codes . . . . .	13-4
<b>Chapter 14. RS-232C Programming</b> . . . . .	<b>14-1</b>
Characteristics of the RS-232C Port . . . . .	14-2
Functions Your Application Performs . . . . .	14-2
Controlling the RS-232C Port . . . . .	14-2
Reading RS-232C Data . . . . .	14-3
Writing RS-232C Data . . . . .	14-4
Getting RS-232C Port Status . . . . .	14-4
Related Information . . . . .	14-4

Subroutines Used with RS-232C . . . . .	14-4
RS-232C PosIOctl() Control Requests . . . . .	14-5
RS-232C Resources . . . . .	14-5
RS-232C Event Messages . . . . .	14-5
RS-232C Error Codes . . . . .	14-5
<b>Chapter 15. Scale Programming . . . . .</b>	<b>15-1</b>
Characteristics of the Scale Devices . . . . .	15-1
IBM 4687 Point of Sale Sale Scanner Model 2 . . . . .	15-1
IBM 4696 Point of Sale Scanner Scale Model 1 . . . . .	15-1
IBM 4698 Point of Sale Scanner Model 2 . . . . .	15-2
IBM USB Scale Interface . . . . .	15-2
Functions Your Application Performs . . . . .	15-2
Reading Scale Data . . . . .	15-2
Configuring the Scale. . . . .	15-4
Zeroing the Scale . . . . .	15-4
Clearing the Scale Display . . . . .	15-4
Scale Default Values . . . . .	15-4
IBM 4687 Point of Sale Scanner Model 2 . . . . .	15-5
IBM 4696 Point of Sale Scanner Scale Model 1 and IBM 4698 Point of Sale Scanner Model 2 . . . . .	15-5
IBM USB Scale Interface . . . . .	15-5
Related Information . . . . .	15-5
Subroutines Used with Scale . . . . .	15-5
Scale PosIOctl() Control Requests. . . . .	15-6
Scale Resources . . . . .	15-6
Scale Error Codes . . . . .	15-6
<b>Chapter 16. Scanner Programming . . . . .</b>	<b>16-1</b>
Characteristics of the Scanners . . . . .	16-1
Hand-Held Bar Code Readers . . . . .	16-1
IBM 1520 Hand-Held Scanner Model A02 . . . . .	16-2
IBM 4686 Retail Point of Sale Scanner . . . . .	16-2
IBM 4687 Point of Sale Scanner. . . . .	16-2
IBM 4696 Point of Sale Scanner Scale . . . . .	16-2
IBM 4697 Point of Sale Scanner. . . . .	16-3
IBM 4698 Point of Sale Scanner. . . . .	16-3
IBM USB Scanner Interface . . . . .	16-3
Functions Your Application Performs . . . . .	16-4
Unlocking and Locking the Scanner . . . . .	16-4
Reading Scanner Data . . . . .	16-4
Discarding Scanner Data . . . . .	16-8
Configuring the Scanner. . . . .	16-8
Writing Data to the Scanner . . . . .	16-8
Processing Unexpected Scanner Data . . . . .	16-9
Scanner Default Values . . . . .	16-9
Hand-Held Bar Code Reader (All Models). . . . .	16-10
IBM 1520 Hand-Held Scanner Model A02 . . . . .	16-10
IBM 4686 Retail Point of Sale Scanner (All Models) . . . . .	16-10
IBM 4687 Point of Sale Scanner (All Models) . . . . .	16-11
IBM 4696 Point of Sale Scanner Scale Model 1. . . . .	16-11
IBM 4697 Point of Sale Scanner Model 1 . . . . .	16-11
IBM 4698 Point of Sale Scanner (All Models) . . . . .	16-12
IBM USB Scanner Interface . . . . .	16-12
Related Information . . . . .	16-13
Subroutines Used with Scanner . . . . .	16-13

Scanner PosIOctl() Control Requests . . . . .	16-13
Scanner Event Messages. . . . .	16-13
Scanner Resources . . . . .	16-14
Scanner Error Codes . . . . .	16-14
<b>Chapter 17. Touch Screen Programming . . . . .</b>	<b>17-1</b>
Characteristics of the Touch Screen . . . . .	17-1
Video Display. . . . .	17-1
Touch Input . . . . .	17-1
Tone Output . . . . .	17-2
Touch Mouse Emulation . . . . .	17-2
Restriction of the Touch Screen Device Handler . . . . .	17-3
Functions Your Application Performs . . . . .	17-3
Reading Touch Event Data. . . . .	17-3
Using the Tone . . . . .	17-3
Controlling Audible Feedback . . . . .	17-4
Determining which Touch Screen is Available . . . . .	17-4
Controlling the LCD Brightness . . . . .	17-4
Controlling the LCD Contrast . . . . .	17-4
Controlling the Screen Saver Time . . . . .	17-5
Controlling the Backlight On Event Messages . . . . .	17-5
Related Information . . . . .	17-5
Subroutines Used with the Touch Screen . . . . .	17-5
Touch PosIOctl Control Requests . . . . .	17-5
Touch Event Messages . . . . .	17-5
Touch Resources . . . . .	17-5
Touch Error Codes. . . . .	17-6
<b>Chapter 18. Application Programming Interface . . . . .</b>	<b>18-1</b>
PosClose(). . . . .	18-3
PosInitialize(). . . . .	18-5
PosIOctl(). . . . .	18-8
PosOpen(). . . . .	18-10
PosRead(). . . . .	18-16
PosWrite(). . . . .	18-19

---

## Part 3. Programming Reference

<b>Chapter 19. PosIOctl() Requests . . . . .</b>	<b>19-1</b>
POS_ALARM_SILENCE_ALARM . . . . .	19-3
POS_ALARM_SOUND_ALARM . . . . .	19-4
POS_DSP_CLEAR_SCREEN. . . . .	19-5
POS_DSP_DEFINE_CHARACTERS . . . . .	19-6
POS_KBD_DISABLE_HOT_KEYS . . . . .	19-7
POS_KBD_DISABLE_NUM_LOCK. . . . .	19-8
POS_KBD_DISABLE_SCROLL_LOCK . . . . .	19-9
POS_KBD_ENABLE_HOT_KEYS. . . . .	19-10
POS_KBD_ENABLE_NUM_LOCK . . . . .	19-11
POS_KBD_ENABLE_SCROLL_LOCK . . . . .	19-12
POS_KBD_SET_NUM_LOCK_OFF . . . . .	19-13
POS_KBD_SET_NUM_LOCK_ON . . . . .	19-14
POS_KBD_SET_SCROLL_LOCK_OFF . . . . .	19-15
POS_KBD_SET_SCROLL_LOCK_ON . . . . .	19-16
POS_KBD_SET_TPEMATIC_OFF . . . . .	19-17
POS_KBD_SET_TPEMATIC_ON . . . . .	19-18
POS_KBD_SILENCE_TONE . . . . .	19-19

POS_KBD_SOUND_TONE . . . . .	19-20
POS_POWER_OFF . . . . .	19-21
POS_POWER_ON . . . . .	19-22
POS_POWER_SET_ALARM . . . . .	19-23
POS_PRN_DEFINE_CHARACTERS . . . . .	19-24
POS_PRN_DISABLE_DI_PRINTING . . . . .	19-25
POS_PRN_DISABLE_FISCAL_PRINTING . . . . .	19-26
POS_PRN_DISCARD_DATA . . . . .	19-27
POS_PRN_ENABLE_DI_PRINTING . . . . .	19-28
POS_PRN_ENABLE_FISCAL_PRINTING . . . . .	19-29
POS_PRN_RESET_PRINTER . . . . .	19-30
POS_PRN_HOLD_PRINTING . . . . .	19-31
POS_PRN_RELEASE_PRINTING . . . . .	19-32
POS_PRN_RESUME_PRINTING . . . . .	19-33
POS_PRN_RETRY_PRINTING . . . . .	19-34
POS_PRN_SILENCE_TONE . . . . .	19-35
POS_PRN_SOUND_TONE . . . . .	19-36
POS_RS232_SEND_BREAK . . . . .	19-37
POS_SCALE_CLEAR_SCREEN . . . . .	19-38
POS_SCALE_ZERO_SCALE . . . . .	19-39
POS_SCAN_DISCARD_DATA . . . . .	19-40
POS_SYS_ACQUIRE_DEVICE . . . . .	19-41
POS_SYS_GET_VALUES . . . . .	19-42
POS_SYS_LOCK_DEVICE . . . . .	19-43
POS_SYS_RELEASE_DEVICE . . . . .	19-44
POS_SYS_SET_VALUES . . . . .	19-45
POS_SYS_UNLOCK_DEVICE . . . . .	19-47
POS_TILL_OPEN_TILL . . . . .	19-48
POS_TOUCH_SILENCE_TONE . . . . .	19-49
POS_TOUCH_SOUND_TONE . . . . .	19-50

<b>Chapter 20. Event Messages . . . . .</b>	<b>20-1</b>
POSM_KBD_STATUS_CHANGE . . . . .	20-3
POSM_KBD_WM_CHAR . . . . .	20-4
POSM_MSR_DATA_AVAIL . . . . .	20-6
POSM_PRN_CHASE_COMPLETE . . . . .	20-7
POSM_PRN_DATA_AVAIL . . . . .	20-8
POSM_PRN_FISCAL_ERROR . . . . .	20-9
POSM_PRN_FISCAL_STATUS . . . . .	20-10
POSM_PRN_PRINTER_ERROR . . . . .	20-11
POSM_PRN_STATUS_CHANGE . . . . .	20-13
POSM_RS232_BREAK_DETECTED . . . . .	20-15
POSM_RS232_DATA_AVAIL . . . . .	20-16
POSM_RS232_XMIT_ABORT . . . . .	20-17
POSM_RS232_XMIT_COMPLETE . . . . .	20-18
POSM_SCAN_DATA_AVAIL . . . . .	20-19
POSM_SYS_DEVICE_OFFLINE . . . . .	20-20
POSM_SYS_DEVICE_ONLINE . . . . .	20-22
POSM_SYS_DEVICE_RELEASED . . . . .	20-24
POSM_TILL_CLOSED . . . . .	20-25
POSM_TILL_OPENED . . . . .	20-26
POSM_TOUCH_DATA . . . . .	20-27
WM_CHAR . . . . .	20-28

<b>Chapter 21. Resource Sets . . . . .</b>	<b>21-1</b>
Resource Access Codes . . . . .	21-5

PosSystem Resource Set . . . . .	21-5
PosNqueueHandle . . . . .	21-6
PosNreadTimeout . . . . .	21-6
PosNvitalProductData . . . . .	21-7
PosDevice Resource Set . . . . .	21-7
PosNdeviceNumber . . . . .	21-8
PosNportNumber . . . . .	21-14
PosNqueueHandle . . . . .	21-14
PosNslotNumber . . . . .	21-15
PosAlarm Resource Set . . . . .	21-16
PosNalarmStatus . . . . .	21-16
PosDisplay Resource Set . . . . .	21-16
PosNcharSize . . . . .	21-17
PosNdisplayCodePage . . . . .	21-17
PosNdisplayCursor . . . . .	21-19
PosNdisplayMode . . . . .	21-19
PosNdisplayLightsOn . . . . .	21-20
PosNpixelX . . . . .	21-21
PosNpixelY . . . . .	21-21
PosDrawer Resource Set . . . . .	21-21
PosNpulseWidth . . . . .	21-21
PosNtillStatus . . . . .	21-22
PosKeyboard Resource Set . . . . .	21-22
PosNdoubleKey01 - PosNdoubleKey60 . . . . .	21-23
PosNfatFingerTimeOut . . . . .	21-24
PosNkeyboardClick . . . . .	21-25
PosNkeyLock . . . . .	21-25
PosNkeyboardLightsOn . . . . .	21-26
PosNnumpadLocation . . . . .	21-26
PosNnumpadStyle . . . . .	21-27
PosNnumpadZero . . . . .	21-27
PosNtoneDuration . . . . .	21-28
PosNtoneFreq . . . . .	21-28
PosNtoneVolume . . . . .	21-28
PosNtypematicDelay . . . . .	21-29
PosNtypematicFreq . . . . .	21-29
PosMsr Resource Set . . . . .	21-30
PosNvram Resource Set . . . . .	21-30
PosNnvramCursor . . . . .	21-30
PosNnvramMode . . . . .	21-31
PosNnvramSize . . . . .	21-31
PosPower Resource Set . . . . .	21-31
PosNpowerAlarm . . . . .	21-31
PosPrinter Resource Set . . . . .	21-32
PosNcodePage . . . . .	21-33
PosNCRWidth . . . . .	21-35
PosNdiOrientation . . . . .	21-35
PosNDIWidth . . . . .	21-35
PosNfeedDirection . . . . .	21-36
PosNfiscalCountry . . . . .	21-37
PosNfiscalNotify . . . . .	21-37
PosNfiscalPLDStatus . . . . .	21-37
PosNfiscalVersion . . . . .	21-38
PosNheadParkedPosition . . . . .	21-38
PosNinterleaved . . . . .	21-39
PosNleftMarginCR . . . . .	21-39

PosNlineFeedCR . . . . .	21-40
PosNlineFeedDI . . . . .	21-40
PosNlineFeedSJ . . . . .	21-41
PosNprintAlignment . . . . .	21-42
PosNprintColorMode . . . . .	21-42
PosNprintCRCharSetx . . . . .	21-42
PosNprintDICharSetx . . . . .	21-43
PosNprintFeatures . . . . .	21-43
PosNprintMode . . . . .	21-44
PosNprintQualityMode . . . . .	21-44
PosNprintStation . . . . .	21-45
PosNprintStatus . . . . .	21-46
PosNprintStatus2 . . . . .	21-47
PosNprintTabStops . . . . .	21-47
PosNprintToneDuration . . . . .	21-48
PosNprintToneFrequency . . . . .	21-48
PosNprintToneNote . . . . .	21-48
PosNprintToneOctave . . . . .	21-49
PosNprintToneVolume . . . . .	21-49
PosNprintUpsideDown . . . . .	21-50
PosNrawPrintStatus . . . . .	21-50
PosNresumeString . . . . .	21-50
PosNretryString . . . . .	21-52
PosRs232c Resource Set . . . . .	21-53
PosNbaudRate . . . . .	21-53
PosNdataBits . . . . .	21-54
PosNlineMode . . . . .	21-54
PosNparity . . . . .	21-55
PosNrs232Status . . . . .	21-55
PosNstopBits . . . . .	21-55
PosNtimeoutChar . . . . .	21-56
PosScale Resource Set . . . . .	21-56
PosNdisplayRequired . . . . .	21-57
PosNnumWeightDigits . . . . .	21-57
PosNoperMode . . . . .	21-58
PosNvibrationFilter . . . . .	21-58
PosNweightMode . . . . .	21-59
PosNzeroIndState . . . . .	21-59
PosNzeroRetState . . . . .	21-60
PosScanner Resource Set . . . . .	21-60
Scanner Model Identifiers . . . . .	21-62
PosNbarCodes1 . . . . .	21-62
PosNbarCodes2 . . . . .	21-64
PosNbarCodes3 . . . . .	21-65
PosNbarCodes4 . . . . .	21-66
PosNbarCodeProgramming . . . . .	21-67
PosNbeepFreq . . . . .	21-67
PosNbeepLength . . . . .	21-67
PosNbeepState . . . . .	21-68
PosNbeepVolume . . . . .	21-68
PosNblinkLength . . . . .	21-69
PosNblockReadMode . . . . .	21-69
PosNblock1Type . . . . .	21-70
PosNblock2Type . . . . .	21-71
PosNblock3Type . . . . .	21-71
PosNbVolSwitchState . . . . .	21-72

PosNcheckModulo . . . . .	21-72
PosNcode128ScansPerRead . . . . .	21-73
PosNcode39ScansPerRead . . . . .	21-73
PosNdecodeAlgorithm . . . . .	21-73
PosNdReadTimeout . . . . .	21-74
PosNdTouchMode . . . . .	21-74
PosNeAN13ScansPerRead . . . . .	21-75
PosNeAN8ScansPerRead . . . . .	21-75
PosNiTFLength1 . . . . .	21-76
PosNiTFLength2 . . . . .	21-77
PosNiTFLengthType . . . . .	21-77
PosNiTFScansPerRead . . . . .	21-78
PosNjANTwoLabelDecode . . . . .	21-78
PosNlabelsQueued . . . . .	21-78
PosNlaserSwitchState . . . . .	21-79
PosNlaserTimeout . . . . .	21-79
PosNmotorTimeout . . . . .	21-80
PosNqueueAllLabels . . . . .	21-80
PosNscansPerRead . . . . .	21-81
PosNstoreScansPerRead . . . . .	21-81
PosNsupplementals . . . . .	21-82
PosNtransmitCheckDigit . . . . .	21-82
PosNtwoLabelFlagPair1 . . . . .	21-83
PosNtwoLabelFlagPair2 . . . . .	21-84
PosNtwoLabelFlagPair3 . . . . .	21-85
PosNtwoLabelFlagPair4 . . . . .	21-86
PosNuPCAScansPerRead . . . . .	21-87
PosNuPCDScansPerRead . . . . .	21-87
PosNuPCEscansPerRead . . . . .	21-88
PosNuPCExpansion . . . . .	21-88
PosNverifyPriceChk . . . . .	21-89
PosTouch Resource Set . . . . .	21-89
PosNtouchBackLightOnEvent . . . . .	21-90
PosNtouchBrightness . . . . .	21-90
PosNtouchClickVolume . . . . .	21-91
PosNtouchContrast . . . . .	21-91
PosNtouchEntryClick . . . . .	21-91
PosNtouchExitClick . . . . .	21-92
PosNtouchMaxX . . . . .	21-92
PosNtouchMaxY . . . . .	21-92
PosNtouchMode . . . . .	21-92
PosNtouchScreenSaverTime . . . . .	21-93
PosNtouchToneDuration . . . . .	21-93
PosNtouchToneFreq . . . . .	21-93
PosNtouchToneVolume . . . . .	21-94

---

## Part 4. Appendixes

<b>Appendix A. Data Types and Macros . . . . .</b>	<b>A-1</b>
Data Types . . . . .	A-1
Macros . . . . .	A-1
<b>Appendix B. Trace Codes . . . . .</b>	<b>B-1</b>
<b>Appendix C. Error Messages . . . . .</b>	<b>C-1</b>

<b>Appendix D. Error Codes</b>	D-1
Error Messages, Numeric Order	D-4
301 POSERR_SYS_OS_ERROR	D-4
302 POSERR_SYS_NOT_INITIALIZED	D-4
303 POSERR_SYS_INVALID_DESCRIPTOR	D-4
304 POSERR_SYS_ALREADY_INITIALIZED	D-4
305 POSERR_SYS_MEMORY_ALLOCATION	D-4
306 POSERR_SYS_HW_ERROR	D-5
307 POSERR_SYS_INVALID_DEVICE	D-5
308 POSERR_SYS_INVALID_QUEUE	D-5
309 POSERR_SYS_TOO_MANY_DEVICES	D-5
311 POSERR_SYS_FUNCTION_NOT_SUPPORTED	D-5
312 POSERR_SYS_BUFFER_TOO_SMALL	D-5
313 POSERR_SYS_ACQUIRED_BY_OTHER	D-5
314 POSERR_SYS_ALREADY_ACQUIRED	D-5
315 POSERR_SYS_NOT_ACQUIRED	D-6
316 POSERR_SYS_INVALID_REQUEST	D-6
317 POSERR_SYS_DEVICE_OFFLINE	D-6
318 POSERR_SYS_INVALID_LENGTH	D-6
319 POSERR_SYS_INVALID_CLASS_DEVICE_COMBO	D-6
320 POSERR_SYS_DATA_DISCARDED	D-6
321 POSERR_SYS_INTERNAL_ERROR	D-7
325 POSERR_SYS_INVALID_NARGS	D-7
326 POSERR_SYS_INVALID_SLOT	D-7
327 POSERR_SYS_UNSUPPORTED_ADAPTER	D-7
328 POSERR_SYS_INVALID_PORT	D-7
329 POSERR_SYS_TIMEOUT	D-7
330 POSERR_SYS_INVALID_NAME	D-7
331 POSERR_SYS_INVALID_CLASS	D-7
332 POSERR_SYS_INTERRUPTED	D-7
334 POSERR_SYS_INVALID_ADDRESS	D-8
335 POSERR_SYS_LOCKED_NO_DATA_READ	D-8
336 POSERR_SYS_INVALID_FILE	D-8
337 POSERR_SYS_SERVICE_NOT_AVAILABLE	D-8
4101 POSERR_NVRAM_NOT_ENOUGH_ROOM	D-8
4102 POSERR_NVRAM_INVALID_CURSOR	D-8
4103 POSERR_NVRAM_EOF	D-8
4104 POSERR_NVRAM_INVALID_MODE	D-9
4105 POSERR_NVRAM_INVALID_LENGTH_RECORD	D-9
4106 POSERR_NVRAM_INVALID_DATA_CRC	D-9
4401 POSERR_DSP_INVALID_CURSOR	D-9
4402 POSERR_DSP_INVALID_MODE	D-9
4403 POSERR_DSP_INVALID_SIZE	D-9
4404 POSERR_DSP_UNSUPPORTED_BITMAP	D-9
4405 POSERR_DSP_BAD_BITMAP	D-9
4406 POSERR_DSP_INVALID_CODE_PAGE	D-10
4701 POSERR_KBD_INVALID_FREQUENCY	D-10
4702 POSERR_KBD_INVALID_DURATION	D-10
4703 POSERR_KBD_INVALID_VOLUME	D-10
4705 POSERR_KBD_INVALID_DOUBLE_KEY	D-10
4706 POSERR_KBD_INVALID_FAT_FINGER_TIMEOUT	D-10
4708 POSERR_KBD_INVALID_KEYBOARD_CLICK	D-10
4709 POSERR_KBD_INVALID_NUMPAD_STYLE	D-10
4710 POSERR_KBD_INVALID_NUMPAD_ZERO	D-10
4711 POSERR_KBD_INVALID_TYPEMATIC_DELAY	D-11
4712 POSERR_KBD_INVALID_TYPEMATIC_FREQ	D-11

4713	POSERR_KBD_INVALID_NUMPAD_LOCATION . . . . .	D-11
4901	POSERR_PRN_INVALID_DI_WIDTH. . . . .	D-11
4902	POSERR_PRN_INVALID_INTERLEAVED_VALUE . . . . .	D-11
4903	POSERR_PRN_INVALID_HEAD_PARKED_POSITION . . . . .	D-11
4904	POSERR_PRN_INVALID_STATION . . . . .	D-11
4905	POSERR_PRN_INVALID_MODE . . . . .	D-12
4906	POSERR_PRN_INVALID_CR_LF_DISTANCE . . . . .	D-12
4907	POSERR_PRN_INVALID_SJ_LF_DISTANCE. . . . .	D-12
4908	POSERR_PRN_INVALID_DI_LF_DISTANCE. . . . .	D-12
4909	POSERR_PRN_INVALID_FEED_DIRECTION . . . . .	D-12
4910	POSERR_PRN_INVALID_FISCAL_NOTIFY . . . . .	D-12
4911	POSERR_PRN_INVALID_DI_ORIENTATION . . . . .	D-12
4912	POSERR_PRN_INVALID_LEFT_MARGIN_CR . . . . .	D-12
4913	POSERR_PRN_INVALID_PRINT_ALIGNMENT . . . . .	D-12
4914	POSERR_PRN_INCORRECT_DATA_FORMAT . . . . .	D-12
4915	POSERR_PRN_LOGO_EXISTS . . . . .	D-13
4916	POSERR_PRN_UDC_CHARACTER_EXISTS . . . . .	D-13
4918	POSERR_PRN_INVALID_QUALITY_MODE . . . . .	D-13
4919	POSERR_PRN_INVALID_UPSIDE_DOWN_MODE . . . . .	D-13
4920	POSERR_PRN_INVALID_TABSTOPS_FORMAT . . . . .	D-13
4921	POSERR_PRN_INVALID_COLOR_MODE. . . . .	D-13
4922	POSERR_PRN_INVALID_TONE_VOLUME . . . . .	D-13
4923	POSERR_PRN_INVALID_TONE_DURATION . . . . .	D-14
4924	POSERR_PRN_INVALID_TONE_NOTE . . . . .	D-14
4925	POSERR_PRN_INVALID_TONE_OCTAVE . . . . .	D-14
4926	POSERR_PRN_INVALID_TONE_FREQUENCY. . . . .	D-14
4927	POSERR_PRN_INVALID_CODE_PAGE . . . . .	D-14
5401	POSERR_CDR_INVALID_PULSE_WIDTH. . . . .	D-14
5702	POSERR_SCAN_INVALID_BAR_CODES_1 . . . . .	D-14
5703	POSERR_SCAN_INVALID_BAR_CODES_2 . . . . .	D-14
5704	POSERR_SCAN_INVALID_BEEP_FREQ . . . . .	D-14
5705	POSERR_SCAN_INVALID_BEEP_LENGTH . . . . .	D-15
5706	POSERR_SCAN_INVALID_BEEP_STATE . . . . .	D-15
5707	POSERR_SCAN_INVALID_BEEP_VOLUME . . . . .	D-15
5708	POSERR_SCAN_INVALID_BLINK_LENGTH. . . . .	D-15
5709	POSERR_SCAN_INVALID_BLOCK_READ_MODE . . . . .	D-15
5710	POSERR_SCAN_INVALID_BLOCK_1_TYPE. . . . .	D-15
5711	POSERR_SCAN_INVALID_BLOCK_2_TYPE. . . . .	D-15
5712	POSERR_SCAN_INVALID_BLOCK_3_TYPE. . . . .	D-15
5713	POSERR_SCAN_INVALID_CHECK_MODULO . . . . .	D-15
5714	POSERR_SCAN_INVALID_D_READ_TIMEOUT . . . . .	D-15
5715	POSERR_SCAN_INVALID_D_TOUCH_MODE . . . . .	D-16
5716	POSERR_SCAN_INVALID_ITF_LENGTH_1 . . . . .	D-16
5717	POSERR_SCAN_INVALID_ITF_LENGTH_2 . . . . .	D-16
5718	POSERR_SCAN_INVALID_LASER_TIMEOUT . . . . .	D-16
5719	POSERR_SCAN_INVALID_MOTOR_TIMEOUT. . . . .	D-16
5720	POSERR_SCAN_INVALID_LASER_SWITCH_STATE . . . . .	D-16
5721	POSERR_SCAN_INVALID_SCANS_PER_READ . . . . .	D-16
5722	POSERR_SCAN_LABEL_TOO_SHORT . . . . .	D-16
5725	POSERR_SCAN_INVALID_BVOL_SWITCH_STATE . . . . .	D-16
5726	POSERR_SCAN_INVALID_DECODE_ALGORITHM . . . . .	D-17
5727	POSERR_SCAN_INVALID_EAN13_SCANS_PER_READ . . . . .	D-17
5728	POSERR_SCAN_INVALID_EAN8_SCANS_PER_READ. . . . .	D-17
5729	POSERR_SCAN_INVALID_STORE_SCANS_PER_READ . . . . .	D-17
5730	POSERR_SCAN_INVALID_UPCA_SCANS_PER_READ . . . . .	D-17
5731	POSERR_SCAN_INVALID_UPCD_SCANS_PER_READ . . . . .	D-17

5732	POSERR_SCAN_INVALID_UPCE_SCANS_PER_READ	D-17
5733	POSERR_SCAN_INVALID_UPC_EXPANSION	D-17
5734	POSERR_SCAN_INVALID_VERIFY_PRICE_CHK	D-17
5735	POSERR_SCAN_INVALID_QUEUE_ALL_INDICATOR	D-17
5736	POSERR_SCAN_CONFIGURATION_ERROR	D-18
5737	POSERR_SCAN_2_LABEL_FLAG_CONFIG_ERROR	D-18
5738	POSERR_SCAN_INVALID_2_LABEL_DECODE_STATE	D-18
5739	POSERR_SCAN_INVALID_FLAG_PAIR_COMBINATION	D-18
5740	POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_1	D-18
5741	POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_2	D-18
5742	POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_3	D-18
5743	POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_4	D-18
5744	POSERR_SCAN_INVALID_CODE39_SCANS_PER_READ	D-19
5745	POSERR_SCAN_INVALID_INT2OF5_SCANS_PER_READ	D-19
5746	POSERR_SCAN_INVALID_CODE128_SCANS_PER_READ	D-19
5747	POSERR_SCAN_INVALID_BAR_CODES_3	D-19
5748	POSERR_SCAN_INVALID_BAR_CODES_4	D-19
5750	POSERR_SCAN_INVALID_ITF_LENGTH_TYPE	D-19
5751	POSERR_SCAN_INVALID_SUPPLEMENTALS	D-19
5752	POSERR_SCAN_INVALID_BARCODE_PROG_STATE	D-19
5753	POSERR_SCAN_INVALID_XMIT_CHECK_DIGIT	D-19
5901	POSERR_RS232_INVALID_BAUD_RATE	D-19
5902	POSERR_RS232_INVALID_STOP_BITS	D-20
5903	POSERR_RS232_INVALID_PARITY	D-20
5904	POSERR_RS232_INVALID_DATA_BITS	D-20
5905	POSERR_RS232_INVALID_TIMEOUT_CHAR	D-20
5907	POSERR_RS232_PREV_NOT_COMPLETE	D-20
6201	POSERR_SCALE_INVALID_OPERATIONS_MODE	D-20
6202	POSERR_SCALE_INVALID_REMOTE_DISPLAY_STATE	D-20
6203	POSERR_SCALE_INVALID_VIBRATION_FILTER	D-20
6204	POSERR_SCALE_INVALID_WEIGHT_MODE	D-20
6205	POSERR_SCALE_INVALID_ZERO_INDICATOR_STATE	D-20
6206	POSERR_SCALE_INVALID_ZERO_RETURN_STATE	D-21
6207	POSERR_SCALE_ZERO_SCALE_FAILED	D-21
6208	POSERR_SCALE_INVALID_CLEAR_SCREEN_REQUEST	D-21
6209	POSERR_SCALE_CONFIGURATION_ERROR	D-21
6211	POSERR_SCALE_INVALID_NUM_WEIGHT_DIGITS	D-21
6401	POSERR_POWER_INVALID_DAY	D-21
6402	POSERR_POWER_INVALID_HOUR	D-21
6403	POSERR_POWER_INVALID_MINUTES	D-21
6701	POSERR_TOUCH_INVALID_BACKLIGHT_ON	D-21
6702	POSERR_TOUCH_INVALID_CLICK_VOLUME	D-22
6703	POSERR_TOUCH_INVALID_CONTRAST	D-22
6704	POSERR_TOUCH_INVALID_ENTRY_CLICK	D-22
6705	POSERR_TOUCH_INVALID_EXIT_CLICK	D-22
6706	POSERR_TOUCH_INVALID_MODE	D-22
6707	POSERR_TOUCH_INVALID_SCREEN_SAVER_TIME	D-22
6708	POSERR_TOUCH_INVALID_TONE_DURATION	D-22
6709	POSERR_TOUCH_INVALID_TONE_FREQUENCY	D-22
6710	POSERR_TOUCH_INVALID_TONE_VOLUME	D-22
6711	POSERR_TOUCH_INVALID_BRIGHTNESS	D-23

<b>Appendix E. IBM Model 4A Font Download</b>	E-1
Manually Downloading Characters	E-1
Font File Format	E-2
Keywords	E-2

Character Definition Record . . . . .	E-2
<b>Appendix F. Downloading fonts.</b> . . . . .	F-1
USB Character/Graphics Display . . . . .	F-1
Downloading Fonts to the Display . . . . .	F-1
IBM 4689 SurePOS Receipt Journal Printer . . . . .	F-2
Converting Printer Font Files . . . . .	F-2
Downloading Fonts to the Printer . . . . .	F-2
4610 SureMark Point of Sale Printer Model TI5, TM7, and TF7 . . . . .	F-3
Converting Printer Font Files . . . . .	F-3
Downloading Fonts to the Printer . . . . .	F-4
Other 4610 SureMark Point of Sale Printer Models . . . . .	F-4
Converting Printer Font Files . . . . .	F-4
Downloading Fonts to the Printer . . . . .	F-5
<b>Appendix G. IBM 4610 Printer Firmware Update</b> . . . . .	G-1
Manually Updating Firmware . . . . .	G-1
<b>Appendix H. Firmware Update Utility for USB Devices.</b> . . . . .	H-1
<b>Appendix I. Using the IBM SureBase UPS Utility</b> . . . . .	I-1
<b>Appendix J. 4820 SurePoint Solution Touch Screen Calibration</b> . . . . .	J-1
USB Calibration . . . . .	J-1
RS-485 Touch Screen Calibration Utility . . . . .	J-1
Properties and methods of the tool . . . . .	J-1
Using the Touch Screen Calibration Utility . . . . .	J-2
<b>Appendix K. Notices</b> . . . . .	K-1
Trademarks. . . . .	K-2
<b>Glossary</b> . . . . .	L-1
<b>Index</b> . . . . .	X-1

created on October 2, 2001

## Tables

1-1.	Memory Requirements for OS/2 . . . . .	1-8
1-2.	Disk Space Requirements for OS/2 . . . . .	1-8
1-3.	Memory Requirements for Microsoft Windows. . . . .	1-8
1-4.	Disk Space Requirements for Microsoft Windows . . . . .	1-8
1-5.	Memory Requirements for Microsoft Windows. . . . .	1-9
1-6.	Disk Space Requirements . . . . .	1-9
8-1.	Bitmap Data Format . . . . .	8-6
11-1.	Maximum Block Size for NVRAM Read and NVRAM Write Operations . . . . .	11-1
12-1.	Maximum Number of Characters for the IBM Model 2 Printer. . . . .	12-2
12-2.	Maximum Number of Characters for IBM Model 3/3F/4/4A Printers . . . . .	12-3
12-3.	Maximum Number of Addressable Print Positions Per Line for the IBM Model 4A Printer . . . . .	12-5
12-4.	Maximum Number of Addressable Print Positions per Character for the IBM Model 4A Printer . . . . .	12-5
12-5.	Maximum Number of Characters for IBM 4689-001/002 Printers . . . . .	12-6
12-6.	Maximum Number of Characters for IBM 4689-3x1 and TD5 Printers. . . . .	12-7
12-7.	Maximum Number of Characters for IBM 4610 SureMark Point of Sale Printer Printers . . . . .	12-9
12-8.	Text Print Attributes for the IBM 4610 SureMark Printer models. . . . .	12-10
12-9.	Maximum Number of Characters for PosWrite() . . . . .	12-14
12-10.	Printer Control Characters and Escape Character Sequences . . . . .	12-16
12-11.	Text Attribute enable and disable codes for 4610 models and 4689 Model 3x1 and TD5 printers . . . . .	12-32
16-1.	Label Types Returned by the Scanner Device Handler . . . . .	16-5
17-1.	DoubleClickHeight and DoubleClickWidth Adjustment for Windows . . . . .	17-3
17-2.	Determining which Touch Screen is available . . . . .	17-4
21-1.	PosSystem Resources. . . . .	21-5
21-2.	PosDevice Resources . . . . .	21-8
21-3.	PosAlarm Resources . . . . .	21-16
21-4.	PosDisplay Resources . . . . .	21-17
21-5.	PosDrawer Resources . . . . .	21-21
21-6.	PosKeyboard Resources . . . . .	21-22
21-7.	PosNvram Resources. . . . .	21-30
21-8.	PosPower Resources. . . . .	21-31
21-9.	PosPrinter Resources. . . . .	21-32
21-10.	PosRs232c Resources . . . . .	21-53
21-11.	Maximum PosNtimeoutChar Value . . . . .	21-56
21-12.	PosScale Resources . . . . .	21-57
21-13.	PosScanner Resources . . . . .	21-61
21-14.	Valid Values for PosNbarCodes1 Resource (SIO and USB Scanners) . . . . .	21-62
21-15.	Valid Values for PosNbarCodes1 Resource (USB Scanners Only) . . . . .	21-63
21-16.	Valid Values for PosNbarCodes2 Resource (SIO and USB Scanners) . . . . .	21-64
21-17.	Valid Values for PosNbarCodes2 Resource (USB Scanners Only) . . . . .	21-64
21-18.	Valid Values for PosNbarCodes3 and PosNbarCodes4 Resources (SIO and USB Scanners) . . . . .	21-65
21-19.	Valid Values for PosNbarCodes3 and PosNbarCodes4 Resources (USB Scanners Only) . . . . .	21-66
21-20.	Valid Values for PosNblock1Type Resource. . . . .	21-70
21-21.	Valid Values for PosNblock2Type Resource. . . . .	21-71
21-22.	Valid Values for PosNblock3Type Resource. . . . .	21-71
21-23.	Valid Values for PosNsupplementals Resource . . . . .	21-82
21-24.	Valid Values for PosNtransmitCheckDigit Resource . . . . .	21-82
21-25.	PosTouch Resources . . . . .	21-90
D-1.	Error Code Reference . . . . .	D-4

created on October 2, 2001

---

## Preface

This manual provides reference information for programming devices used by the IBM Point of Sale Subsystem.

---

### Who Should Read this Manual

This manual is intended for use by point-of-sale programmers who choose to use the IBM point-of-sale hardware, and either the IBM OS/2® operating system, the Microsoft® Windows® operating system, or the Linux operating system.

Depending on the operating system that is being used, this manual assumes that the reader is familiar with the following terms:

- For IBM Point of Sale Subsystem for OS/2:
  - Information Presentation Facility (IPF)
  - OS/2
  - Point of Sale environment
  - Presentation Manager® programming
- For IBM Point of Sale Subsystem for Windows:
  - Microsoft Windows Help
  - Microsoft Windows
  - Point of Sale environment
  - Microsoft Windows programming
- For IBM Point of Sale Subsystem for Linux:
  - Point of Sale environment
  - Linux programming

This manual also assumes that the user is proficient with the C programming language.

---

### How to Use this Manual

This manual is divided into four parts:

- **Part 1, User's Guide**, provides general information about configuring and troubleshooting the IBM Point of Sale Subsystem.
- **Part 2, Programming Guide**, explains what devices are supported by the device handlers, and how to program these devices.
- **Part 3, Programming Reference** contains reference information about PosIOctl requests, event messages, and resource sets.
- **Part 4, Appendixes** contains information about error codes, and POSS utilities.

Unless specifically stated, the information in this manual applies the Point of Sale Subsystems as follows: the IBM Point of Sale Subsystem for OS/2, the IBM Point of Sale Subsystem for Windows, and the IBM Point of Sale Subsystem for Linux.

When information applies specifically to only some of the Point of Sale Subsystem operating environments, the text indicates which operating systems are supported.



---

## Related Publications

The section lists related publications. For information about ordering these publications, contact your IBM authorized dealer or marketing representative.

Between major revisions of this manual we may make minor technical updates. The latest softcopy version of the *IBM Point of Sale Subsystem Installation, Keyboards, and Code Pages* manual is available on the IBM Retail Store Solutions Web site. To access this publication:

1. Go to [www.ibm.com/solutions/retail/store](http://www.ibm.com/solutions/retail/store)
2. Select **Support**
3. Click the appropriate hardware or peripheral driver

---

## IBM Point of Sale Subsystem Related Publications

- *IBM Point of Sale Subsystem Programming Reference and User's Guide*, SC30-3560 (this publication)
- *IBM Point of Sale Subsystem Installation, Keyboards, and Code Pages*, GC30-3623
- *FFST/2 Administration Guide*, S96F-8593

---

## OS/2 Publications

*OS/2 2.1 Documentation Only*, S61G-0905  
*OS/2 Tool Kit Technical Library*, SB0F-1206  
*Object Oriented Interface Design Common User Access Guideline*, SC34-4399  
*Control Program Programming Reference*, S10G-6263  
*Presentation Manager Programming Reference Volume 1*, S10G-6264  
*Presentation Manager Programming Reference Volume 2*, S10G-6265  
*Presentation Manager Driver Reference*, S10G-6267  
*OS/2 WARP, V3 Technical Library*, SB0F-8511

---

## C/C++ for OS/2 Library

*C/C++ Tools: Programming Guide*, S61G-1181  
*C/C++ Tools: Debugger Introduction*, S61G-1184  
*C/C++ Tools: Execution Trace Analyzer Introduction*, S61G-1398  
*C/C++ Tools: Browser Introduction*, S61G-1397  
*C/C++ Tools: Class Libraries Reference Summary*, S61G-1186  
*C/C++ Tools: C Library Reference*, S61G-1183  
*C/C++ Tools: C Language Reference*, S61G-1399  
*C/C++ Tools: C++ Language Reference*, S61G-1185  
*C/C++ Tools: Standard Class Library Reference*, S61G-1180  
*C/C++ Tools: User Interface Class Library Reference*, S61G-1179  
*C/C++ Tools: Collection Class Library Reference*, S61G-1178  
*C/C++ Tools: Reference Summary*, S61G-1441

---

## C related Publications

*Portability Guide for IBM C*, SC09-1405

---

## WorkFrame/2 Publications

*C++: WorkFrame/2: Introduction*, S61G-1428

---

## VisualAge® Publications

*VisualAge C++ for OS/2, V3 Standard Manuals, S30H-1679*  
*VisualAge C++ for OS/2, V3 Extended Reference, S30H-1680*  
*VisualAge C++ for Windows V3.5 Standard Manuals, S33H-4981*  
*VisualAge C++ for Windows V3.5 Reference Manuals, S33H-4982*

---

## Non-IBM related Publications

*The ANSI Specifications for Magnetic-Stripe Encoding for Credit Cards, ANSI X4.16-1983*  
*The ANSI Specifications for Credit Cards, ANSI X4.13*  
*The Korean Industry Code for Information Exchange, KSC-5601*  
*KANJI Code Table, N:GC18-2040-3*

For publications relating to non-IBM software, contact the software vendor.

---

## Store System Related Publications—Hardware

### Scanners

*1520 Hand-Held Scanner User's Guide, GA27-3685*  
*4686 Retail Point of Sale Scanner: Physical Planning, Installation, and Operation Guide, SA27-3854*  
*4686 Retail Point of Sale Scanner: Maintenance Manual, SY27-0319*  
*4687 Point of Sale Scanner Model 1: Physical Planning, Installation, and Operation Guide, SA27-3855*  
*4687 Point of Sale Scanner Model 1: Maintenance Manual, SY27-0317*  
*4687 Point of Sale Scanner Model 2: Physical Planning Guide, SA27-3882*  
*4687 Point of Sale Scanner Model 2: Operator's Guide, SA27-3884*  
*4687 Point of Sale Scanner Model 2: Maintenance Manual, SY27-0324*  
*4696 Point of Sale Scanner: Maintenance Manual, SY27-0333*  
*4696 Point of Sale Scanner: Physical Planning, Installation, and Operation Guide, GA27-3965*  
*4697 Point of Sale Scanner Model 001: Maintenance Manual, SY27-0338*  
*4697 Point of Sale Scanner Model 001: Physical Planning, Installation, and Operation Guide, GA27-3990*  
*4698 Point of Sale Scanner Scale Model 001 & 002: Physical Planning, Installation, and Operation Guide, GA27-4055*  
*4698 Point of Sale Scanner Scale Model 001 & 002: Maintenance Manual, SY27-0344*

### Cabling

*A Building Planning Guide for Communication Wiring, G320-8059*  
*Cabling System Planning and Installation Guide, GA27-3361*  
*Cabling System Catalog, G570-2040*  
*Using the IBM Cabling System with Communication Products, GA27-3620*

### 4610 SureMark® Point of Sale Printer

*4610 SureMark Point of Sale Printer: User's Guide, GA27-4151*

### 4683/4684 Point of Sale Terminals

*4683 Point of Sale Terminal: Installation Guide, SA27-3783*  
*4684 Point of Sale Terminal: Installation Guide, SA27-3837*  
*4684 Point of Sale Terminal: Introduction and Planning Guide, SA27-3835*  
*4683/4684 Point of Sale Terminal: Operations Guide, SA27-3704*

*4680 Store System and IBM 4683/4684 Point of Sale Terminal: Problem Determination Guide, SY27-0330*

*4684 Point of Sale Terminal: Maintenance Summary Card, SX27-3885*

*4680 Store System: Terminal Test Procedures Reference Summary, GX27-3779*

*IBM 4683/4684 Point of Sale Terminal: Maintenance Manual, SY27-0295*

## **4693/4694/4695 Point of Sale Terminals**

*4683/4684/4693/4694 Point of Sale Terminal: Parts Catalog, S131-0097*

*4693 Point of Sale Terminal: Setup Instructions*

*4693 Point of Sale Terminal: Quick Reference Card*

*4693 Point of Sale Terminal: Configuration and Operation Guide, SA27-3978*

*4693/4694/4695 Point of Sale Terminal: Maintenance and Test Summary, SX27-3919*

*Store Systems: Technical Reference, SY27-0336*

*4693/4694/4695 Point of Sale Terminal: Hardware Service Manual, SY27-0337*

*Store Systems: Hardware Service Manual for Point of Sale Input/Output Devices, SY27-0339*

*4694 Point of Sale Terminal: User's Guide, SA27-4005*

*4694 Point of Sale Terminal: Hardware Service Manual, SY27-0364*

*4695 Point of Sale Terminal: Installation and Operation Guide, GA27-4031*

*4695 Point of Sale Terminal: Hardware Service Manual, SY27-0361*

*Store Systems: Installation and Operation Guide for Point of Sale Input/Output Devices, GA27-4028*

*Store Systems: Point of Sale Terminals—Supplement for Installation, Operation, and Service, GA27-4035*

## **SurePOS 700 Series**

*SurePOS 700 Series: Installation and Operation Guide, GA27-4223*

*SurePOS 700 Series: Hardware Service Manual, GY27-0363*

*SurePOS 700 Series: System Reference, SA27-4224*

*SurePOS 700 Series Options and I/O Devices Service Guide, SY27-0392*

## **4820 SurePoint® Solution**

*4820 SurePoint Solution: Installation and Service Guide, GY27-4231*

*4820 SurePoint Solution: System Reference, SA27-4249*

## **7497 Point of Sale Attachment Adapter**

*Point of Sale Terminal Attachment Kit: Physical Planning, Installation, and Service Manual, GA27-4034*

---

## **Related Software**

Utility software, LAN drivers, video drivers, and diagnostic software are available. See the latest list on the IBM Retail Store Solutions Web site at:

<http://www.ibm.com/solutions/retail/store/>

Select **Support**, and then click the appropriate hardware or peripheral driver.



---

## Summary of Changes

---

### September 2001

Updates for Linux Support.

---

### June 2001

Updates to include the following information:

- Fiscal printer support
  - Additional user-defined character resources
  - Updates to AIPFNT46.exe
  - Trace utilities information
  - Calibration tool. See “Appendix J. 4820 SurePoint Solution Touch Screen Calibration” on page J-1 for additional information.
- 

### September 2000

- Added updates for Single Station SureMark printers (4610 –TM6/TF6/TM7/TF7)
- 

### May 2000

- Addition of Double-byte information for printers
  - Additional information for configuring printers
- 

### March 2000

- Additional information added for PosNcodePage and PDF417 Barcode.
  - Clarifications to existing text for USB scanner interface
- 

### January 2000

- Support for Universal Serial Bus (USB) architecture
- Support for the IBM 4610 SureMark™ Point of Sale Printer Model TI5

created on October 2, 2001

---

## Part 1. User's Guide

	<b>Chapter 1. Introduction</b>	1-1
	Windows and OS/2 Sample Source Code.	1-1
	Linux Sample Source Code	1-2
	System Requirements	1-2
	Hardware Environment.	1-2
	Point of Sale Terminals (for Windows and OS/2):	1-2
	Point of Sale Terminals (Linux):	1-2
	Point of Sale Adapter Cards (Windows and OS/2):	1-3
	RS-485 Point of Sale Devices (Windows and OS/2):	1-3
	RS-485 Point of Sale Devices (Linux):	1-6
	USB Point of Sale Devices (Windows 98 and Windows 2000 only):	1-7
	Software Environment	1-7
	Memory and Disk Space Requirements	1-8
	Memory and Disk Space Requirements for OS/2	1-8
	Memory and Disk Space Requirements for Microsoft Windows	1-8
	Memory and Disk Space Requirements for Linux	1-9
	<b>Chapter 2. Universal Serial Bus Architecture and IBM Point of Sale Subsystem for Windows.</b>	2-1
	USB Device Considerations	2-1
	USB Tree Traversal Order	2-1
	Device Role Assignment	2-2
	USB Hot Plug Maintenance	2-2
	USB Alphanumeric Point of Sale Keyboards	2-2
	<b>Chapter 3. Customizing the IBM Point of Sale Subsystem</b>	3-1
	Configuring Your Applications	3-1
	The Resource File	3-1
	Using the Resource File	3-3
	Configuring the Alphanumeric Point of Sale Keyboard	3-4
	<b>Chapter 4. Performing Problem Determination</b>	4-1
	Problem Determination on OS/2	4-1
	Viewing Point of Sale Error Messages on OS/2.	4-1
	Viewing Point of Sale Trace Events on OS/2.	4-1
	Problem Determination on the Microsoft Windows Operating System.	4-2
	Viewing Events on Microsoft Windows	4-2
	Using Built-in Tracing on Microsoft Windows.	4-2
	Problem Determination on the Linux Operating System.	4-2
	Viewing Events on Linux	4-3
	Using Tracing on Linux	4-3

created on October 2, 2001

---

## Chapter 1. Introduction

The purpose of the IBM Point of Sale Subsystem product is to manage input/output (I/O) devices that are unique to point-of-sale (POS) usage. The IBM Point of Sale Subsystem operates by actively controlling the I/O devices and providing an application programming interface (API) to applications registered with the subsystem. The IBM Point of Sale Subsystem provides the following:

- Dynamic determination of the devices attached to the system. The IBM Point of Sale Subsystem notifies all registered applications which devices are attached via their event message queue. This means that there is no configuration file that has to be set up during installation time.
- An automated installation facility. See the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book for details about the installation process.
- Dynamic loading of device handlers. Only the point-of-sale device drivers are loaded at initial program load (IPL) time. The rest of the IBM Point of Sale Subsystem is loaded into the system as needed.
- A “C” application programming interface for opening, closing, reading, writing, and controlling point-of-sale I/O devices. The application programming interface is designed to give the application flexibility. See Chapter 5. General Point of Sale Device Programming, Chapter 18. Application Programming Interface, and Part 3. Programming Reference for detailed information about the IBM Point of Sale Subsystem application programming interface.
- Problem determination using error logs and tracing. See Chapter 4. Performing Problem Determination for details about the problem determination process.
- Sample code that shows how to use the IBM Point of Sale Subsystem API.

---

## Windows and OS/2 Sample Source Code

On Windows and OS/2 systems, this source code resides in the SAMPLE subdirectory of the directory in which the product was installed; the default location is C:\POS\SAMPLE. The Windows and OS/2 sample source code includes the following:

### **AIPTSTR**

Calls the IBM Point of Sale Subsystem API by way of prompted input. The results of the application programming interface calls are displayed on the screen. Also includes a sample resource file.

### **ANPOSKEY**

Traces keys received from the ANPOS keyboard.

### **CHECKOUT**

Shows how to program using the IBM Point of Sale Subsystem API.

**DEMO** Illustrates the use of the IBM Point of Sale Subsystem API. It does this without prompting the user for parameters and only prompts for various actions (such as to sound a tone).

### **VBSAMPLE**

Illustrates the use of the IBM Point of Sale Subsystem for WindowsAPI from Visual Basic®. Sample code is included for customer display and NVRAM devices. These sample programs are only installed if Visual Basic support is installed.

---

## Linux Sample Source Code

On Linux systems, the default location for the sample source code is `/usr/doc/pos/sample`. The Linux sample source code includes the following:

### CHECKOUT

Shows how to use the IBM Point of Sale Subsystem API.

---

## System Requirements

This section describes the hardware, software, disk space, and memory that are required for the IBM Point of Sale Subsystem.

## Hardware Environment

The IBM Point of Sale Subsystem supports the following hardware:

### Point of Sale Terminals (for Windows and OS/2):

- 4674 Point of Sale Terminal Models (Japan only):
  - 4674-001, 4674-010, 4674-011
- 4674 Point of Sale Terminal Models (Japan only):
  - 4674-DS1
- 4683 Point of Sale Terminal Models (**OS/2 only**):
  - 4683-002, 4683-A02, 4683-421
- 4684 Point of Sale Terminal Models (**OS/2 only**):
  - 4684-300
- 4693 Point of Sale Terminal Models:
  - 4693-321, 4693-331, 4693-3S1, 4693-3W1, 4693-421, 4693-431, 4693-4S1, 4693-541, 4693-551, 4693-5S1, 4693-741, 4693-7S1, 4693-202, 4693-212, 4693-2S2
- 4694 Point of Sale Terminal Models:
  - 4694-001, 4694-004, 4694-024, 4694-041, 4694-044, 4694-S01, 4694-S04, 4694-S41, 4694-S44, 4694-104, 4694-SS4, 4694-114, 4694-124, 4694-144, 4694-1S4, 4694-154, 4694-205, 4694-206, 4694-207, 4694-244, 4694-245, 4694-254, 4694-2S4
- 4694 Point of Sale Terminal Models (**Windows only**):
  - 4694-206, 4694-207, 4694-246, 4694-247, 4694-2S6, 4694-2L6, 4694-307, 4694-347
- 4694 Point of Sale Terminal Models (Preloaded Microsoft **Windows NT**<sup>®</sup> **Models**):
  - 4694-LNT, 4694-SNT
- 4695 Point of Sale Integrated Touch Terminal Models:
  - 4695-201, 4695-211, 4695-321, 4695-322, 4695-324, 4695-331, 4695-342, 4695-344, 4695-N43
- SurePOS<sup>™</sup> 700 Series Models (**Windows only**):
  - SureBase<sup>™</sup> Model 001, SurePOS 730 Model 102, SurePOS 730 Model 142, SurePOS 750 Model 202, SurePOS 750 Model 20E, SurePOS 750 Model 242, SurePOS 750 Model 24E

### Point of Sale Terminals (Linux):

- 4694 Point of Sale Terminal Models:

- 4694-104, 4694-106, 4694-146, 4694-205, 4694-206, 4694-207, 4694-245, 4694-246, 4694-247, 4694-307, 4694-347

### **Point of Sale Adapter Cards (Windows and OS/2):**

- 4695 Point of Sale Adapter (Feature Code 3941)
- 4695 Point of Sale Adapter/A (Feature Code 3941)
- 4695 Point of Sale Adapter II (Feature Code 3930)
- 7497 Point of Sale Attachment Adapter Model 001:
  - ISA Bus Adapter (P/N 73G2529, Feature Code 3529)
  - Micro Channel Adapter (P/N 83G0986, Feature Code 0986)

### **RS-485 Point of Sale Devices (Windows and OS/2):**

#### ***Cash Drawers:***

- Cash Drawer, No Till (Feature Code 3360)
- Adjustable Till (Feature Code 1092)
- Fixed Till (Feature Code 3879)
- Cash Drawer, Removable Till (Feature Code 3361)
- Flip-Top Cash Drawer (Feature Code 3362)
- Cash Drawer I (P/N 6238669)
- Cash Drawer IV (P/N 09F3519)
- Cash Drawer V (Feature Code 3370)
- Compact Cash Drawer with Vertical Till (Feature Code 3368)
- Compact Cash Drawer with Horizontal Till (Feature Code 3378)

#### ***Displays:***

- Shopper Display (Feature Code 3339)
- Operator Display (Feature Code 3340)
- 40 Character Alphanumeric Display (Feature Code 3343)
- 40 Character Vacuum Fluorescent Display (Feature Code 3343)
- Character/Graphics Display (Feature Code 3400)
  - Japan - Tall (Feature Code 3402)
  - Japan - Short (Feature Code 3403)
  - Korean - Tall (Feature Code 3405)
  - Korean - Short (Feature Code 3406)
- 40-Character Vacuum Fluorescent Display II (Feature Code 3501)
- 40-Character Vacuum Fluorescent Display II - Japan (Feature Code 3506)
- Two-Sided Vacuum Fluorescent Display II (Feature Code 3502)
- Two-Sided Vacuum Fluorescent Display II - Japan (Feature Code 3507)
- 40-Character Liquid Crystal Display (Feature Code 3503)
- 2x20 Character Vacuum Fluorescent Display Customer Display (Feature Code 2826)

#### ***Keyboards:***

- 50-Key Modifiable Keyboard (Feature Code 3320)
- 50-Key Modifiable Layout Keyboard/Operator Display (Feature Code 6300)
- Alphanumeric Point of Sale Keyboard (Feature Code 3324)
- Retail Point of Sale Keyboard with Card Reader:
  - Brazil/Portuguese (Feature Code 3200)
  - Danish (Feature Code 3211)
  - Canada/French (Feature Code 3201)
  - French (Feature Code 3203)
  - German (Feature Code 3204)
  - Italian (Feature Code 3205)
  - Norwegian (Feature Code 3212)
  - Spanish (Feature Code 3206)
  - Swedish/Finnish (Feature Code 3213)
  - U.K. English (Feature Code 3202)
  - U.S. English (Feature Code 3324)

- Retail Point of Sale Keyboard (Feature Code 3315)
- Retail Point of Sale Keyboard with Card Reader (Feature Code 3320)
- Modifiable Layout Keyboard with Card Reader (Feature Code 3323)
- 4820 SurePoint™ Solution 32-Key Keypad (Feature Code 5140)
- PC Point of Sale Keyboard
  - Japan (Feature Code 3207)
  - Korea (Feature Code 3208)
- Point of Sale Keyboard V
  - Japan (Feature Code 3220)
  - Korea (Feature Code 3221)
- Point of Sale Keyboard VI - Korea (Feature Code 3209)
- PLU Keyboard/Display III
  - Japan (Feature Code 3230)
  - Korea (Feature Code 3232)
- Retail Point of Sale Keyboard with Card Reader and Display (Feature Code 6300)
- 4685 Point of Sale Keyboard Model K01 (4685-K01)

**Magnetic Stripe Readers:**

- One-Track Magnetic Stripe Reader - ISO Track 2 (Feature Code 4010)
- Dual-Track Magnetic Stripe Reader - ISO Tracks 1 and 2 (Feature Code 4192)
- Dual-Track Magnetic Stripe Reader - ISO Tracks 2 and 3 (Feature Code 4193)
- Low-Profile Dual-Track Magnetic Stripe Reader - ISO Tracks 1 and 2 (Feature Code 6310)
- Low-Profile Dual-Track Magnetic Stripe Reader - ISO Tracks 2 and 3 (Feature Code 6320)
- Three-Track Magnetic Stripe Reader (Feature Code 2905)
- Two-Sided Magnetic Stripe Reader (Feature Code 2906)
- SurePoint™ Magnetic Stripe Reader (Feature Code 3951)
- SurePoint JUCS Magnetic Stripe Reader (Feature Code 3953)
- 4820 SurePoint Solution Magnetic Stripe Reader (**Windows Only**)

**Non-volatile Random Access Memory:**

- 4693 Point of Sale Terminal Models (except Microsoft Windows 3.1):  
4693-202, 4693-212, 4693-2S2
- 4693 Point of Sale Terminal Models:  
4693-321, 4693-331, 4693-3S1, 4693-3W1, 4693-421, 4693-431,  
4693-4S1, 4693-541, 4693-551, 4693-5S1, 4693-741, 4693-7S1
- 4694 Point of Sale Terminal Models:  
4694-001, 4694-004, 4694-024, 4694-041, 4694-044, 4694-S01,  
4694-S04, 4694-S41, 4694-S44, 4694-104, 4694-SS4, 4694-114,  
4694-124, 4694-144, 4694-1S4, 4694-154, 4694-205, 4694-244,  
4694-245, 4694-254, 4694-2S4
- 4694 Point of Sale Terminal Models (**Windows Only**):  
4694-206, 4694-207, 4694-246, 4694-247, 4694-2S6, 4694-2L6,  
4694-307, 4694-347, 4694-LNT, 4694-SNT
- 4695 Point of Sale Integrated Touch Terminal Models:  
4695-201, 4695-211, 4695-321, 4695-322, 4695-324, 4695-331,  
4695-342, 4695-344, 4695-N43
- Point of Sale Adapter Cards:  
4695 Point of Sale Adapter (Feature Code 3941)  
4695 Point of Sale Adapter/A (Feature Code 3941)  
4695 Point of Sale Adapter II (Feature Code 3930)  
7497 Point of Sale Attachment Adapter Model 001

**Printers:**

- Model 2 Printer (Feature Code 6400)
- Model 3 Printer (Feature Code 4700)
- Model 3F Fiscal Printer
- IBM Model 3R Printer (Feature Code 4701)
- IBM Model 4 Printer (Feature Code 4800)
- IBM Model 4A Printer (Feature Code 4805)
- IBM Model 4R Printer (Feature Code 4801)
- 4610 SureMark Point of Sale Printer Models:  
4610-TI1, 4610-TI2, 4610-TI3, 4610-TI4
- 4689 Point of Sale Printer Models:  
4689-001 Japan (Feature Code 4802)  
4689-002 Korea (Feature Code 4803)  
4689-301
- **Windows only:**
  - 4610 SureMark Point of Sale Printer Models:  
4610-TI5, 4610-TF6, 4610-TF7, 4610-TM6, 4610-TM7, 4610-TN3,  
4610-TN4
  - 4610 SureMark Point of Sale Fiscal Printers
  - 4689 Point of Sale Printer Models:  
4689-3G1, 4689-3M1, 4689-TD5

**Programmable Power:**

- 4693 Point of Sale Terminals:  
4693-202, 4693-212, 4693-2S2
- 4683 Point of Sale Terminals (**OS/2 Only**):  
4683-002, 4683-421, 4684-300
- 4693 Point of Sale Terminals (**OS/2 Only**):  
4693-321, 4693-331, 4693-3S1, 4693-3W1, 4693-421, 4693-431,  
4693-4S1, 4693-541, 4693-551, 4693-5S1, 4693-741, 4693-7S1

**Scales (except Microsoft Windows 3.1):**

- 4687 Point of Sale Scanner Model 002
- 4696 Point of Sale Scanner Scale Model 001
- 4698 Point of Sale Scanner Model 002

**Scanners:**

- Hand-Held Bar Code Reader Model 2 (Feature Code 4501)
- IBM 1520 Hand-Held Scanner Model A02
- 4685 Hand-Held Bar Code Reader Models:  
4685-001 (Feature Code 4502)  
4685-L01 (Handy Scanner III)  
4685-L0A
- 4685 Point of Sale Scanner Model L0F
- 4685 SurePOS Scanner Models:  
4685-S01, 4685-L0C, 4685-L0H, 4685-101
- 4687 Point of Sale Scanner Models:  
4687-001, 4687-002
- 4696 Point of Sale Scanner Scale Model 001
- 4697 Point of Sale Scanner Model 001
- 4698 Point of Sale Scanner Models:  
4698-001, 4698-002
- **OS/2 Only:**
  - 4686 Point of Sale Scanner Models:  
4686-001, 4686-002, 4686-003, 4686-004

**Touch:**

- 4695 Point Of Sale Distributed Touch Terminal Models:  
4695-002, 4695-012, 4695-022, 4695-032, 4695-042

- 4695 Point Of Sale Integrated Touch Terminal Models:  
4695-201, 4695-211, 4695-321, 4695-322, 4695-324, 4695-331,  
4695-342, 4695-344, 4695-N43
- SurePoint Monochrome Touch Screen (Feature Code 3950)
- SurePoint Color Touch Screen (Feature Code 3960)
- **Windows only:**
  - 4820 SurePoint Solution Color Touch Screen Models:  
4820-46T, 4820-46R

**Miscellaneous:**

- Alarm (second cash drawer)
- Feature E Card Devices (RS-232)

**RS-485 Point of Sale Devices (Linux):**

**Cash Drawers:**

- Cash Drawer, No Till (Feature Code 3360)  
Adjustable Till (Feature Code 1092)  
Fixed Till (Feature Code 3879)
- Cash Drawer, Removable Till (Feature Code 3361)
- Flip-Top Cash Drawer (Feature Code 3362)
- Cash Drawer I (P/N 6238669)
- Cash Drawer IV (P/N 09F3519)
- Cash Drawer V (Feature Code 3370)
- Compact Cash Drawer with Vertical Till (Feature Code 3368)
- Compact Cash Drawer with Horizontal Till (Feature Code 3378)

**Displays:**

- 40-Character Vacuum Fluorescent Display II (Feature Code 3501)
- 40-Character Vacuum Fluorescent Display II - Japan (Feature Code 3506)
- Two-Sided Vacuum Fluorescent Display II (Feature Code 3502)
- Two-Sided Vacuum Fluorescent Display II - Japan (Feature Code 3507)
- 40-Character Liquid Crystal Display (Feature Code 3503)

**Keyboards:**

- Retail Alphanumeric Point of Sale Keyboard with Card Reader
- Retail Point of Sale Keyboard (Feature Code 3315)
- Retail Point of Sale Keyboard with Card Reader and Display (Feature Code 3320)

**Non-volatile Random Access Memory:**

- 4694 Point of Sale Terminal Models:
  - 4694-104, 4694-106, 4694-146, 4694-205, 4694-206, 4694-207, 4694-245,  
4694-246, 4694-247, 4694-307, 4694-347

**Magnetic Stripe Readers:**

- Three-Track Magnetic Stripe Reader (Feature Code 2905)

**Printers:**

- Model 3F Fiscal Printer
- 4610 SureMark Point of Sale Printer Models:  
4610-TI1, 4610-TI2, 4610-TI3, 4610-TI4, 4610-TF6, 4610-TM6, 4610-TF7,  
4610-TM7
- 4610 SureMark Point of Sale Fiscal Printers

**Scales**

- 4698 Point of Sale Scanner Models:
  - 4698-002

**Scanners:**

- Hand-Held Bar Code Reader Model 2 (Feature Code 4501)
- IBM 1520 Hand-Held Scanner Model A02
- 4698 Point of Sale Scanner Models:  
4698-001, 4698-002

**USB Point of Sale Devices (Windows 98 and Windows 2000 only):**

**Cash Drawers:**

- Full-size Cash Drawer (Fixed Till)
- Full-size Cash Drawer (Adjustable Till)
- Compact Cash Drawer (Vertical Till)
- Compact Cash Drawer (Horizontal Till)

**Displays:**

- USB 40 Character Vacuum Fluorescent Display
- USB Two-Sided Vacuum Fluorescent Display
- USB 40 Character Liquid Crystal Display
- USB Character/Graphics Display

**Keyboards:**

- USB 50-Key Keyboard
- USB 50-Key Keyboard with Magnetic Stripe Reader
- USB 50-Key Keyboard with Magnetic Stripe Reader and Liquid Crystal Display
- USB Alphanumeric Point of Sale Keyboard
- USB 133-Key Keyboard with Magnetic Stripe Reader
- USB 4820 SurePoint Solution 32-Key Keypad

**Magnetic Stripe Readers:**

- USB 4820 SurePoint Solution Magnetic Stripe Reader

**Non-volatile Random Access Memory:**

- 4800 Point of Sale Terminals

**Printers:**

- USB 4610 SureMark Point of Sale Printer Models:  
4610-TI3, 4610-TI4, 4610-TI5, 4610-TM6, 4610-TM7
- USB 4610 SureMark Point of Sale Fiscal Printers

## Software Environment

The IBM Point of Sale Subsystem requires the following:

- OS/2 Version 2.1 or later
- Microsoft Windows Version 3.1 with Win32s Version 1.25A
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows NT Version 3.51 or later
- Microsoft Windows 2000
- Red Hat Linux Version 7.1 (2.4 Kernel)

Depending on the operating system that you use, you will need the following software to develop applications for the IBM Point of Sale Subsystem:

**OS/2**

IBM VisualAge® C/C++ for OS/2  
Borland C/C++ for OS/2

**Windows**

Microsoft Visual C++® Version 1.5 (16-bit applications)

Microsoft Visual C++ Version 2.0 or later (32-bit applications)  
 Borland C/C++ for Windows Version 4.5 or later  
 IBM VisualAge C/C++ for Windows Version 3.5 or later

## Linux

GNU's Compiler Collection (GCC)

---

## Memory and Disk Space Requirements

This section describes the memory and disk space required to run the IBM Point of Sale Subsystem. The memory requirements are approximate. The disk space requirements are listed for each selectable component of the IBM Point of Sale Subsystem.

### Memory and Disk Space Requirements for OS/2

*Table 1-1. Memory Requirements for OS/2*

Function	Recommended Memory (MB)
Device Handlers	1.2
Alphanumeric Point of Sale Keyboard	0.3
First Failure Support Technology/2 <sup>®</sup>	0.3

*Table 1-2. Disk Space Requirements for OS/2*

Component	Disk Space (MB)
Device Handlers	1.4
Online Documentation	0.7
Programming Libraries	0.1
Sample Programs	0.8
Fonts	0.3

**Note:** First Failure Support Technology/2 (FFST/2)<sup>®</sup> is installed automatically if the IBM Point of Sale Subsystem Device Handlers are selected. FFST/2 requires approximately 0.5 MB of disk space.

### Memory and Disk Space Requirements for Microsoft Windows

*Table 1-3. Memory Requirements for Microsoft Windows*

Function	Recommended Memory (MB)
Device Handlers	1.0
Alphanumeric Point of Sale Keyboard	0.3
Microsoft Win32s	1.5

*Table 1-4. Disk Space Requirements for Microsoft Windows*

Component	Disk Space (MB)
Device Handlers	5.0

*Table 1-4. Disk Space Requirements for Microsoft Windows (continued)*

<b>Component</b>	<b>Disk Space (MB)</b>
Microsoft Win32s	1.7
Online Documentation	0.8
Programming Libraries	0.3
Sample Programs	0.7
Fonts	0.4

## Memory and Disk Space Requirements for Linux

*Table 1-5. Memory Requirements for Microsoft Windows*

<b>Function</b>	<b>Recommended Memory (MB)</b>
Device Handlers	1.5
Alphanumeric Point of Sale Keyboard	0.5

*Table 1-6. Disk Space Requirements*

<b>Component</b>	<b>Disk Space (MB)</b>
Device Handlers	5.0
Online Documentation	2.7
Programming Libraries	0.13
Sample Programs	0.1
Fonts	0.3



---

## Chapter 2. Universal Serial Bus Architecture and IBM Point of Sale Subsystem for Windows

The IBM Point of Sale Subsystem for Windows V2.0.0 and later provides support for USB point-of-sale devices that conform to IBM Universal Serial Bus (USB) interface specifications. USB devices are currently supported under the Microsoft Windows 98 SE and Windows 2000 operating systems. Properly written applications that use the IBM Point of Sale Subsystem for Windows API with SIO hardware should run (unmodified) with USB hardware in place of the SIO hardware.

---

### USB Device Considerations

USB devices are identified to IBM Point of Sale Subsystem applications exactly as their SIO counterparts are identified: slot number, port number, and device number. All USB devices have the same slot number and port number; device number is dynamically assigned by the IBM Point of Sale Subsystem. See Chapter 5. General Point of Sale Device Programming and Chapter 21. Resource Sets for more information about determining which devices are available.

### USB Tree Traversal Order

The USB tree traversal order is important in understanding the assignment of device role (device number). USB ports are traversed from low port number to high port number. The traversal order for the IBM Hub is from port A to port F. If a USB device is itself a hub, that hub is traversed before the traversal of the original hub continues. In the figure below, the numbers indicate the USB tree traversal order.

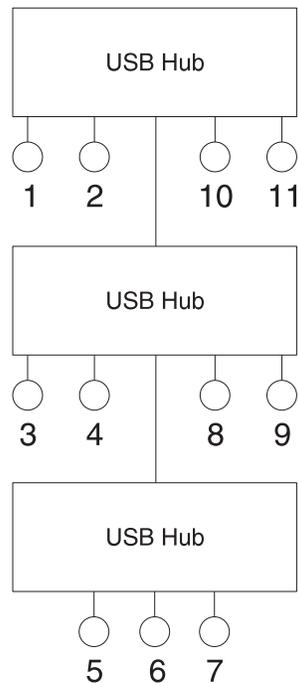


Figure 2-1. USB tree traversal order

## Device Role Assignment

In some point-of-sale situations, there is a requirement for two identical devices on the same USB lane. The assignment of a device role allows IBM Point of Sale Subsystem for Windows to uniquely identify each device.

The term *device role*, refers to the identification of a particular USB device as either primary or secondary. The notion of primary and secondary devices is taken from legacy IBM point-of-sale hardware in which a single device reports a different bus address (device number) based on which SIO channel port it is currently using. For USB devices, when two identical devices are found on the same system, the role of primary device goes to the first device found during USB tree traversal and the role of secondary device goes to the second device found.

**Note:** Currently, the IBM Point of Sale Subsystem for Windows supports the following combination of devices in the same system:

- Up to two single-sided 40-character displays
- Up to two character graphics VFDs
- Up to four all-points-addressable (APA) VFDs
- Up to two line displays:
  - One double-sided VFD or,
  - Two single-sided VFD/LCDs
- one double-sided VFD, or two single-sided VFD/LCDs
- Up to two SBCS keyboards (not including normal PC keyboards):
  - One POS system keyboard: ANPOS/ANKPOS keyboard
  - One POS keyboard: ANPOS/ANKPOS keyboard or other POS keyboard
- Up to three DBCS POS keyboards (not including normal PC keyboards):
  - One POS system keyboard: ANPOS/ANKPOS keyboard
  - One POS system keyboard on the system unit
  - One POS system keyboard (normally PLU keyboard) on the PLU extension box
- Up to two cash drawers
- Up to two MSRs
- Up to four (DBCS) APA VFDs:
  - Two on the system unit
  - Two on the PLU extension box

## USB Hot Plug Maintenance

The term *hot plug* refers to connecting a USB input/output (I/O) device to the USB without powering the host system down;

The term *hot unplug* refers to disconnecting a USB I/O device from the USB without powering the host system down.

Failing USB devices can be easily replaced by taking advantage of hot plugging and unplugging. In order to maintain device role, devices must be serviced (hot unplugged then hot plugged) one at a time.

---

## USB Alphanumeric Point of Sale Keyboards

A USB-attached Alphanumeric Point of Sale Keyboard can be configured in two ways:

- As a point-of-sale system keyboard or,
- As a point-of-sale keyboard.

Only one point-of-sale system keyboard is allowed on the host computer.

When a USB-attached Alphanumeric Point of Sale Keyboard is used as the system keyboard, the keyboard status and indicator lights are consistent with a standard Windows keyboard. The keystroke data is sent through the Microsoft Windows keyboard buffer as a standard Microsoft Windows WM\_CHAR message and is available to the application that is currently in focus. The unique point-of-sale features of the keyboard are non-functional until an IBM Point of Sale Subsystem application that uses them is started.

When a USB-attached Alphanumeric Point of Sale Keyboard is used as a point-of-sale keyboard, keystroke data is only available to IBM Point of Sale Subsystem applications. Keystroke data can be read directly from the IBM Point of Sale Subsystem or sent to the Microsoft Windows message queue as POSM\_KBD\_WM\_CHAR messages. The point-of-sale keyboard behavior is consistent with the behavior of an SIO-attached Alphanumeric Point of Sale Keyboard.



---

## Chapter 3. Customizing the IBM Point of Sale Subsystem

The IBM Point of Sale Subsystem dynamically determines which point-of-sale devices you have attached to your system. It also provides defaults for all resources associated with devices that the IBM Point of Sale Subsystem supports. This chapter discusses how the application can use a value different from the assigned default, or can even allow the user to specify some resource values.

---

### Configuring Your Applications

A *resource* in the IBM Point of Sale Subsystem, is a characteristic of appearance or behavior that is associated with an instance of a device. A resource value can be set in an application or a resource file.

A *device connection* is the connection between an application, and a hardware device created by the IBM Point of Sale Subsystem when the application opens a device. Applications can customize device connections each time a device is opened.

You have the following options when specifying the resources for the devices your application opens:

1. Specify all the resources in your application at the time the application opens, or after the application opens by using the *PosSetValues* macro. You can choose one of the following methods:
  - Specify values in your application.
  - Read values from your own configuration file, and convert them into the appropriate resource names and values.
2. Allow the users of your application to specify the resource values for the devices your application opens.
3. A combination of the above two options. Some resource values are specified by your application, and some are specified by the users of your application.

If you choose to have the users of your application specify the resource values, you should either provide a method for your users to specify the resource values to your application, or you should have them list the resource values in a resource file. If you have your users specify the resource values in a resource file, your application then specifies that file name to the IBM Point of Sale Subsystem on the *PosInitialize()* subroutine. Your program can then open devices with few or no resources on the *PosOpen()* subroutine call, and the IBM Point of Sale Subsystem looks for the resource values in the file you specified. The IBM Point of Sale Subsystem will use the default value for any resource not listed in the resource file, and not specified as a resource on the *PosOpen()* subroutine call.

### The Resource File

The resource file is an ASCII text file. The following are allowed in the resource file:

- Blank lines
- Comment records
- Resource records

#### Notes:

1. Leading blank characters on any line in the resource file are ignored.
2. The resource file is case-sensitive. Be sure to use the correct case for each of the resource names and values.

- Blank lines are ignored by the IBM Point of Sale Subsystem.

The *comment record* is any line that has an exclamation point (!) as the first non-blank character. The *resource record* consists of a qualified resource name, a colon (:), and the value for the resource. The colon separates the qualified resource name from the value being specified. A *qualified resource name* consists of the following parts separated by a period:

- Application name
- Device instance name
- Resource name

The *application name* and the *device instance name* parts are optional. An asterisk (\*) character can be substituted for the separating period to indicate a wild card match for either the application name, or the device instance name.

For example, the first line in the following example is the comment record, and the second line is the resource record.

Device resource values must be specified in the resource file if the default value is not acceptable, and if your application does not specify a value for the resource. The possible resource values and the default values for each device are listed in “Chapter 21. Resource Sets” on page 21-1. This chapter uses the `#define` constants provided for the resource name and the resource values. You should use these constant names in the resource record without the `Pos` or `PosN` prefix. For example, `PosHIGH` is a value for the resource **PosNtoneVolume**. To specify this in a resource file, enter

```
*toneVolume:          HIGH
```

The asterisk (\*) indicates that this is not a fully-qualified resource value. It is used if a more qualified resource value is not found.

In some instances, the resource value does not have a `#define` constant. In this case, you should use the values listed for the resource directly. For example:

```
*pulseWidth:          100
```

The resource file in the following example shows a checkout keyboard with a tone volume of `HIGH`, a tone frequency of `HIGH` and a tone duration of one second. The `ANPOS` keyboard has a tone volume of `LOW`, a tone frequency of `LOW` and a tone duration of 0.5 second. See “Using Resources in Your Application” on page 5-12 for information about how resources are defined and used.

```

!*****
!
! Example Resource File
!
!*****

!
! common keyboard resources
!*****
*toneVolume:          LOW
*toneFreq:            LOW
*toneDuration:       10

*till.pulseWidth:    100
*scanner.blockReadMode: 2
*scanner.beepState:  ENABLE
*checkout.toneVolume: HIGH
*checkout.toneFreq:   HIGH
*anpos.toneDuration:  5
*prnt.resumeString:  #####
    
```

## Using the Resource File

When your application issues the *PosInitialize()* subroutine, the IBM Point of Sale Subsystem builds an internal resource table comprised of the entries in the resource file specified by the *file* parameter. There is one table built for each *PosInitialize()* subroutine. The IBM Point of Sale Subsystem takes the *file* parameter of the *PosInitialize()* subroutine and looks for a file with that name in the directory specified in the environment variable POSAPPLRESDIR. If this environment variable is not set, the current directory is used.

**Note:** If the resource file is not found or cannot be read, no error is returned to the application. The default values and the values specified by your application will be used.

When a device is opened, the IBM Point of Sale Subsystem queries the internal resource table to get any application characteristics for the device being opened that were not specified using the *args* and *nargs* parameters on the *PosOpen()* subroutine call.

The application name specified by the *name* parameter on the *PosInitialize()* subroutine call and the device name specified by *name* on the *PosOpen()* subroutine are prefixed to the name of each resource required for the device being opened. This fully-qualified resource name is used to query the internal resource table. The most qualified resource specification satisfies the query. For example, if *myappl* is specified for the application *name* parameter on the *PosInitialize()* and *mydisplay* is specified for the device *name* on the *PosOpen()*, one of the resources that is queried is the resource *myappl.mydisplay.cursor*. If

```
myappl.mydisplay.cursor:      20
```

is specified in the resource file, then there is an exact match. If, however, only

```
*mydisplay.cursor:          0
```

is specified, then this entry satisfies the query, because a more qualified resource was not also specified in the resource file. For example, if the resource file contains

```
myappl.mydisplay.cursor:      20
*mydisplay.cursor:          0
```

then the cursor for the display is 20 because the most fully qualified resource name `myappl.mydisplay.cursor` is selected.

**Note:** If equally qualified duplicate definitions for a resource are found in the resource file, the value of the last, most-qualified definition is used.

---

## Configuring the Alphanumeric Point of Sale Keyboard

If your system has one of the alphanumeric point-of-sale keyboards attached as the Point of Sale system keyboard, you should use the ANPOS utility program to:

- Define double keys on the Point of Sale system keyboard. These key switches 77, 78, 82, 87, 88, 90, 94, 95, 99, 100, 105–109, 112–123, and 125–128, can be doubled.
- Override the keyboard default values for the following:
  - Numeric Keypad zero (Key 94, 99)
  - Key click
  - POS LEDs initial setting

The ANPOS utility is run automatically at boot time. The utility uses the resource file, `aipsys.res`, which is included with the IBM Point of Sale Subsystem:

- On OS/2 and Windows systems, `aipsys.res` is located in the IBM Point of Sale Subsystem root directory (default `C:\POS`).
- On Linux systems, `aipsys.res` is located in the `/etc` directory.

On OS/2, the ANPOS utility program can be run from the `STARTUP.CMD` file or from an OS/2 command prompt. On Microsoft Windows 3.1, the ANPOS utility program can be run from the `WIN.INI` file or from the Run prompt under Program Manager. The ANPOS utility program requires as a parameter the fully qualified path, and file name of a resource file containing the resources to be overridden. For example, to use the resource file `C:\POS\AIPSYS.RES` as a parameter to the ANPOS utility program the following command would be used:

```
C:\POS\BIN\AIPANPOS C:\POS\AIPSYS.RES
```

A sample resource file named `AIPSYS.RES` is included in the default `C:\POS` directory. If you installed the IBM Point of Sale Subsystem in a different location, substitute the correct directory name where `C:\POS` appears above.

### Notes:

1. Use the ANPOS utility program to set the double keys and the numeric keypad zero on an alphanumeric point-of-sale keyboard that is attached as the Point of Sale system keyboard. Requests from an application using the IBM Point of Sale Subsystem subroutines are ignored.
2. The definitions of the keyboard characteristics must use the application name `aipanpos` and the device name `system`. This allows them to be specified in the resource file used by your application. For example:

```
aipanpos.system.keyboardClick:      SOFT
aipanpos.system.doubleKey01:        77,82
aipanpos.system.doubleKey02:        90,95
```

3. The ANPOS utility records any errors in the file, `aipanpos.log`. On OS/2 and Windows systems, `aipanpos.log` is created in the LOG directory of the IBM Point of Sale Subsystem root directory (default `C:\POS\LOG`). On Linux systems, `aipanpos.log` will be created in the `/var/log` directory.
4. The AIPANPOS utility has been modified to support the USB system keyboard (there is no SIO system keyboard). The main functions of this utility are:
  - Double key table download

- POS LEDs initial setting
- Key click setting

The POS LEDs and the key click setting are not supported for the USB system keyboard.

**Note:** There is no change in the function of the Point of Sale system keyboard that is attached as the system keyboard port.

See the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book for the key switch number maps.



---

## Chapter 4. Performing Problem Determination

This chapter provides information about problem determination for the IBM Point of Sale Subsystem.

---

### Problem Determination on OS/2

The IBM Point of Sale Subsystem for OS/2 device handlers make use of the First Failure Support Technology/2 (FFST/2) software for providing problem determination services. The FFST/2 is a collection of software reliability, availability, and serviceability (RAS) functions.

The features of FFST/2 that are used by the IBM Point of Sale Subsystem for OS/2 are:

- Console services
- Customized dump
- Error logging

For more information about using FFST/2, see the *FFST/2 Administration Guide*.

### Viewing Point of Sale Error Messages on OS/2

The IBM Point of Sale Subsystem for OS/2 device handlers use internal subroutines to perform the error logging functions, and provide the appropriate parameters and data to FFST/2. FFST/2 provides the logging and maintenance of the diagnostic data along with an OS/2 Presentation Manager user interface for viewing:

- Console messages
- Customized dumps
- Logged messages

See the *FFST/2 Administration Guide* for more information about FFST/2.

IBM Point of Sale Subsystem for OS/2 customer support requires the FFST/2 error log and dump files to perform problem determination. The error log file is C:\OS2\SYSTEM\EPW\OS2MLOG.DAT. The dump files are in the C:\OS2\SYSTEM\EPW directory. Each dump file is named OS2SYS???.DMP where the ?? is an integer. FFST/2 defaults to a maximum of 32 dump files.

### Viewing Point of Sale Trace Events on OS/2

The IBM Point of Sale Subsystem for OS/2 device handlers use the OS/2 system trace facility for keeping track of major IBM Point of Sale Subsystem events. The IBM Point of Sale Subsystem for OS/2 device handlers use a major trace code of 93 (0x5D). Within this major trace code, the IBM Point of Sale Subsystem for OS/2 uses various minor trace codes for different events. The OS/2 trace facility provides user access to the system trace data. System traces can have an impact on performance, so you should only be switch traces ON during problem determination. In the normal operating environment, system traces should be off.

For more information about the OS/2 system trace facility, see the *IBM OS/2 Command Reference*. For a list of the IBM Point of Sale Subsystem minor trace codes see "Appendix B. Trace Codes" on page B-1.

---

## Problem Determination on the Microsoft Windows Operating System

The IBM Point of Sale Subsystem for Windows device handlers use event logging and tracing to assist with problem determination. However, event logging and tracing are intended for problem determination only, and should not be used in the normal operating environment.

### Viewing Events on Microsoft Windows

The IBM Point of Sale Subsystem for Windows provides an event logging facility. To see logged events, create a file called DISPLAY.ON in the C:\POS directory. Reboot your system to start viewing events.

When event viewing is enabled, the program, AIPCTRL.EXE, will be visible either on the desktop (Microsoft Windows 3.1 and Microsoft Windows NT 3.51 ), or in the Task Bar (Microsoft Windows 95, Microsoft Windows 98, Microsoft Windows NT 4.0, and Windows 2000). Click **AIPCTRL** to view events. Event information should be communicated to your IBM Point of Sale Subsystem for Windows customer support representative. Be sure to include the error ID, major return code, and minor return code.

To switch event viewing OFF, delete or rename the DISPLAY.ON file in the C:\POS directory. With event viewing switched ON, it is possible to close the AIPCTRL Window and inadvertently shut down the IBM Point of Sale Subsystem for Windows. Therefore, it is not recommended that event viewing be switched ON in a production system environment.

### Using Built-in Tracing on Microsoft Windows

The IBM Point of Sale Subsystem for Windows provides a tracing facility. To switch tracing ON, create a file called TRACE.ON in the C:\POS directory. Reboot your system to start tracing.

When tracing is enabled, a file called AIPTRACE.LOG is created in the C:\POS\LOG directory the first time IBM Point of Sale Subsystem for Windows is started after tracing has been enabled; thereafter, AIPTRACE.LOG will be appended. The AIPTRACE.LOG file should be sent to your IBM Point of Sale Subsystem for Windows customer support representative; this file is not in a human-readable format.

To switch tracing OFF, delete or rename the TRACE.ON file in the C:\POS directory.

If tracing is switched ON, the AIPTRACE.LOG file can become very large and impact system performance. Therefore, it is not recommended that tracing be switched ON in a production system environment.

---

## Problem Determination on the Linux Operating System

The IBM Point of Sale Subsystem for Linux device handlers use event logging and tracing to assist with problem determination. Event logging always occurs for Linux, but tracing is intended only for problem determination and should not be enabled in the normal operating environment.

## Viewing Events on Linux

To see any logged events on a Linux system, look at the `aipctrl.log` file in `/var/log`. Event information should be communicated to your IBM Point of Sale Subsystem customer support representative. Be sure to include the error ID, major return code, and minor return code.

## Using Tracing on Linux

The IBM Point of Sale Subsystem for Linux provides a tracing facility. To turn tracing ON, modify the file called `aipsys.conf` that is in the `/etc` directory. Assure that there is a line in the file as follows:

```
trace=on
```

Reboot your system to start tracing.

When tracing is enabled, a file called `aiptrace.log` is created in the `/var/log` directory the first time that IBM Point of Sale Subsystem is started; thereafter, `aiptrace.log` will be appended. The `aiptrace.log` file should be sent to your IBM Point of Sale Subsystem customer support representative; this file is not in a human-readable format.

If tracing is ON, the `aiptrace.log` file can become very large and can impact system performance. It is not recommended that tracing be left on in a store environment.



---

## Part 2. Programming Guide

<b>Chapter 5. General Point of Sale Device Programming.</b>	5-1
Your Application and the IBM Point of Sale Subsystem	5-1
Initializing Your Application	5-1
Getting Input Messages	5-2
Determining which Devices are Available	5-3
Opening Your Device	5-4
Controlling Your Device	5-5
Acquiring and Releasing Your Devices	5-6
Acquiring and Releasing Devices from a Presentation Facility Application	5-7
Acquiring and Releasing Devices from a Non-Presentation Facility Application	5-8
Defining User-Defined Characters.	5-8
Reading from Your Device	5-9
Writing to Your Device	5-10
Closing Your Device Connections	5-10
Terminating Your Application	5-11
OS/2 Exit List Processing	5-11
Using Resources in Your Application	5-12
Argument Lists	5-12
Setting Argument List Values	5-12
Retrieving and Modifying Resources	5-13
Building your Application	5-14
C Language Header Files	5-14
IBM Point of Sale Subsystem Libraries	5-15
Optimizing Application Performance	5-16
Application Priority (OS/2 Only)	5-16
Presentation Manager Considerations (OS/2 Only)	5-16
Microsoft Windows Considerations	5-17
Polling Considerations	5-17
Multi-threaded Application Design	5-18
Improving the Maintainability of Your Application	5-19
<b>Chapter 6. Alarm Programming</b>	6-1
Characteristics of the Alarm	6-1
Functions Your Application Performs.	6-1
Sounding an Alarm	6-1
Silencing an Alarm	6-1
Getting Alarm Status	6-2
Related Information	6-2
Subroutines Used with Alarm	6-2
Alarm PosIOctl() Control Requests	6-2
Alarm Resources	6-2
Alarm Error Codes	6-2
<b>Chapter 7. Cash Drawer Programming</b>	7-1
Characteristics of the Cash Drawer	7-1
Functions Your Application Performs.	7-1
Opening a Cash Drawer Till	7-1
Getting Cash Drawer Status.	7-2
Setting Cash Drawer Pulse Width.	7-2
Related Information	7-2
Subroutines Used with Cash Drawer	7-2
Cash Drawer PosIOctl() Control Requests	7-2

Cash Drawer Event Messages . . . . .	7-3
Cash Drawer Resources . . . . .	7-3
Cash Drawer Error Codes . . . . .	7-3
<b>Chapter 8. Display Programming . . . . .</b>	<b>8-1</b>
Characteristics of the Displays . . . . .	8-1
Alphanumeric Display . . . . .	8-1
Operator Display . . . . .	8-2
Shopper Display . . . . .	8-2
Character and Graphics Display . . . . .	8-2
40-Character Liquid Crystal Display . . . . .	8-3
40-Character Vacuum Fluorescent Display II and 2x20 Character VFD Customer Display . . . . .	8-3
Two-Sided Vacuum Fluorescent Display II. . . . .	8-3
Functions Your Application Performs. . . . .	8-3
Code Page Support . . . . .	8-4
Writing Characters to the Display . . . . .	8-4
Writing Bitmaps to the Display . . . . .	8-5
Setting the Guidance Lights . . . . .	8-6
Clearing the Display Screen. . . . .	8-6
User-Defined Characters . . . . .	8-6
Related Information . . . . .	8-7
Subroutines Used with Displays . . . . .	8-8
Display PosiOctl() Control Requests . . . . .	8-8
Display Resources . . . . .	8-8
Display Error Codes. . . . .	8-8
<b>Chapter 9. Keyboard Programming . . . . .</b>	<b>9-1</b>
Characteristics of the Keyboards . . . . .	9-1
Keyboard Microcode Updates . . . . .	9-2
50-Key Modifiable Layout Keyboard and 50-Key Modifiable Layout Keyboard and Operator Display . . . . .	9-3
Retail Point of Sale Keyboards. . . . .	9-3
Modifiable Layout Keyboard with Card Reader . . . . .	9-4
ANPOS Keyboard . . . . .	9-5
Retail Alphanumeric Point of Sale Keyboard with Card Reader . . . . .	9-6
PC Point of Sale Keyboard (ANKPOS Keyboard) . . . . .	9-7
Point of Sale Keyboard V. . . . .	9-8
PLU Keyboard and Display-III . . . . .	9-9
4685 Point of Sale Keyboard Model K01. . . . .	9-10
IBM 4820 SurePoint Solution Keypad . . . . .	9-10
Defining Keys. . . . .	9-11
Restriction of the Keyboard Device Handler . . . . .	9-12
Functions Your Application Performs . . . . .	9-12
Reading Keyboard Data. . . . .	9-12
Using the Manager Keyboard Lock. . . . .	9-13
Using the Keyboard Tone . . . . .	9-13
Using the Keyboard Point of Sale Lights. . . . .	9-13
Controlling the Keyboard Click . . . . .	9-14
Controlling the Num Lock Key . . . . .	9-14
Controlling the Scroll Lock key . . . . .	9-14
Controlling the Point of Sale-Unique Keys . . . . .	9-14
Controlling the System Hot Keys . . . . .	9-15
Controlling the Keyboard Typematic Function . . . . .	9-15
Specifying the Numeric Keypad Style . . . . .	9-15
Specifying the Numeric Keypad Location . . . . .	9-15

- Related Information . . . . . 9-15
  - Subroutines Used with Keyboard . . . . . 9-15
  - Keyboard PosIOCtl() Control Requests . . . . . 9-15
  - Keyboard Event Messages. . . . . 9-16
  - Keyboard Resources . . . . . 9-16
  - Keyboard Error Codes . . . . . 9-16
  
- Chapter 10. Magnetic Stripe Reader Programming . . . . . 10-1**
  - Characteristics of the MSR. . . . . 10-1
    - One-Track Magnetic Stripe Reader. . . . . 10-1
    - Dual-Track Magnetic Stripe Reader . . . . . 10-2
    - Three-Track Magnetic Stripe Reader . . . . . 10-2
    - Two-Head/Two-Sided Magnetic Stripe Reader. . . . . 10-3
  - Restriction of the MSR Device Handler . . . . . 10-3
  - Functions Your Application Performs . . . . . 10-3
    - Unlocking the MSR . . . . . 10-3
    - Reading MSR Data . . . . . 10-4
      - MSR Read Buffer Format . . . . . 10-4
    - Locking the MSR . . . . . 10-5
  - Related Information . . . . . 10-5
    - Subroutines Used with MSR . . . . . 10-6
    - MSR PosIOCtl() Control Requests . . . . . 10-6
    - MSR Event Messages . . . . . 10-6
    - MSR Error Codes . . . . . 10-6
  
- Chapter 11. Non-Volatile Random Access Memory Programming . . . . . 11-1**
  - Characteristics of the NVRAM Device . . . . . 11-1
  - Functions Your Application Performs . . . . . 11-2
    - Available NVRAM Application Address Space . . . . . 11-2
    - The Cursor Position . . . . . 11-2
    - Opening NVRAM in Direct Mode or Sequential Mode . . . . . 11-3
    - Reading Data in Direct Mode or Sequential Mode . . . . . 11-3
    - Writing Data in Direct Mode or Sequential Mode . . . . . 11-3
  - Related Information . . . . . 11-4
    - Subroutines Used with NVRAM . . . . . 11-4
    - NVRAM PosIOCtl() Control Requests . . . . . 11-4
    - NVRAM Resources . . . . . 11-4
    - NVRAM Error Codes . . . . . 11-4
  
- Chapter 12. Printer Programming . . . . . 12-1**
  - Characteristics of the Printers. . . . . 12-1
    - IBM Model 2 Printer . . . . . 12-1
      - Print Mechanism . . . . . 12-1
      - Fonts. . . . . 12-2
      - Line Length . . . . . 12-2
      - Line Spacing . . . . . 12-2
      - Emphasized Printing . . . . . 12-2
      - Performance . . . . . 12-2
    - IBM Model 3 and IBM Model 4 Printers . . . . . 12-2
      - Print Mechanism . . . . . 12-3
      - Fonts. . . . . 12-3
      - Line Length . . . . . 12-3
      - Line Spacing . . . . . 12-3
      - Emphasized Printing . . . . . 12-4
      - Font Specification . . . . . 12-4
      - Performance . . . . . 12-4

IBM Model 3R and Model 4R Printers . . . . .	12-4
IBM Model 3F Fiscal Printer . . . . .	12-4
IBM Model 3F Fiscal Printer Restrictions. . . . .	12-4
IBM Model 4A Printer . . . . .	12-5
Addressable Print Positions Per Line . . . . .	12-5
Addressable Print Positions Per Character . . . . .	12-5
IBM 4689-001 and IBM 4689-002 Printer . . . . .	12-6
Print Mechanism . . . . .	12-6
Fonts. . . . .	12-6
Line Length . . . . .	12-6
Line Spacing . . . . .	12-6
Font Specification . . . . .	12-6
Performance . . . . .	12-6
Restrictions . . . . .	12-6
IBM 4689 Point of Sale Printer Model 301, 3G1, 3M1, and TD5 . . . . .	12-7
Print Mechanism . . . . .	12-7
Fonts. . . . .	12-7
Line Length . . . . .	12-7
Line Spacing . . . . .	12-8
Print Direction . . . . .	12-8
Contrast (Amikake) . . . . .	12-8
Line Characters (Keisen) . . . . .	12-8
Performance . . . . .	12-8
Restrictions . . . . .	12-8
IBM 4610 SureMark Point of Sale Printer . . . . .	12-8
Print Mechanism . . . . .	12-9
Fonts. . . . .	12-9
Line Length . . . . .	12-9
Line Spacing . . . . .	12-10
Text Print Attributes . . . . .	12-10
Barcode Printing . . . . .	12-10
Pre-defined Messages and Logos. . . . .	12-10
Font Specification. . . . .	12-10
Performance . . . . .	12-11
MICR Recognition and Check Flipping . . . . .	12-11
IBM 4610 SureMark Point of Sale Fiscal Printer . . . . .	12-11
Functions Your Application Performs . . . . .	12-11
Code Page Support . . . . .	12-11
Reading Data from the Printer . . . . .	12-12
Reading MICR Data. . . . .	12-12
MICR Information. . . . .	12-13
Reading Fiscal Data. . . . .	12-13
Writing Data to the Printer . . . . .	12-14
Writing Data in Normal Mode . . . . .	12-14
Printing a Line of Text at the Printer . . . . .	12-15
Printer Errors . . . . .	12-15
Changing the Print Characteristics . . . . .	12-15
Line Buffering . . . . .	12-17
Determining When Printing is Complete . . . . .	12-17
Font Interactions . . . . .	12-17
Writing Data in Logo Mode . . . . .	12-18
IBM Model 2, Model 3, Model 4, and Model 4A Printers. . . . .	12-18
IBM 4610 SureMark Printer . . . . .	12-18
4689 Models 3x1 and TD5 Printers . . . . .	12-19
Printer Errors . . . . .	12-19
Writing Data in Download Message Mode (4610 SureMark Printers Only) . . . . .	12-19

Writing Data in Download Logo Mode . . . . .	12-20
IBM 4610 SureMark Point of Sale Printers . . . . .	12-20
IBM 4689 Models 3x1 and TD5 Printers . . . . .	12-20
Printer Errors . . . . .	12-20
Writing Data in Fiscal Mode (Fiscal Printer Only) . . . . .	12-20
Printing a Line of Text at the Printer . . . . .	12-21
Printer Errors . . . . .	12-21
Control Characters . . . . .	12-21
Escape Character Sequences . . . . .	12-22
Printer Input/Output Control Requests (IOCtl) . . . . .	12-33
Printer Resources . . . . .	12-35
Printer Event Messages . . . . .	12-35
Determining the Printer Status . . . . .	12-35
Printer Queues. . . . .	12-35
Document Insert Station . . . . .	12-36
IBM Model 2 Printer . . . . .	12-36
IBM Model 3, Model 3F, Model 3R, Model 4, Model 4A, and Model 4R Printers . . . . .	12-37
IBM 4689-001 and IBM 4689-002 . . . . .	12-39
IBM 4610 SureMark Point of Sale Printers . . . . .	12-41
Receipt Paper Cutter . . . . .	12-42
Printing Checks . . . . .	12-43
MICR Reader . . . . .	12-43
Fiscal Printing . . . . .	12-44
User-Defined Characters . . . . .	12-44
IBM 4689-00x in 25 CPL Mode. . . . .	12-44
IBM 4689-00x in 30 CPL Mode. . . . .	12-45
IBM 4689-301, 3G1, 3M1, and TD5 . . . . .	12-46
Performance Considerations. . . . .	12-47
Related Information . . . . .	12-48
Subroutines Used with the Printer. . . . .	12-48
Printer PosIOCtl Control Requests . . . . .	12-48
Printer Event Messages . . . . .	12-48
Printer Resources . . . . .	12-48
Printer Error Codes . . . . .	12-49
<b>Chapter 13. Programmable Power Programming.</b> . . . . .	13-1
Characteristics of the Programmable Power Device. . . . .	13-1
Functions Your Application Performs . . . . .	13-1
Turning Power Off to a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 . . . . .	13-2
Turning Power On and Off to the 4693-2x2. . . . .	13-2
Querying the Time that Power Is to Be Turned On . . . . .	13-3
Related Information . . . . .	13-3
Subroutines Used with Programmable Power Subsystem . . . . .	13-3
Programmable Power Subsystem PosIOCtl() Control Requests . . . . .	13-3
Programmable Power Resources . . . . .	13-4
Programmable Power Device Error Codes . . . . .	13-4
<b>Chapter 14. RS-232C Programming</b> . . . . .	14-1
Characteristics of the RS-232C Port . . . . .	14-2
Functions Your Application Performs . . . . .	14-2
Controlling the RS-232C Port . . . . .	14-2
Reading RS-232C Data . . . . .	14-3
Error Definitions. . . . .	14-3
Writing RS-232C Data . . . . .	14-4
Getting RS-232C Port Status . . . . .	14-4

Related Information . . . . .	14-4
Subroutines Used with RS-232C. . . . .	14-4
RS-232C PosIOCtl() Control Requests . . . . .	14-5
RS-232C Resources . . . . .	14-5
RS-232C Event Messages . . . . .	14-5
RS-232C Error Codes . . . . .	14-5
<b>Chapter 15. Scale Programming . . . . .</b>	<b>15-1</b>
Characteristics of the Scale Devices . . . . .	15-1
IBM 4687 Point of Sale Sale Scanner Model 2 . . . . .	15-1
IBM 4696 Point of Sale Scanner Scale Model 1 . . . . .	15-1
IBM 4698 Point of Sale Scanner Model 2 . . . . .	15-2
IBM USB Scale Interface . . . . .	15-2
Functions Your Application Performs . . . . .	15-2
Reading Scale Data . . . . .	15-2
Configuring the Scale. . . . .	15-4
Zeroing the Scale . . . . .	15-4
Clearing the Scale Display . . . . .	15-4
Scale Default Values . . . . .	15-4
IBM 4687 Point of Sale Scanner Model 2 . . . . .	15-5
IBM 4696 Point of Sale Scanner Scale Model 1 and IBM 4698 Point of Sale Scanner Model 2 . . . . .	15-5
IBM USB Scale Interface . . . . .	15-5
Related Information . . . . .	15-5
Subroutines Used with Scale . . . . .	15-5
Scale PosIOCtl() Control Requests. . . . .	15-6
Scale Resources . . . . .	15-6
Scale Error Codes . . . . .	15-6
<b>Chapter 16. Scanner Programming . . . . .</b>	<b>16-1</b>
Characteristics of the Scanners . . . . .	16-1
Hand-Held Bar Code Readers . . . . .	16-1
IBM 1520 Hand-Held Scanner Model A02 . . . . .	16-2
IBM 4686 Retail Point of Sale Scanner . . . . .	16-2
IBM 4687 Point of Sale Scanner. . . . .	16-2
IBM 4696 Point of Sale Scanner Scale . . . . .	16-2
IBM 4697 Point of Sale Scanner. . . . .	16-3
IBM 4698 Point of Sale Scanner. . . . .	16-3
IBM USB Scanner Interface . . . . .	16-3
Functions Your Application Performs . . . . .	16-4
Unlocking and Locking the Scanner . . . . .	16-4
Reading Scanner Data . . . . .	16-4
Discarding Scanner Data . . . . .	16-8
Configuring the Scanner. . . . .	16-8
Writing Data to the Scanner . . . . .	16-8
Processing Unexpected Scanner Data . . . . .	16-9
Scanner Default Values . . . . .	16-9
Hand-Held Bar Code Reader (All Models). . . . .	16-10
IBM 1520 Hand-Held Scanner Model A02. . . . .	16-10
IBM 4686 Retail Point of Sale Scanner (All Models) . . . . .	16-10
IBM 4687 Point of Sale Scanner (All Models) . . . . .	16-11
IBM 4696 Point of Sale Scanner Scale Model 1. . . . .	16-11
IBM 4697 Point of Sale Scanner Model 1 . . . . .	16-11
IBM 4698 Point of Sale Scanner (All Models) . . . . .	16-12
IBM USB Scanner Interface . . . . .	16-12
Related Information . . . . .	16-13

Subroutines Used with Scanner . . . . .	16-13
Scanner PosIOctl() Control Requests . . . . .	16-13
Scanner Event Messages. . . . .	16-13
Scanner Resources . . . . .	16-14
Scanner Error Codes . . . . .	16-14
<b>Chapter 17. Touch Screen Programming . . . . .</b>	<b>17-1</b>
Characteristics of the Touch Screen . . . . .	17-1
Video Display. . . . .	17-1
Touch Input . . . . .	17-1
Touch Screen Microcode Updates . . . . .	17-2
Tone Output . . . . .	17-2
Touch Mouse Emulation . . . . .	17-2
Touch Mouse Emulation on OS/2 . . . . .	17-2
Touch Mouse Emulation on the Microsoft Windows Operating System . . . . .	17-2
Restriction of the Touch Screen Device Handler . . . . .	17-3
Functions Your Application Performs . . . . .	17-3
Reading Touch Event Data. . . . .	17-3
Using the Tone . . . . .	17-3
Controlling Audible Feedback . . . . .	17-4
Determining which Touch Screen is Available . . . . .	17-4
Controlling the LCD Brightness . . . . .	17-4
Controlling the LCD Contrast . . . . .	17-4
Controlling the Screen Saver Time . . . . .	17-5
Controlling the Backlight On Event Messages . . . . .	17-5
Related Information . . . . .	17-5
Subroutines Used with the Touch Screen . . . . .	17-5
Touch PosIOctl Control Requests . . . . .	17-5
Touch Event Messages . . . . .	17-5
Touch Resources . . . . .	17-5
Touch Error Codes. . . . .	17-6
<b>Chapter 18. Application Programming Interface . . . . .</b>	<b>18-1</b>
PosClose(). . . . .	18-3
PosInitialize(). . . . .	18-5
PosIOctl(). . . . .	18-8
PosOpen(). . . . .	18-10
PosRead(). . . . .	18-16
PosWrite(). . . . .	18-19

created on October 2, 2001

---

## Chapter 5. General Point of Sale Device Programming

This chapter explains the necessary steps for using the IBM Point of Sale Subsystem with your application program.

---

### Your Application and the IBM Point of Sale Subsystem

Your application can be one of many applications that uses the IBM Point of Sale Subsystem concurrently. In addition, your application can share access to a point-of-sale device with other applications. The following sections describe the general interaction between your application and the IBM Point of Sale Subsystem.

### Initializing Your Application

The first thing your application must do before using the IBM Point of Sale Subsystem to access any of the supported point-of-sale devices is to register with the subsystem by calling the *PosInitialize()* subroutine.

When you call the *PosInitialize()* subroutine, the following resources can be used to control how the IBM Point of Sale Subsystem operates:

- PosNqueueHandle
- PosNreadTimeout

Use the **PosNqueueHandle** resource to specify the input queue that will receive system event messages. See “PosInitialize()” on page 18-5 for details.

Use the **PosNreadTimeout** resource to specify how long your application waits for an event message on the IBM Point of Sale Subsystem input queue. The default value for this resource is zero (0). This causes the *PosRead()* subroutine call to return immediately if there are no event messages in the queue. If your application changes the default value for this resource to -1, the read will wait indefinitely until an event message is available for your application. Any other non-zero value indicates the number of milliseconds the *PosRead()* subroutine should wait for an event message to become available for your application.

If no event message becomes available within the number of milliseconds requested, the 329 POSERR\_SYS\_TIMEOUT error code is returned to your application.

#### Notes:

1. The **PosNreadTimeout** resource is used only when your application reads from the IBM Point of Sale Subsystem input queue. It is not used when calling *PosRead()* for individual devices, or for the presentation facility queue.
2. The **PosNreadTimeout** resource is ignored for applications compiled with a 16-bit compiler for the Microsoft Windows operating system, or for 32-bit applications using Microsoft Win32s.
3. The **PosNqueueHandle** resource is ignored on systems that use the IBM Point of Sale Subsystem for Linux. Linux systems must use the IBM Point of Sale Subsystem input queue.

The following example shows how to register your application with the IBM Point of Sale Subsystem:

```
#include <pos/pos.h>
```

```

int rc;

/** Register the application */
rc = PosInitialize( "myappl",      /* application name */
                  "checkout",    /* resource file name */
                  0,             /* argc from command line */
                  0,             /* argv from command line */
                  0,             /* no resources overridden */
                  0 );          /* number of resources */

```

## Getting Input Messages

Non-presentation facility applications obtain event messages, such as user input and device errors, by calling the *PosRead()* subroutine with a device descriptor of zero. The device descriptor of zero indicates that the read is directed to the IBM Point of Sale Subsystem input queue. The IBM Point of Sale Subsystem input queue is treated as a special input device. If you specify that you want your application program to receive event messages on the IBM Point of Sale Subsystem input queue, it must read the event messages from the queue. If the application program does not read the event messages, the queue will eventually become full and event messages will be lost. If this occurs, the IBM Point of Sale Subsystem logs an error.

The following example shows how to read data from the IBM Point of Sale Subsystem input queue:

```

#include <pos/pos.h>

int rc;
POSQMSG qmsg;                                /* defined in pos/pos.h */

/** Register the application */
rc = PosInitialize( "myappl",      /* application name */
                  "checkout",    /* resource file name */
                  0,             /* argc from command line */
                  0,             /* argv from command line */
                  0,             /* no resources overridden */
                  0 );          /* number of resources */

/** Read the input queue */
rc = PosRead( 0,                    /* indicate input queue */
             &qmsg,                /* buffer address */
             sizeof(qmsg) );       /* buffer length */

```

The *PosRead()* subroutine for a device and any subsequent *PosIOctl()* requests use the device descriptor obtained from the *PosOpen()* subroutine call for that device.

On OS/2, Presentation Manager applications can use the Presentation Manager input queue instead of the IBM Point of Sale Subsystem input queue by specifying the appropriate value for the **PosQueueHandle** resource on the *PosInitialize()* subroutine, and the **PosQueueHandle** resource on each *PosOpen()* subroutine.

On Microsoft Windows, applications written for Microsoft Windows can use the windows message queue instead of the IBM Point of Sale Subsystem input queue by specifying the appropriate window handle for the **PosQueueHandle** resource on the *PosInitialize()* subroutine, and the **PosQueueHandle** resource on each *PosOpen()* subroutine.

On Linux systems, the IBM Point of Sale Subsystem input queue must be used. The **PosNqueueHandle** resource is ignored on systems that use the IBM Point of Sale Subsystem for Linux.

The **PosNqueueHandle** resource for the *PosInitialize()* subroutine applies only to system event messages, and the **PosNqueueHandle** resource for the *PosOpen()* subroutine applies to device specific event messages. If your application is a Presentation Manager application, or a Microsoft Windows application, it should always override these resources on both the *PosInitialize()* and the *PosOpen()* subroutine calls.

See Chapter 20. Event Messages for a complete list of the event messages.

## Determining which Devices are Available

The IBM Point of Sale Subsystem tells your application which devices are attached to the system by putting a POSM\_SYS\_DEVICE\_ONLINE event message on the queue specified by the **PosNqueueHandle** resource on the *PosInitialize()* subroutine call.

Each event message consists of an event message identifier and two message parameters, *mp1* and *mp2*. For the Microsoft Windows operating system, *mp1* corresponds to *wParam* and *mp2* corresponds to *lParam*. Your application uses the fields in the *mp1* and *mp2* parameters to determine if a particular online event message is for a device that you want it to work with. Use the device type field to validate the type of device. Use the slot number field, the port number field, and the device number field to verify that the device is plugged into a location that your application supports. Use the sub-type field in *mp2* to distinguish between multiple devices that have the same device number. See Chapter 20. Event Messages for more information about the *mp1* and *mp2* message parameters. See "POSM\_SYS\_DEVICE\_ONLINE" on page 20-22 for more information about the POSM\_SYS\_DEVICE\_ONLINE event message.

Once your application detects an online event message for a device that you want it to use, it must issue the *PosOpen()* subroutine call to open a connection to the device.

The following example shows how to read data from the IBM Point of Sale Subsystem input queue to determine which devices are online:

```
#include <pos/pos.h>
#include <pos/scanner.h>

int rc; /* Function return code */
int DevType = 0; /* Device type from ONLINE msg */
int SubType = 0; /* SubType from ONLINE msg */
POSQMSG qmsg; /* defined in pos/pos.h */

/** Register the application */
rc = PosInitialize( "myappl", /* application name */
                  "checkout", /* resource file name */
                  0, /* argc from command line */
                  0, /* argv from command line */
                  0, /* no resources overridden */
                  0 ); /* number of resources */

/** Read the input queue */
rc = PosRead( 0, /* indicate input queue */
             &qmsg, /* buffer address */
             sizeof(qmsg) ); /* buffer length */
```

```

/** Look for online messages */
if ( POSM_SYS_DEVICE_ONLINE == qmsg.msg )
{
    DevType = &lparint) pMsg->mp1 & 0x00FF;

    switch(DevType)
    {
        case PosTYPE_SCANNER:
        {
            SubType = (int) qmsg.mp2 & 0x00FF;
            switch(SubType)
            {
                case PosSCAN_SUBTYPE_4696:
                {
                    /*
                     * Code to open the IBM 4696 scanner goes here
                     */
                    break;
                } /* end case (PosSCAN_SUBTYPE_4696) */
            } /* end switch (SubType) */
            break;
        } /* end case (PosTYPE_SCANNER) */

        .
        .
        .

        default:
        {
            break;
        }

    } /* end switch (DevType) */
} /* end if ( POSM_SYS_DEVICE_ONLINE == qmsg.msg ) */

```

## Opening Your Device

Any application that intends to use a device must first issue a *PosOpen()* subroutine call for that device. The *PosOpen()* subroutine opens a device connection between the application and the device. A *PosOpen()* subroutine call must be issued for each device that you want your application to use. A successful *PosOpen()* subroutine call returns the device descriptor.

It is possible for an application to have multiple device connections for the same device. For example: One application opens two connections to the keyboard. A second application opens one connection to the keyboard. The IBM Point of Sale Subsystem will then have three device connections that share the same keyboard. The applications must coordinate their use of this one keyboard.

In order to open a device using the *PosOpen()* subroutine, you must identify the resource set that represents the type of device that you want to use. A *resource set* is the set of resources associated with a particular device class. A resource set is identified to the *PosOpen()* subroutine by the *class* parameter. Using the appropriate resource set, issue a *PosOpen()* subroutine call for each device that you want to open. See “PosOpen()” on page 18-10 for more information about the *PosOpen()* subroutine.

Use the **PosNqueueHandle** resource on the *PosOpen()* subroutine to specify the application’s input queue that will receive device specific event messages. See “PosNqueueHandle” on page 21-14 for more information.

**Note:** The `PosNqueueHandle` resource is ignored on systems that use the IBM Point of Sale Subsystem for Linux. Linux systems must use the IBM Point of Sale Subsystem input queue.

The following example shows how to open a device:

```
#include <pos/pos.h>

int msrdes;
char * mymsr = "msr";
PosArg resources[3];

/** Register the application */
rc = PosInitialize( "myappl",          /* application name */
                  "checkout",        /* resource file name */
                  0,                  /* argc from command line */
                  0,                  /* argv from command line */
                  0,                  /* no resources overridden */
                  0 );               /* number of resources */

/** Setup the resources for opening the device */
resources[ 0 ].name = PosNslotNumber;
resources[ 0 ].value = PosSLOT_1;
resources[ 1 ].name = PosNportNumber;
resources[ 1 ].value = PosPORT_1;
resources[ 2 ].name = PosNdeviceNumber;
resources[ 2 ].value = PosDEVICE_MSR_3_TRACK_A;

/** Open the MSR */
msrdes = PosOpen( mymsr,              /* name in resource file */
                 PosMsr,              /* resource set */
                 &resources,         /* resources overridden */
                 3 );                /* number of resources */
```

## Controlling Your Device

Your application can control how the point-of-sale devices work by using the `PosIOctl()` subroutine call. The `PosIOctl()` subroutine takes as parameters a device descriptor, the input/output control command request, an argument list, and the number of arguments in the argument list. The device descriptor must be a valid device descriptor opened by the `PosOpen()` subroutine, the input/output request is specific to each device, and the argument list is used to specify undefined resource values or to override resource values previously defined.

Upon successful completion, the `PosIOctl()` subroutine returns a 0 (zero). If an error occurs, a -1 will be returned and `errno` is set to indicate the error. See “`PosIOctl()`” on page 18-8 and Chapter 19. `PosIOctl()` Requests for more information.

The following example shows how to control a device:

```
#include <pos/pos.h>

int kbddes;
int rc;
char * mykbd = "kbd";

/** Register the application */
rc = PosInitialize( "myappl",          /* application name */
                  "checkout",        /* resource file name */
                  0,                  /* argc from command line */
                  0,                  /* argv from command line */
                  0,                  /* no resources overridden */
                  0 );               /* number of resources */
```

```

/** Open the keyboard */
kbddes = PosOpen( mykbd,          /* name in resource file */
                 PosKeyboard,    /* resource set */
                 0,              /* no resources overridden */
                 0 );           /* number of resources */

/** Acquire exclusive use of the keyboard */
rc = PosIOctl( kbddes,          /* descriptor from PosOpen */
              POS_SYS_ACQUIRE_DEVICE, /* request */
              0,              /* no resources overridden */
              0 );           /* number of resources */

/** Sound the keyboard tone */
rc = PosIOctl( kbddes,          /* descriptor from PosOpen */
              POS_KBD_SOUND_TONE, /* request */
              0,              /* no resources overridden */
              0 );           /* number of resources */

```

## Acquiring and Releasing Your Devices

IBM Point of Sale Subsystem devices can be shared between applications if the applications coordinate the acquiring and releasing of the devices. Your application can acquire exclusive use of a device and not release the device if you do not want any other application to use the device.

Once your application has created a device connection by using the *PosOpen()* subroutine call, it must use the `POS_SYS_ACQUIRE_DEVICE` *PosIOctl()* request to acquire the device. As long as no other application has the device acquired, the application requesting the device is granted exclusive use of it. If some of the devices required by your application are acquired by another application, do not send repetitive *PosIOctl()* requests to try to acquire them. Instead, wait until the `POSM_SYS_DEVICE_RELEASED` event message is posted to the message queues that were specified on the *PosOpen()* call for those devices.

All device connections require that you acquire exclusive use of the device before operations using the device connection affect the device. Usually, setup operations using the `POS_SYS_SET_VALUES` *PosIOctl()* request can be done using the device connection without acquiring the device. However, the setup operation will not take effect until you have acquired exclusive use of the device. Issue the `POS_SYS_ACQUIRE_DEVICE` request on the *PosIOctl()* subroutine to acquire the device. Issue the `POS_SYS_RELEASE_DEVICE` request on the *PosIOctl()* subroutine to release the device.

After your application has acquired exclusive use of a device, it will receive all event messages for that device. These event messages are put on the queue that was specified on the *PosOpen()* subroutine. See Chapter 20. Event Messages for a complete list of event messages.

If an application has opened the same device more than once, the application has multiple connections for the device. However, the application can only have one device connection to that device acquired at a given time. When an application releases a device, the device is available for any application to acquire it.

### Notes:

1. If the target device is not acquired, all *PosIOctl()* requests fail, except for `POS_SYS_ACQUIRE_DEVICE`, `POS_SYS_GET_VALUES`, and `POS_SYS_SET_VALUES`, with error code 315 `POSERR_SYS_NOT_ACQUIRED`.

- If the target device is not acquired, all other IBM Point of Sale Subsystem subroutine calls fail, except for *PosClose()*, *PosOpen()*, and *PosInitialize()*, with error code 315 POSERR\_SYS\_NOT\_ACQUIRED.

The following example shows how to acquire and release a device:

```
#include <pos/pos.h>

int msrdes;
int rc;
char * mymsr = "msr";

/** Register the application */
rc = PosInitialize( "myappl",          /* application name */
                  "checkout",        /* resource file name */
                  0,                  /* argc from command line */
                  0,                  /* argv from command line */
                  0,                  /* no resources overridden */
                  0 );                /* number of resources */

/** Open the MSR */
msrdes = PosOpen( mymsr,              /* name in resource file */
                 PosMsr,             /* resource set */
                 0,                  /* no resources overridden */
                 0 );                /* number of resources */

/** Acquire exclusive use of the MSR */
rc = PosIOctl( msrdes, POS_SYS_ACQUIRE_DEVICE, 0, 0 );

.
.
.

/** Release exclusive use of the MSR */
rc = PosIOctl( msrdes, POS_SYS_RELEASE_DEVICE, 0, 0 );
```

### Acquiring and Releasing Devices from a Presentation Facility Application

The guidelines for presentation facility applications (for example, Presentation Manager) that need to cooperatively share a single device are:

- When your application is told that none of its windows has the focus any longer, it should release all acquired devices.
- When your application is told that one of its windows now has the focus, it should try to acquire only the devices it requires.

Because the presentation facility informs the application losing the focus of the focus change event before informing the application gaining the focus, the release of the device occurs before a subsequent acquire is issued.

These are only guidelines. However, there can be times when you choose not to have your application relinquish control of a particular device. For example, suppose there are times when your checkout application cannot share the scanner. When your application has the focus, it acquires exclusive use of the scanner. When the user decides to open a window to run a calculator program, your checkout program is told that it has lost the focus. You can choose not to release the scanner because you know that the application is in the middle of a transaction.

In the previous example, the calculator program receives an error if it attempts to acquire the scanner. The calculator program can display an information box informing the user that it was unable to acquire the scanner, and as such is unable to fulfill any application function that requires use of the scanner.

## Acquiring and Releasing Devices from a Non-Presentation Facility Application

Applications that do not use a presentation facility application (like Presentation Manager) are assumed to be text-based applications running in a text window or a full screen.

**Note:** DOS applications running on the Microsoft Windows operating system are not supported by the IBM Point of Sale Subsystem for Windows on Microsoft Windows 3.1.

Applications running in this environment do not have a presentation facility that tells the application when it is losing or gaining focus. Because of this, applications in this environment are responsible for designing their own guidelines for coordinating multiple application use of a single device. Some methods that can be used singularly or in conjunction with one another are:

- Applications developed that can detect other similar applications
- Applications developed to provide an interface for the user to indicate when to use which devices
- Applications developed with flexibility to handle situations when devices cannot be immediately acquired

## Defining User-Defined Characters

In order to define characters to a point-of-sale device, your application must call the *PosIOctl()* subroutine. The *PosIOctl()* subroutine takes as parameters a device descriptor, the input/output control command request, an argument list, and the number of arguments in the argument list. The device descriptor must be a valid device descriptor opened by the *PosOpen()* subroutine, the input/output request is specific to each device, and the argument list is used to specify the new user defined characters. For the Character/Graphics Display, the input/output request is `POS_DSP_DEFINE_CHARACTERS` and for the IBM 4689-00x printers, the input/output request is `POS_PRN_DEFINE_CHARACTERS`.

Upon successful completion, the *PosIOctl()* subroutine returns a zero (0). If an error occurs, -1 will be returned and *errno* is set to indicate the error. See "PosIOctl()" on page 18-8 for more information.

The following example shows how to define a user defined character:

```
#include <pos/pos.h>
#include <pos/display.h>

int dspdes;
int rc;
char * mydsp = "dsp";
PosArg udc[1];
char udc1_char[2];
char udc1_definition[32];

/** Register the application **/
rc = PosInitialize( "myappl",          /* application name */
                  "checkout",        /* resource file name */
                  0,                 /* argc from command line */
                  0,                 /* argv from command line */
                  0,                 /* no resources overridden */
                  0 );               /* number of resources */

/** Open the display **/
dspdes = PosOpen( mydsp,              /* name in resource file */
                 PosDisplay,         /* resource set */
```

```

        0,                /* no resources overridden */
        0 );             /* number of resources */

/** Acquire exclusive use of the display */
rc = PosIOctl( dspdes, POS_SYS_ACQUIRE_DEVICE, 0, 0 );

/** Define the new character */
sprintf( udc1_char, "%c%c", 0x81, 0x41 );
memset( udc1_definition, 0, sizeof(udc1_definition) );

/** Create a pointer to the new character */
udc[0].name = udc1_char;
udc[0].value = (long) udc1_definition;

/** Defining a new character to the display */
rc = PosIOctl( dspdes,
               POS_DSP_DEFINE_CHARACTERS,
               udc,
               sizeof(udc)/sizeof(PosArg) );    /* number of characters */

```

## Reading from Your Device

In order to read data from a point-of-sale device, your application must call the *PosRead()* subroutine. The *PosRead()* subroutine takes as parameters a device descriptor, a pointer to a buffer into which the device data will be read, and the number of bytes to read. The device descriptor must be a valid device descriptor returned by the *PosOpen()* subroutine, and the number of bytes to read should be less than or equal to the size of the buffer into which data will be read.

Upon successful completion, the *PosRead()* subroutine returns the number of bytes placed into the read buffer. If there is no data to be read, a 0 (zero) is returned to your application as the number of bytes read. If an error occurs, -1 is returned and *errno* is set to indicate the error. See “PosRead()” on page 18-16 for more information.

The following example shows how to read data from a device:

```

#include <pos/pos.h>

int msrdes;
int rc;
char buffer[200];
char * mymsr = "msr";

/** Register the application */
rc = PosInitialize( "myappl",    /* application name */
                  "checkout",   /* resource file name */
                  0,            /* argc from command line */
                  0,            /* argv from command line */
                  0,            /* no resources overridden */
                  0 );         /* number of resources */

/** Open the MSR */
msrdes = PosOpen( mymsr,        /* name in resource file */
                 PosMsr,       /* resource set */
                 0,            /* no resources overridden */
                 0 );         /* number of resources */

/** Acquire exclusive use of the MSR */
rc = PosIOctl( msrdes, POS_SYS_ACQUIRE_DEVICE, 0, 0 );

/** Unlock the MSR to allow input */
rc = PosIOctl( msrdes, POS_SYS_UNLOCK_DEVICE, 0, 0 );

```

```

/*
   code to prompt operator to pass a card through, then
   wait on MSR data available event message goes here
*/

/** Read data from the MSR */
rc = PosRead( msrdes,           /* indicate MSR device */
             &buffer[0],      /* buffer address      */
             sizeof(buffer) ); /* buffer length      */

```

## Writing to Your Device

In order to write data to a point-of-sale device, your application must call the *PosWrite()* subroutine. The *PosWrite()* subroutine takes as parameters a device descriptor, a pointer to a buffer containing the data to write, and the number of bytes to write. The device descriptor must be a valid device descriptor opened by the *PosOpen()* subroutine and the number of bytes to write should be less than or equal to the size of the buffer containing the data to be written.

Upon successful completion, the *PosWrite()* subroutine returns the number of bytes processed from the buffer. If an error occurs, -1 is returned and *errno* is set to indicate the error. See “PosWrite()” on page 18-19 for more information.

The following example shows how to write data to a device:

```

#include <pos/pos.h>

int prndes;
int rc;
char * test = "test line";
char * myprn = "prn";

/** Register the application */
rc = PosInitialize( "myappl", /* application name */
                  "checkout", /* resource file name */
                  0,          /* argc from command line */
                  0,          /* argv from command line */
                  0,          /* no resources overridden */
                  0 );       /* number of resources */

/** Open the printer */
prndes = PosOpen( myprn,      /* name in resource file */
                 PosPrinter, /* resource set */
                 0,          /* no resources overridden */
                 0 );       /* number of resources */

/** Acquire exclusive use of the printer */
rc = PosIOctl( prndes, POS_SYS_ACQUIRE_DEVICE, 0, 0 );

/** Write data to the printer */
rc = PosWrite( prndes, /* indicate printer device */
              test,    /* buffer address */
              sizeof(test) ); /* buffer length */

```

## Closing Your Device Connections

When your application no longer requires a device, it should issue a *PosClose()* subroutine call to close the device connection. The *PosClose()* subroutine releases

the device connection if your application has the device acquired. Your application should call the *PosClose()* subroutine for each device connection that has been successfully opened.

The following example shows how to close a device:

```
#include <pos/pos.h>

int msrdes;
int rc;
char * mymsr = "msr";

/** Register the application **/
rc = PosInitialize( "myappl",          /* application name */
                  "checkout",        /* resource file name */
                  0,                  /* argc from command line */
                  0,                  /* argv from command line */
                  0,                  /* no resources overridden */
                  0 );               /* number of resources */

/** Open the MSR **/
msrdes = PosOpen( mymsr,              /* name in resource file */
                 PosMsr,             /* resource set */
                 0,                  /* no resources overridden */
                 0 );               /* number of resources */

.
.
.

/** Close the MSR **/
rc = PosClose( msrdes );
```

## Terminating Your Application

When your application program exits to the operating system, the IBM Point of Sale Subsystem closes any device connections not explicitly closed, and frees any resources your application owns that were created as a result of the *PosInitialize()*, and the *PosOpen()* subroutine calls. For OS/2, if your application terminates abnormally, the same clean-up work takes place. On the Microsoft Windows 3.1 operating system, no such clean-up work can take place. As a result, if your application has any devices acquired at the time of the abnormal termination, the device will remain acquired and your application, when started again, will not be able to use the device. In this situation, it will be necessary to restart the Microsoft Windows operating system.

### OS/2 Exit List Processing

Whenever an application program calls the *PosInitialize()* subroutine, the IBM Point of Sale Subsystem registers an exit list routine for that process. This exit list routine gains control when the process ends, and ensures that all system resources are released. This is especially important when the application program terminates abnormally.

If the application program also registers an exit list routine and the routine calls the IBM Point of Sale Subsystem, then the application exit list routine must run before the IBM Point of Sale Subsystem exit list routines.

The order in which routines run is specified by the high order byte of the low order word of the *ulFunctionOrder* parameter on the *DosExitList()* function call. The IBM Point of Sale Subsystem reserves the values 0x70 to 0x73 for use in calling

*DosExitList()*. The application program should use a value less than 0x70 to ensure that its exit list routine is called before the IBM Point of Sale Subsystem exit list routines.

---

## Using Resources in Your Application

Most devices allow you to affect the way the device behaves by specifying values for resources used by the device. To help in specifying these resource values, the IBM Point of Sale Subsystem provides some macros for you to use in your application program. See “Macros” on page A-1 for more information about the macros that are used with the IBM Point of Sale Subsystem.

## Argument Lists

Argument lists are used to specify device resources. Each entry in a list contains pairs of values of the form (*name*, *value*), as seen in the following example. For the *name* field, a predefined resource name is used. For example, the name field for the **PosNslotNumber** resource is "slotNumber". The *value* field should be cast to long. If the size of the resource stored in the *value* member is less than or equal to the size of long, the value is stored directly in the structure. Otherwise, the *value* member represents a pointer to the resource value. The following example shows the argument list structure.

```
typedef struct
{
    char * name;
    long value;
}
PosArg, *PosArgPtr;
```

### Setting Argument List Values

There are different ways of setting argument values to pass to a device connection. You should use the one that best fits your application needs.

The device connection receives the arguments when it is created by the *PosOpen()* subroutine, or afterwards by means of the *PosSetValues* macro. See “Retrieving and Modifying Resources” on page 5-13 for more information about the *PosSetValues* macro. Arguments can be set by:

- Statically initializing the argument list
- Using the macro *PosSetArg*
- Assigning the value directly

**Statically Initialize Argument List:** The following example shows how to set argument values by statically initializing an argument list.

```
static PosArg args[] =
{
    { PosNslotNumber,    PosSLOT_5 },
    { PosNportNumber,   PosPORT_1 },
    { PosNdeviceNumber, PosDEVICE_SHOPPER_DISPLAY_A }
};
```

The string in the first member of the entries should be specified by using the corresponding macro definition that is the resource name prefixed with PosN. Predefined names are listed in either the device’s header file, or in the `pos/device.h` header file.

When the device connection is created, a pointer to the list and to the number of items in the list can be passed to the device connection on the *PosOpen()* subroutine call. To pass the number of items, use the macro *PosNumber*. The

macro *PosNumber* determines the length of a fixed-size array. Using *PosNumber* allows you to change the size of the argument list easily, and eliminates some errors that are possible when using hard-coded numbers. The following example shows how to use *PosNumber*.

```
PosArg args [5];
PosOpen( "dsp860", PosDisplay, args, PosNumber(args) );
```

**Use the PosSetArg Macro:** The following example shows how to set argument values using the *PosSetArg* macro:

```
PosArg args[10]; // allocate enough space for future arguments
PosSetArg( args[0], PosNslotNumber, PosSLOT_5 );
PosSetArg( args[1], PosNportNumber, PosPORT_1 );
PosSetArg( args[2], PosNdeviceNumber, PosDEVICE_ALPHANUMERIC_DISPLAY_A );
PosOpen( "dsp850", PosDisplay, args, 3 );
```

In the above example, *PosNumber* returns 10 instead of 3, so it cannot be used to specify the number of arguments passed to *PosOpen()*. Use an index that is incremented for each use of *PosSetArg*, but do not increment or decrement it within the first argument of *PosSetArg* because of the way this macro is implemented.

The following example shows how to set argument values in an argument list using an index:

```
PosArg args[10]; // allocate enough space for future arguments
int n = 0;
PosSetArg( args[n], PosNslotNumber, PosSLOT_5 ); n++;
PosSetArg( args[n], PosNportNumber, PosPORT_1 ); n++;
PosSetArg( args[n], PosNdeviceNumber, PosDEVICE_OPERATOR_DISPLAY_A ); n++;
PosOpen( "dsp863", PosDisplay, args, n );
```

**Assign Value Directly:** The following example shows how to fill in or modify argument values by assigning values directly.

```
PosArg args[10]; // allocate enough space for future arguments
args[0].name = PosNslotNumber;
args[0].value = (long) 7;
```

## Retrieving and Modifying Resources

Resources can be retrieved and modified by means of the macros *PosGetValues* and *PosSetValues*. The application is responsible for allocating space into which the resource value is copied when the *value* field of the argument contains a pointer.

The following example does not provide an argument list when the device is opened. In this case, *PosOpen()* opens the device using the device default values of and the resources for the device specified in the resource file from the *PosInitialize()* subroutine. After the device has been opened, use the *PosSetValues* macro to alter only those device's resources with an "S" (settable) access code.

```
PosArg args[10];
int n = 0;

int device = PosOpen( "myscanner", PosScanner, 0, 0 );

PosSetArg( args[n], PosNbeepState, PosENABLE ); n++;
PosSetValues( device, args, n );
```

You can retrieve the current value of device resources with a "G" (gettable) access code by using the macro *PosGetValues*. *PosGetValues* retrieves the named resources from the specified device and copies the data into the given address. See Chapter 21. Resource Sets for more information about access codes.

The following example illustrates how to retrieve the slot number, the port number, and the device number of a device connection:

```
PosArg args[10];
int slot, port, device;
int n = 0;

PosSetArg( args[n], PosNslotNumber, -1 ); n++;
PosSetArg( args[n], PosNportNumber, -1 ); n++;
PosSetArg( args[n], PosNdeviceNumber, -1 ); n++;
PosGetValues( device, args, n );
```

When *PosGetValues* returns, *args[0].value* contains a copy of the device's **PosNslotNumber** resource value, *args[1].value* contains a copy of the device's **PosNportNumber** resource value, and *args[2].value* contains a copy of the device's **PosNdeviceNumber** resource value. Declaring variables as the wrong data type when retrieving resources can result in errors, because the IBM Point of Sale Subsystem copies the data bitwise into the provided address.

---

## Building your Application

The IBM Point of Sale Subsystem for OS/2 supports the following compilers:

- IBM VisualAge C++ for OS/2
- Borland™ C/C++ for OS/2

The IBM Point of Sale Subsystem for Windows supports the following compilers:

- Microsoft Visual C++ Version 1.5 or later (16-bit applications)
- Microsoft Visual C++ Version 2.0 or later (32-bit applications)
- Borland C++ Version 4.5 or later (16-bit applications)
- Borland C++ Version 4.5 or later (32-bit applications)
- IBM VisualAge C++ Version 3.5 or later (32-bit applications)

The IBM Point of Sale Subsystem for Linux supports the GNU Compiler Collection (GCC).

For information on compiling and linking your application, refer to your compiler's documentation. All supported compilers have an Integrated Development Environment (IDE) in addition to the traditional command line interface.

On your application development system, you will need to have the IBM Point of Sale Subsystem header files and libraries.

---

## C Language Header Files

The IBM Point of Sale Subsystem provides the C language header files required to compile your program using the application programming interface (API). See Chapter 18. Application Programming Interface for more information about application programming interfaces. The default location of the header files is in the directory C:\POS\INCLUDE\POS. The header files in this directory are:

alarm.h	nvram.h
device.h	pos.h
display.h	power.h
drawer.h	printer.h
errno.h	rs232c.h
helper.h	scale.h
keyboard.h	scanner.h

msr.h

touch.h

---

## IBM Point of Sale Subsystem Libraries

The IBM Point of Sale Subsystem provides compiler-specific libraries necessary to link your application for your target operating system. For 16-bit applications, be sure to specify the same memory model for compilation and linking.

The following libraries are to be used for linking a IBM Point of Sale Subsystem application for OS/2:

<b>Library Name</b>	<b>Description</b>
<b>AIPAPI.LIB</b>	IBM VisualAge C++ for OS/2
<b>AIPAPI.LIB</b>	Borland C++ for OS/2

Use the following libraries to link a 16-bit IBM Point of Sale Subsystem application for Microsoft Windows:

<b>Library Name</b>	<b>Description</b>
<b>AIPW16MC.LIB</b>	Microsoft Visual C++ (16-bit), compact memory model
<b>AIPW16MS.LIB</b>	Microsoft Visual C++ (16-bit), small memory model
<b>AIPW16MM.LIB</b>	Microsoft Visual C++ (16-bit), medium memory model
<b>AIPW16ML.LIB</b>	Microsoft Visual C++ (16-bit), large memory model
<b>AIPW16MH.LIB</b>	Microsoft Visual C++ (16-bit), huge memory model
<b>AIPW16BC.LIB</b>	Borland C++ (16-bit), compact memory model
<b>AIPW16BS.LIB</b>	Borland C++ (16-bit), small memory model
<b>AIPW16BM.LIB</b>	Borland C++ (16-bit), medium memory model
<b>AIPW16BL.LIB</b>	Borland C++ (16-bit), large memory model

Use the following libraries to link a 32-bit IBM Point of Sale Subsystem application for Microsoft Windows:

<b>Library Name</b>	<b>Description</b>
<b>AIPW32MD.LIB</b>	Microsoft Visual C++ (32-bit), dynamic run-time model
<b>AIPW32MS.LIB</b>	Microsoft Visual C++ (32-bit), single-threaded library
<b>AIPW32MM.LIB</b>	Microsoft Visual C++ (32-bit), multi-threaded library
<b>AIPW32BS.LIB</b>	Borland C++ (32-bit), compact memory model
<b>AIPW16BM.LIB</b>	Borland C++ (32-bit), small memory model
<b>AIPWIMED.LIB</b>	Multi-threaded, dynamically-linked library
<b>AIPWIMES.LIB</b>	Multi-threaded, statically-linked library
<b>AIPWISED.LIB</b>	Single-threaded, dynamically-linked library
<b>AIPWISES.LIB</b>	Single-threaded, statically-linked library

Use the following libraries to link a 32-bit IBM Point of Sale Subsystem dynamic link library (DLL) for Microsoft Windows:

<b>Library Name</b>	<b>Description</b>
<b>AIPWIMDD.LIB</b>	IBM VisualAge C++ for Windows, multi-threaded, dynamically linked library
<b>AIPWIMDS.LIB</b>	IBM VisualAge C++ for Windows, multi-threaded, statically linked library
<b>AIPWISDD.LIB</b>	IBM VisualAge C++ for Windows, single-threaded, dynamically linked library
<b>AIPWISDS.LIB</b>	IBM VisualAge C++ for Windows, single-threaded, statically linked library

Use the following library when linking an IBM Point of Sale Subsystem for Linux application:

Library Name	Description
libaiplib.so	GNU Compiler Collection

---

## Optimizing Application Performance

Before you write applications, read this section to learn how to optimize the rate at which the applications function in conjunction with the IBM Point of Sale Subsystem. Some performance tuning can be done with little modification to the application, but some must be designed into the program. The following sections discuss several ways to optimize the performance of an application that is using the IBM Point of Sale Subsystem.

### Application Priority (OS/2 Only)

When dynamic OS/2 system priority variation is enabled, the process that currently has the focus receives a higher priority. (Dynamic priority variation is the default.) The priority it receives is between regular, level 31, and fixed-high, level 0. Any thread running at fixed-high priority has an advantage over threads running at regular priority or below.

Dynamic priority variation allows the operating system to adjust the base priority of a thread, based on factors such as system load and process activity. Allowing the operating system to raise the priority of the process that currently has the focus, prevents process starvation. Dynamic priority variation is the default, but can be explicitly specified by indicating `PRIORITY=DYNAMIC` in the `CONFIG.SYS` file.

You can set the operating system to run with absolute priority (`PRIORITY=ABSOLUTE`). However, under absolute priority, starving processes and processes that have the focus never receive higher priorities. Dynamic priority variation produces the best operating system performance under most conditions.

If you want your application to have processor access at a higher priority than other applications that might currently have the focus, and if those applications have a regular priority class, use a priority class of fixed-high, level 0. Threads that have a priority class of fixed-high have a lower priority than time-critical threads, but a higher priority than threads that have the default priority class (regular, level 0). In addition, when an application thread issues an IBM Point of Sale Subsystem subroutine call, it receives a temporary higher priority of fixed-high, level 31.

**Advantages:** An application that is running at a fixed-high priority has a processor priority well above the default priority of most applications. You should not set the application to time-critical priority because time-critical priority lowers the efficiency of the system.

**Disadvantages:** If there is more than one application set to fixed-high priority, the applications will contend against each other for processor time.

### Presentation Manager Considerations (OS/2 Only)

If your application is a Presentation Manager application, use a Presentation Manager message queue instead of the default IBM Point of Sale Subsystem input queue (device descriptor zero) for receiving IBM Point of Sale Subsystem event messages. Specify the Presentation Manager queue handle using the

**PosNqueueHandle** resource on the *PosInitialize()* subroutine call, and the **PosNqueueHandle** resource on the *PosOpen()* subroutine call.

For Presentation Manager programs, the Presentation Manager call to read the queue waits until a message appears on the queue before returning to the application, so the application never needs to poll the input queue.

## Microsoft Windows Considerations

If your application is a Microsoft Windows application, use the Microsoft Windows message queue instead of the default IBM Point of Sale Subsystem input queue (device descriptor zero) for receiving IBM Point of Sale Subsystem event messages. Specify the appropriate Microsoft Windows window handle using the **PosNqueueHandle** resource on the *PosInitialize()* subroutine call and the **PosNqueueHandle** resource on the *PosOpen()* subroutine call.

For Microsoft Windows programs, the Microsoft Windows call to read the queue waits until a message appears on the queue before returning to the application, so the application never needs to poll the input queue.

## Polling Considerations

By default, the IBM Point of Sale Subsystem application programming interface does not wait for an event message to appear on the default IBM Point of Sale Subsystem input queue. Applications must continually read (or poll) the IBM Point of Sale Subsystem input queue to receive notification of an event message. Polling uses a great deal of processor time.

For non-Presentation Manager OS/2 applications, if you want your application to wait for event messages, you should set the **PosNreadTimeout** resource to the number of milliseconds your application should wait for an event message. A value of -1 indicates the application wants to wait indefinitely for an event message and a value of 0 (zero) indicates the application does not want to wait for an event message.

**If you must poll:** If your application is not an OS/2 Presentation Manager application or a Microsoft Windows application and your application uses the **PosNreadTimeout** resource value of 0 (zero), your application must poll the IBM Point of Sale Subsystem input queue to receive input on a timely basis. While polling does use a great deal of processor time, there are ways to make polling more efficient.

**Note:** The **PosNreadTimeout** resource is ignored for 16-bit applications for the Microsoft Windows operating system and for 32-bit applications using Microsoft Win32s on Microsoft Windows 3.1.

The following example shows one way to poll the IBM Point of Sale Subsystem input queue. Keep the following notes in mind for the example:

### Notes:

1. The example sleeps for a short amount of time rather than continually calling the *PosRead()* subroutine. This allows other processes to have access to the processor while this process is waiting.
2. The example only sleeps if there is no event message in the IBM Point of Sale Subsystem input queue at the time of the *PosRead()* subroutine call. If there is more than one event message in the input queue, there will not be an intervening *DosSleep()* call between the two reads.

3. This example is similar to Presentation Manager programs. After initialization, the main purpose of the example is to wait for event messages and then act upon those event messages.
4. If your application also responds to input from the system keyboard, user input must be looked for as well. (The AIPTSTR sample code has an example of how to do this.)
5. Error checking was omitted in the example code for brevity.
6. Polling is not recommended, and should be avoided if possible.

```

.
.
.
int Done = 0;
int BytesRead;
POSQMSG qmsg;

while (! Done)
{
    while (0 == (BytesRead = PosRead(0, &qmsg, sizeof(qmsg))))
    {
        DosSleep(50);          /* Sleep for 50 milliseconds */
    }

    switch(qmsg.msg)
    {
        /* Process the event message here */
        .
        .
        .
    }
}
.
.
.

```

## Multi-threaded Application Design

Multi-threaded applications have many useful aspects in programming, including improving the apparent performance and simplifying logic design. However, using multiple threads when they are not needed can cause inefficiencies.

No more than one IBM Point of Sale Subsystem application programming interface subroutine can be active at the same time. When an IBM Point of Sale Subsystem application programming interface subroutine is called by one thread in a process, IBM Point of Sale Subsystem application programming interface calls by any other thread in the same process are blocked (prevented from processing) until the first call is complete. When the first call completes, one of the waiting calls is processed.

The only exception to this is when the *PosRead()* subroutine is called for the IBM Point of Sale Subsystem input queue (device descriptor zero). The **PosnreadTimeout** resource allows a read operation on the IBM Point of Sale Subsystem input queue to be suspended until data is available to be read. When one thread is waiting for data on the input queue, other threads can successfully call IBM Point of Sale Subsystem application programming interface subroutines for other device descriptors and not be blocked. Even with this exception, only one of the other IBM Point of Sale Subsystem application programming interface calls can be active simultaneously.

A good example of a multi-threaded application design is one thread controlling one set of input/output devices, such as a base point-of-sale terminal and the devices

attached to it, and a second thread controlling another set of input/output devices, such as a satellite point-of-sale terminal and the devices attached to it.

An example of an unnecessary multi-threaded design is one thread to control each device. Having one thread control each device does not improve performance, and can be an unnecessary complication to program logic flow.

## **Improving the Maintainability of Your Application**

This section contains some suggestions for improving the maintainability of your application.

1. Use the `#define` constants provided in the IBM Point of Sale Subsystem header files instead of hard-coding values.
2. Use the `sizeof()` and the `PosNumber()` macros when passing array information to the IBM Point of Sale Subsystem. This ensures that the values are correct if array or buffer sizes change.
3. The *resource names* and *values* in the resource file must be spelled correctly, and must have the correct case (upper or lower) to be recognized.
4. For OS/2, all applications must be compiled as multi-threaded applications.
5. For OS/2, if possible, use the version of the runtime library that is statically linked. This increases the size of the application, but reduces load time in most cases.
6. Allow the compiler to perform the program linking. The compiler sets many switches that are necessary for trouble-free linking. If you want to call the linker explicitly, be sure that you use the same switches that the compiler does.
7. Set the compiler to a high warning level. This ensures that potential errors, like calling non-prototyped functions are caught early.



---

## Chapter 6. Alarm Programming

The alarm device handler is software that manages communication between your application and the alarm. An application can access the device only through the device handler.

---

### Characteristics of the Alarm

- Each point-of-sale terminal supports one alarm. An alarm can only be connected to socket 3B of the point-of-sale terminal.
- Socket 3B output can be set high or low to activate an attached alarm.

#### Notes:

1. The alarm device is not supported on any model of the IBM 4695.
2. The alarm device is not supported on the IBM Point of Sale Subsystem for Linux.

A cash drawer, instead of an alarm, can be connected to socket 3B. The IBM Point of Sale Subsystem cannot distinguish whether the attached device is an alarm or a cash drawer. The alarm device handler, upon detecting a device in socket 3B, notifies your application that both a cash drawer and an alarm are available. If the application can support both a cash drawer and an alarm, the application must allow the user to specify which is attached.

---

### Functions Your Application Performs

An application program can perform the following functions with the alarm:

- Sound an alarm
- Silence an alarm
- Get alarm status

Before your application program can access the alarm, it must open the alarm (see "Opening Your Device" on page 5-4) and acquire exclusive use of the alarm.

### Sounding an Alarm

To sound an alarm, issue a *PosIOctl()* subroutine call and specify `POS_ALARM_SOUND_ALARM` for the *request* parameter. If the alarm is not acquired, the request fails with the error code 315 `POSERR_SYS_NOT_ACQUIRED`. If the alarm is not online, the request fails with the error code 317 `POSERR_SYS_DEVICE_OFFLINE`.

The alarm continues to sound until you silence it. If you sound an alarm that is already sounding, no error is returned.

### Silencing an Alarm

To silence an alarm, issue a *PosIOctl()* subroutine call and specify `POS_ALARM_SILENCE_ALARM` for the *request* parameter. This request is only accepted by the acquired alarm connection, and it does not matter if the alarm is online at the time. If the alarm is not acquired, the request fails with the error code 315 `POSERR_SYS_NOT_ACQUIRED`.

If you silence an alarm that is not sounding, no error is returned.

The alarm can also be silenced implicitly by issuing one of the following:

- *PosIOctl()* subroutine call, specifying `POS_SYS_RELEASE_DEVICE` for the *request* parameter. Re-acquiring the device does not automatically sound it again. You must explicitly sound the alarm if you want it back on.
- *PosClose()* subroutine call, if your application has the alarm acquired at the time of this call.

## Getting Alarm Status

The status of the alarm can be determined by calling the *PosIOctl()* subroutine with the `POS_SYS_GET_VALUES` request for the **PosNalarmStatus** resource. You can get the alarm status whether the device is acquired or not. This request returns the current state of the alarm, whether it is sounding or silent, and whether it is presently online or offline. See “PosNalarmStatus” on page 21-16 for more information.

If the device is reported as being offline, it is also reported as being silent.

---

## Related Information

Additional information about alarm programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOctl() Requests
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with Alarm

See Chapter 18. Application Programming Interface:

`PosClose()`  
`PosIOctl()`  
`PosOpen()`

## Alarm PosIOctl() Control Requests

See Chapter 19. PosIOctl() Requests:

`POS_ALARM_SILENCE_ALARM`  
`POS_ALARM_SOUND_ALARM`  
`POS_SYS_ACQUIRE_DEVICE`  
`POS_SYS_GET_VALUES`  
`POS_SYS_RELEASE_DEVICE`

## Alarm Resources

See Chapter 21. Resource Sets:

`PosNalarmStatus`

## Alarm Error Codes

There are no error codes that are specific to the alarm.

---

## Chapter 7. Cash Drawer Programming

The cash drawer device handler is software that manages communication between your application and the cash drawer. An application can access the device only through the device handler.

---

### Characteristics of the Cash Drawer

The cash drawer has the following characteristics:

- Each point-of-sale terminal supports a maximum of two cash drawers. Cash drawers can be connected to socket 3A or 3B of the point-of-sale terminal.
- To open a cash drawer till, your application must send a pulse of a minimum duration to the cash drawer. If you use an IBM cash drawer, the pulse time is 80 milliseconds. If you use a non-IBM cash drawer, other pulse times might be required. The pulse width is specified by the **PosNpulseWidth** resource. See “PosNpulseWidth” on page 21-21 for more information.

An alarm, instead of a cash drawer, can be connected to socket 3B. The IBM Point of Sale Subsystem cannot distinguish whether the attached device is a cash drawer or an alarm. The cash drawer device handler, upon detecting a device in socket 3B, notifies your application that both a cash drawer and an alarm are available. If the application can support both an alarm and a cash drawer, the application must allow the user to specify which is attached.

---

### Functions Your Application Performs

An application program can perform the following functions with the cash drawer:

- Open a cash drawer till
- Get cash drawer status
- Set cash drawer pulse width

**Note:** Before your application program can access the cash drawer, it must open the cash drawer device and acquire exclusive use of it.

### Opening a Cash Drawer Till

To open a cash drawer till, issue a *PosIOctl()* subroutine call and specify `POS_TILL_OPEN_TILL` for the *request* parameter. If the cash drawer is not acquired, the request fails with the error code 315 `POSERR_SYS_NOT_ACQUIRED`. If the cash drawer is not on-line, the request fails with the error code 317 `POSERR_SYS_DEVICE_OFFLINE`. Opening a cash drawer till that is already open returns no error. Opening a cash drawer till that is locked with a manager key returns no error.

When a till is detected as being physically opened, the application with the acquired device connection receives the `POSM_TILL_OPENED` event message. Usually, this event message is the response to your application requesting the cash drawer till to be opened. If not, it probably means that the manager’s key was used to open the cash drawer till manually. This event message is sent to your application through the input queue associated with the acquired device connection. This allows your application to notify the operator that it detected the action and how to proceed. See “`POSM_TILL_OPENED`” on page 20-26 for more information.

## Getting Cash Drawer Status

The status of the cash drawer can be determined by calling the *PosIOCtl()* subroutine with the `POS_SYS_GET_VALUES` request for the **PosNtillStatus** resource. You can get the cash drawer status whether the cash drawer is acquired or not. This request returns the current state of the cash drawer till, whether it is open or closed. In addition, the state of whether the device is presently on-line or off-line is returned. See “PosNtillStatus” on page 21-22 for more information.

When you issue an open cash drawer till request, give the cash drawer till time to physically open before requesting status, or wait for the `POSM_TILL_OPENED` event message.

When the cash drawer till is detected as being closed, the application with the acquired cash drawer connection receives the `POSM_TILL_CLOSED` event message, indicating that the cash drawer till is closed. This condition is also reflected in the **PosNtillStatus** resource.

## Setting Cash Drawer Pulse Width

The cash drawer pulse width can be set or queried by calling the *PosIOCtl()* subroutine with the `POS_SYS_SET_VALUES` request or the `POS_SYS_GET_VALUES` request for the **PosNpulseWidth** resource. See “PosNpulseWidth” on page 21-21 for more information.

**Note:** The default value for **PosNpulseWidth** is normally long enough that your application should rarely, if ever, find it necessary to set the cash drawer’s pulse width. If you do need to use a value other than the default, specify the value in a resource file to enable all applications to use the same value. This prevents the possible situation of each application having a different value for the pulse width, causing a possibly incorrect value to be used to open the cash drawer.

---

## Related Information

Additional information about cash drawer programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOCtl() Requests
- Chapter 20. Event Messages
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with Cash Drawer

See Chapter 18. Application Programming Interface:

PosClose()  
PosIOCtl()  
PosOpen()

## Cash Drawer PosIOCtl() Control Requests

See Chapter 19. PosIOCtl() Requests:

POS\_SYS\_ACQUIRE\_DEVICE  
POS\_SYS\_GET\_VALUES  
POS\_SYS\_RELEASE\_DEVICE  
POS\_SYS\_SET\_VALUES  
POS\_TILL\_OPEN\_TILL

## **Cash Drawer Event Messages**

See Chapter 20. Event Messages:  
POSM\_TILL\_CLOSED  
POSM\_TILL\_OPENED

## **Cash Drawer Resources**

See Chapter 21. Resource Sets:  
PosNpulseWidth  
PosNtillStatus

## **Cash Drawer Error Codes**

There are no error codes that are specific to the cash drawer.



---

## Chapter 8. Display Programming

The display device handler is software that manages communication between your application and the display. An application can access the device only through the device handler.

---

### Characteristics of the Displays

The IBM Point of Sale Subsystem supports the following RS-485 point-of-sale displays:

- Alphanumeric Display
- Operator Display
- Shopper Display
- Character and Graphics Display
- 50-Key Modifiable Layout Keyboard/Operator Display
- Retail Point of Sale Keyboard with Card Reader and Display
- PLU Keyboard/Display-III
- 40-Character Liquid Crystal Display (LCD)
- 40-Character Vacuum Fluorescent Display II (VFD II)
- Two-Sided Vacuum Fluorescent Display II (Two-Sided VFD II)
- 2x20 Character Vacuum Fluorescent (Customer) Display (VFD)

Some of these displays can be connected directly to the point-of-sale terminal, some of these displays are integrated into keyboards. Displays integrated into a keyboard are attached when the keyboard is connected to socket 5A or 5B of the point-of-sale terminal. The IBM Point of Sale Subsystem supports the following USB point-of-sale displays:

- 40-Character Vacuum Fluorescent Display
- 40-Character Liquid Crystal Display

#### Notes:

1. Connecting an Operator Display into socket 4A while a 50-Key Modifiable Layout Keyboard/Operator Display or Retail Point of Sale Keyboard with Card Reader and Display is connected to socket 5A will produce unpredictable results.
2. Connecting an Operator Display into socket 4B while a 50-Key Modifiable Layout Keyboard/Operator Display or Retail Point of Sale Keyboard with Card Reader and Display is connected to socket 5B will produce unpredictable results.

### Alphanumeric Display

- The display consists of two rows of 20 alphanumeric character positions numbered 0-39, where 0-19 are the twenty characters on the top row, and 20-39 are the twenty characters on the bottom row.
- Each character position contains a 5-dots-wide by 12-dots-high matrix (5x12), that is used to form the character.
- Code pages are supplied for the display that define the matrix definition for each character in the code page.
- The display has a blank space between character positions and between the two rows. This should be considered if you attempt to use multiple character positions to produce a graphic display.

**Note:** The Alphanumeric Display is not supported on the IBM Point of Sale Subsystem for Linux.

## Operator Display

- The Operator Display consists of two rows of 20 alphanumeric character positions numbered 0-39, where 0-19 are the twenty characters on the top row, and 20-39 are the twenty characters on the bottom row.
- Each character position contains a 5-dots-wide by 8-dots-high matrix (5x8), that is used to form the character.
- The dot matrix pattern for each character is defined within the Operator Display electronics and cannot be modified.
- Code pages are supplied for the Operator Display that translate an application's character code point to the appropriate character in the Operator Display character set.

**Note:** The Operator Display can be housed in the 50-Key Modifiable Layout Keyboard and Operator Display or the Retail Point of Sale Keyboard with Card Reader and Display.

## Shopper Display

- The Shopper Display consists of 12 positions, numbered 0-11, composed of 8 alphanumeric positions and 4 comma/decimal positions, plus 6 discrete guidance lights.
- Each alphanumeric position contains 7 bar-segments used to form the character. The bar-segment pattern for each character is defined within the Shopper Display electronics and cannot be modified.  
Positions 0, 1, 3, 5, 6, 8, 10 and 11 can display the alphanumeric characters supported by the device. Positions 4, 7 and 9 can display commas and periods. Position 2 can only display a comma.

### Notes:

1. Attempting to display a character that is not valid for a given position results in that position being blank. The supported character set consists of the numerals plus a limited number of alphabetic and special characters. See the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book for more information.
2. The Shopper Display is not supported on IBM Point of Sale Subsystem for Linux.

## Character and Graphics Display

- The Character and Graphics Display consists of a 160x36 pixel grid that can be subdivided into two rows of 20 characters or three rows of 32 characters plus two rows of 12 indicator lights, one above the pixel grid and one below.
- This display supports 8x16 and 5x12 pixel characters.
- Code pages are supplied for the Character and Graphics Display that define the matrix definition for each character in the code page.
- Using 8x16 characters, the top 32 rows of pixels are used to display the ASCII box characters correctly. In double-byte mode, the four rows of pixels in the middle of the display are blank so that the single-byte characters line up properly with the double-byte characters.

**Notes:**

1. The Character and Graphics Display can be housed in the PLU Keyboard and Display-III.
2. The Character and Graphics Display is not supported on the IBM Point of Sale Subsystem for Linux.

## 40-Character Liquid Crystal Display

- The display consists of two rows of 20 alphanumeric character positions numbered 0-39, where 0-19 are the twenty characters on the top row, and 20-39 are the twenty characters on the bottom row.
- Each character position contains a 5-dots-wide by 8-dots-high matrix (5x8), that is used to form the character.
- The dot matrix pattern for each character is defined within the LCD electronics and cannot be modified.
- Code pages are supplied for the LCD which translate an application's character code point to the appropriate character in the LCD character set.
- The display has a blank space between character positions and between the two rows. This should be considered if you attempt to use multiple character positions to produce a graphic display.

## 40-Character Vacuum Fluorescent Display II and 2x20 Character VFD Customer Display

- The display consists of two rows of 20 alphanumeric character positions numbered 0-39, where 0-19 are the twenty characters on the top row, and 20-39 are the twenty characters on the bottom row.
- Each character position contains a 5-dots-wide by 8-dots-high matrix (5x8), that is used to form the character.
- The dot matrix pattern for each character is defined within the VFD and VFD II electronics and cannot be modified.
- Older VFD II models have two character sets stored in ROM while newer models store eleven character sets in ROM.
- Code pages are supplied for the VFD that translate an application's character code point to the appropriate character in the VFD character set.
- The display has a blank space between character positions and between the two rows. This should be considered if you attempt to use multiple character positions to produce a graphic display.

## Two-Sided Vacuum Fluorescent Display II

- The display consists of two VFD II displays under the same display cover.
- The characteristics of each display is the same as the 40-Character Vacuum Fluorescent Display II and 2x20 Character VFD Customer Display.

---

## Functions Your Application Performs

Your application can perform the following functions on the display:

- Use different code pages
- Write characters to the display
- Write bitmaps to any position on the display
- Set guidance lights on the display
- Clear the display
- Define user-defined characters

Before your application program can access the display, it must open the display (see “Opening Your Device” on page 5-4) and acquire exclusive use of the display. When the display is opened, a buffer is created to contain the data that the application writes to the display. Whenever your application acquires the display, the contents of this buffer are written to the display.

## Code Page Support

The code page of the process at the time the *PosOpen()* subroutine call is issued is queried and saved by the IBM Point of Sale Subsystem.

### Notes:

1. Code pages are defined for the Operator Display, the LCD, the VFD, the VFD II, and the Two-Sided VFD II but due to the fixed internal character set, not all characters are available in each code page. Code pages are not defined at all for the Shopper Display, due to its specialized type of function. See the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book for a list of the character code points that are supported.
2. For the VFD, the VFD II, and the Two-Sided VFD II, which have only two resident code pages, the defined code pages are used to translate the application character code points to the appropriate display character when the application is using a single-byte code page. When the application is using a double-byte code page, the application character code points are not translated.
3. The Character and Graphics Display uses the code page support described above for single-byte characters. The double-byte characters are located in ROM in the device. For the USB Character/Graphics Displays, which have eleven resident code pages, the application character code page points are not translated.
4. The IBM Point of Sale Subsystem uses code page 850 for any OS/2 code page not listed in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

## Writing Characters to the Display

Before writing characters to the display, you must set the current character position to the location where you want to start writing, assuming that the character position is not already where you want it. Set the starting row and column of where you want to start writing characters by setting the **PosNdisplayCursor** resource. See “PosNdisplayCursor” on page 21-19 for more information.

After setting the current row and column position, use the *PosWrite()* subroutine to display the characters. If the number of characters written exceeds the number of positions on the current row, the display device handler writes the remaining characters on the next row. If the last row of the display is exceeded, the display device handler stops writing characters and returns the number of characters actually written. The current character position, **PosNdisplayCursor**, is set to the position following the last character written, or to the last position of the display if the end of the display was reached.

You can set the **PosNdisplayCursor** resource without first having to acquire the display. However, you cannot write to the display unless it has been successfully acquired and then only if the display is online (connected). If the device is not acquired, the write fails with the error code `POSERR_SYS_NOT_ACQUIRED`. If the display is offline, the write fails with the error code `POSERR_SYS_DEVICE_OFFLINE`.

Even if you receive the error code `POSERR_SYS_DEVICE_OFFLINE`, your information has been successfully written to the display buffer. When the display comes back online, the display is refreshed with the latest information written to the display.

In most cases, the characters displayed are from the code page associated with the display device connection. A code page is associated with a display device connection at the time the device is opened. For newer VFD II devices and USB Character/Graphics devices, the device character set can be changed by setting the **PosNdisplayCodePage** resource. (See “PosNdisplayCodePage” on page 21-17 for more information.) This applies to those VFD II devices that support the **PosNdisplayCodePage** resource.

## Writing Bitmaps to the Display

This function is specific to the Character and Graphics Display only. Before writing bitmaps to the display, you must set **PosNdisplayMode** to `PosDSP_MODE_LOGO`. You must also set the values of **PosNpixelX** and **PosNpixelY** to the location of the upper-left-hand corner of the bitmap.

After setting the bitmap location, use the *PosWrite()* subroutine to display the bitmap. If the bitmap is too large to fit on the display at the location specified, the visible portion of the bitmap will be displayed and the number of bytes passed to the subroutine will be returned.

You cannot write to the display unless it is successfully acquired and then only if the display is online (connected). If the display is not acquired, the write fails with the error code 315 `POSERR_SYS_NOT_ACQUIRED`. If the display is offline, the write fails with the error 317 `POSERR_SYS_DEVICE_OFFLINE`.

Even if you receive the error code 317 POSERR\_SYS\_DEVICE\_OFFLINE, your information has been successfully written to the display buffer. When the display comes back online, the display is refreshed with the latest information written to the display.

The format of the bitmap data is illustrated in the following table.

*Table 8-1. Bitmap Data Format. Defines a DBCS character (32 bytes); only SBCS requires 8 bytes*

Field	Size	Values
type	unsigned long	1
width	unsigned long	1–160
height	unsigned long	1–36
bits	$(( \text{width} + 7 ) / 8 ) * \text{height} \text{ bytes}$	

**Note:** Currently, the only valid value for the type field is 1.

## Setting the Guidance Lights

This function is specific to the following displays:

- Shopper Display
- Character and Graphics Display
- 40-Character Vacuum Fluorescent Display II
- 40-Character VFD Customer Display
- Two-Sided Vacuum Fluorescent Display II

You can set the guidance lights by specifying the value for the resource **PosNdisplayLightsOn** in the resource file or by sending it on the POS\_SYS\_SET\_VALUES *PosIOctl()* request.

## Clearing the Display Screen

Use the POS\_DSP\_CLEAR\_SCREEN *PosIOctl()* request to clear all the characters from a display.

**Note:** Clearing the display screen does not clear the indicator lights.

See “POS\_DSP\_CLEAR\_SCREEN” on page 19-5 for more information about this request.

## User-Defined Characters

The Character and Graphics Display supports the definition of new characters by your application program. The following diagram shows the layout for defining a new user-defined character.

In the table, the first column indicates which display dot position will be turned on when the corresponding bit is set. Each numbered column defines the 16 vertical dot positions for one column of the character. Two bytes are needed to define the 16 display dot positions for each column. See “Defining User-Defined Characters” on page 5-8 for more information about defining user-defined characters.

**Note:** The first byte of the two-byte column definition corresponds to the lower 8 display dot positions. The second byte defines the upper 8 display dot positions.

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
Position 1 (Bit 0)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 2 (Bit 1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 3 (Bit 2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 4 (Bit 3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 5 (Bit 4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 6 (Bit 5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 7 (Bit 6)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 8 (Bit 7)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Byte	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
Position 9 (Bit 0)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 10 (Bit 1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 11 (Bit 2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 12 (Bit 3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 13 (Bit 4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 14 (Bit 5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 15 (Bit 6)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Position 16 (Bit 7)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Byte	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31

---

## Related Information

Additional information about display programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOCtl() Requests
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with Displays

See Chapter 18. Application Programming Interface:

- PosClose()
- PosIOctl()
- PosOpen()
- PosWrite()

## Display PosIOctl() Control Requests

See Chapter 19. PosIOctl() Requests:

- POS\_DSP\_CLEAR\_SCREEN
- POS\_DSP\_DEFINE\_CHARACTERS
- POS\_SYS\_ACQUIRE\_DEVICE
- POS\_SYS\_GET\_VALUES
- POS\_SYS\_RELEASE\_DEVICE
- POS\_SYS\_SET\_VALUES

## Display Resources

See Chapter 21. Resource Sets:

- PosNcharSize
- PosNdisplayCodePage
- PosNdisplayCursor
- PosNdisplayMode
- PosNdisplayLightsOn
- PosNpixelX
- PosNpixelY

## Display Error Codes

See Appendix D. Error Codes:

- 4405 POSERR\_DSP\_BAD\_BITMAP
- 4406 POSERR\_DSP\_INVALID\_CODE\_PAGE
- 4401 POSERR\_DSP\_INVALID\_CURSOR
- 4402 POSERR\_DSP\_INVALID\_MODE
- 4403 POSERR\_DSP\_INVALID\_SIZE
- 4404 POSERR\_DSP\_UNSUPPORTED\_BITMAP

---

## Chapter 9. Keyboard Programming

The keyboard device handler is software that manages communication between your application and the keyboard. An application can only access the device through the device handler.

**Note:** Use this chapter in conjunction with the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

---

### Characteristics of the Keyboards

The IBM Point of Sale Subsystem supports the following point-of-sale keyboards:

- 50-Key Modifiable Layout Keyboard<sup>1</sup>
- 50-Key Modifiable Layout Keyboard and Operator Display<sup>1</sup>
- Alphanumeric Point of Sale (ANPOS) Keyboard<sup>1</sup>
- Modifiable Layout Keyboard with Card Reader (133-key)<sup>1</sup>
- Retail Alphanumeric Point of Sale Keyboard with Card Reader
- Retail Point of Sale Keyboards
  - Retail Point of Sale Keyboard (50-key)
  - Retail Point of Sale Keyboard with Card Reader (50-key)
  - Retail Point of Sale Keyboard with Card Reader and Display (50-key)
  - Point of Sale Keyboard VI<sup>1</sup>
- USB Keyboards<sup>1</sup>:
  - USB Retail Point of Sale Keyboard
  - USB Retail Point of Sale Keyboard with Card Reader
  - USB Retail Point of Sale Keyboard with Card Reader and Display
  - USB Retail Alphanumeric Point of Sale with Card Reader
  - USB Modifiable Layout Keyboard with Card Reader
- PC Point of Sale Keyboard (ANKPOS) Keyboards<sup>1</sup>
- Point of Sale Keyboard V<sup>1</sup>
- PLU Keyboard/Display-III<sup>1</sup>
- 4685 Point of Sale Keyboard Model K01<sup>1</sup>
- IBM 4820 SurePoint Solution Keypad<sup>2</sup>

The following keyboards are referred to as the *checkout keyboards*:

- 50-Key Modifiable Layout Keyboard<sup>1</sup>
- 50-Key Modifiable Layout Keyboard and Operator Display<sup>1</sup>
- Retail Point of Sale Keyboards
  - Retail Point of Sale Keyboard (50-key)
  - Retail Point of Sale Keyboard with Card Reader (50-key)
  - Retail Point of Sale Keyboard with Card Reader and Display (50-key)
  - Point of Sale Keyboard VI<sup>1</sup>
- IBM 4820 SurePoint Solution Keypad<sup>2</sup>

The following keyboards are referred to in this book as the *alphanumeric point-of-sale keyboards*:

- Alphanumeric Point of Sale (ANPOS) Keyboard
- Retail Alphanumeric Point of Sale Keyboard with Card Reader (USB supported)
- PC Point of Sale Keyboard

All IBM (SIO) point-of-sale keyboards, except the PLU Keyboard and Display-III, can be attached to socket 5A or 5B of the point-of-sale terminal. In addition, the SIO alphanumeric point-of-sale keyboards can be plugged into the system keyboard port.

The keyboard attached to serial input/output (SIO) socket 5A or 5B is referred to as the Point of Sale keyboard. The keyboard attached as the system keyboard port is referred to as the *system keyboard*.

A USB-attached keyboard can also function as the Point of Sale system keyboard. See Chapter 2. Universal Serial Bus Architecture and IBM Point of Sale Subsystem for Windows.

When the alphanumeric point-of-sale keyboard is plugged into the system keyboard port (using the system keyboard port or the USB port), it simulates either the PS/2 101-key or 102-key enhanced keyboard with additional keys, or the PS/55 5576-002 or 5576-A01 keyboard with additional keys. The point-of-sale keyboards require that you install the IBM Point of Sale Subsystem in order to function correctly as the Point of Sale system keyboard.

The PLU Keyboard and Display-III attaches to the PLU III Extension Box.

Additional characteristics of individual keyboards are listed in the following sections.

## Keyboard Microcode Updates

The internal microprocessor code can be updated by IBM for the following non-USB keyboards:

- Modifiable Layout Keyboard with Card Reader (133-key)
- PC Point of Sale Keyboard
- Point of Sale Keyboard VI
- Retail Alphanumeric Point of Sale Keyboard with Card Reader
- Retail Point of Sale Keyboard (50 key)
- Retail Point of Sale Keyboard with Card Reader (50-key)
- Retail Point of Sale Keyboard with Card Reader and Display (50-key)

IBM delivers the microprocessor code update in a file named AIPxxx.KBD, where xxx is the microprocessor engineering change (EC) level. For OS/2, this file must be in a directory specified in the DPATH statement in the CONFIG.SYS file in order to be found and applied by the IBM Point of Sale Subsystem for OS/2. For the

---

1. These keyboards are not supported on IBM Point of Sale Subsystem for Linux.

2. •The 4820 SurePoint Solution Keypad will come online even when there is no keypad physically attached to the 4820 Display.

- When the keypad is attached to the 4820 display, the reported subtype will be PosKBD\_SUBTYPE\_4820\_KEYPAD; when the keypad is not physically attached to the 4820 display, the reported subtype will be PosKBD\_SUBTYPE\_4820\_NO\_KEYS.
- This behavior can affect the enumeration of attached POS keyboards. For example, attaching a POS keyboard downstream from the 4820 display enumerates the keyboard as a secondary POS keyboard; attaching a POS keyboard upstream from the 4820 display enumerates the keyboard as a primary POS keyboard.

•The 4820 SurePoint Solution Keypad is not supported on Linux.

Microsoft Windows operating system, this file must be in the IBM Point of Sale Subsystem for Windows BIN directory in order to be found and applied by the IBM Point of Sale Subsystem for Windows.

**Note:** The preceding Keyboard Microcode Updates section applies to non-USB keyboards only. For USB keyboards, see “Appendix H. Firmware Update Utility for USB Devices” on page H-1.

## 50-Key Modifiable Layout Keyboard and 50-Key Modifiable Layout Keyboard and Operator Display

These keyboards can only function as point-of-sale keyboards.

**Note:** These keyboards are not supported on the IBM Point of Sale Subsystem for Linux.

The 50-Key Modifiable Layout Keyboard and the 50-Key Modifiable Layout Keyboard and Operator Display have the following characteristics:

- Four point-of-sale indicator lights
- Internal speaker for audible feedback
- Pairs of keys can be treated as a single key by applications
- Optional manager keylock
- Total of 50 keys made up of:
  - One 11-key numeric keypad
  - Two system keys
  - 37 function keys

Key switch numbers are used to define double keys for these keyboards. The assigned key switch numbers for these keyboards, and the scan codes generated by the key switches, are in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The IBM Point of Sale Subsystem assigns ASCII characters to the numeric keypad keys. All other keys on the 50-Key Modifiable Layout Keyboard are unassigned. Your application should use the generated scan codes to assign functions to key switches.

## Retail Point of Sale Keyboards

The retail point-of-sale keyboards are the following keyboards:

- Retail Point of Sale Keyboard (50-key)
- Retail Point of Sale Keyboard with Card Reader (50-key)
- Retail Point of Sale Keyboard with Card Reader and Display (50-key)
- Point of Sale Keyboard VI

**Note:** The Point of Sale Keyboard VI is not supported on the IBM Point of Sale Subsystem for Linux.

These keyboards can only function as point-of-sale keyboards. The keyboards have the following characteristics:

- Four point-of-sale indicator lights
- Internal speaker for audible feedback
- Pairs of keys can be treated as a single key by applications
- Optional manager keylock
- Total of 50 keys made up of:
  - One 12-key numeric keypad
  - One **Ctrl** key

- 37 function keys with primary and secondary capability

If the keyboard has a built-in card reader, it is a low-profile 3-track Magnetic Stripe Reader (MSR).

**Note:** The Point of Sale Keyboard VI has an integrated “two-head” Magnetic Stripe Reader. One side will read ABA/ISO track 2/JIS-I track 2 and the other side will read JUCC/JIS-II.

Key switch numbers are used to define double keys for these keyboards. The assigned key switch numbers for these keyboards, and the scan codes generated by the key switches, are in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The default configuration provides 49 keys because the numeric keypad zero key is a double key. This key can be defined as two single keys using the **PosNumpadZero** resource. Key switch 28 is inactive by default.

Key switch 50 is the Ctrl key. It can be used in combination with other keys to access a second layer of functions.

The S1 and S2 keys are secondary functions of key switch 31 and key switch 32. They are activated by pressing the **Ctrl** key in combination with key 31 or 32. These keys cannot be part of a double key pair.

## Modifiable Layout Keyboard with Card Reader

The Modifiable Layout Keyboard with Card Reader functions as a point-of-sale keyboard.

**Note:** This keyboard is not supported on the IBM Point of Sale Subsystem for Linux.

This keyboard has the following characteristics:

- Four point-of-sale indicator lights
- Internal speaker for audible feedback
- Pairs of keys can be treated as a single key by applications
- Optional manager keylock
- Built-in low-profile 3-track Magnetic Stripe Reader
- Total of 133 keys made up of the following keys:
  - One 12-key numeric keypad
  - One **Ctrl** key
  - 120 function keys with primary and secondary capability

Key switch numbers are used to define double keys for this keyboard. The assigned key switch numbers for this keyboard, and the scan codes generated by the key switches, are in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The default configuration provides 132 keys because the numeric keypad zero key is a double key. This key can be defined as two single keys using the **PosNumpadZero** resource. Key switch 105 is inactive by default.

Key switch 111 is the Ctrl key. It can be used in combination with other keys to access a second layer of functions.

The S1 and S2 keys are secondary functions of key switch 135 and key switch 124. They are activated by pressing the **Ctrl** key in combination with key 135 or 124. These keys cannot be part of a double key pair.

There are three possible locations for the numeric keypad. All the locations are shaded in the illustration of this keyboard in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book. The default location for the numeric keypad is the right-most shaded area. The location of the numeric keypad is specified by the **PosNumpadLocation** resource.

## ANPOS Keyboard

The ANPOS keyboard can function either as a Serial Input Output (SIO) keyboard or as a system keyboard.

**Note:** This keyboard is not supported on the IBM Point of Sale Subsystem for Linux.

This keyboard has the following characteristics:

- Four point-of-sale indicator lights
- Three standard indicator lights
- Internal speaker for audible feedback
- Pairs of keys can be treated as a single key by applications
- Optional manager keylock
- Alphanumeric key section similar to the PS/2 101-key Enhanced Keyboard and the PS/2 102-key Enhanced Keyboard
- Point of Sale section containing programmable point-of-sale keys, similar to those on the 50-Key Modifiable Layout Keyboard
- Numeric keypad, shared by the alphanumeric and point-of-sale sections
- Optional low-profile dual-track Magnetic Stripe Reader
- Total of 116 keys
- Typematic rate and delay are settable (typematic feature can be set on or off)
- Double keys can be configured for this keyboard by using the ANPOS utility for system-attached keyboards; the point-of-sale application can configure double keys for the point-of-sale keyboards.

Key switch numbers are used to define double keys for the ANPOS keyboard. Key switch numbers are also used in this document when it is necessary to refer to a specific key. The assigned key switch numbers for this keyboard, and the scan codes generated by the key switches, are in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The default configuration provides 114 (U.S.) or 115 (world trade) keys because the numeric keypad zero key is a double key. This key can be defined as two single keys using the **PosNumpadZero** resource. Key switch 94 is inactive by default.

The S1 and S2 keys are secondary functions of key switch 110 and key switch 111. They are activated by pressing the **Ctrl** key (key switch 58 or 64) in combination with key 110 or 111. These keys cannot be part of a double key pair.

**Default State:** The keyboard hardware initializes itself to the following default state:

- All typematic functions are off
- Typematic rate is set to 10.9 characters per second  $\pm 20\%$

- Typematic delay is set to 500 milliseconds
- “Fat-finger” timeout is set to 30 milliseconds
- Key “click” is off
- Wait and OffLine indicator lights are on
- MSR is disabled if attached

When the ANPOS keyboard device handler is initialized and has established communications with the keyboard, commands are sent to the keyboard to set the following default state:

- NumLock is OFF
- All indicator lights are OFF
- Typematic function is ON
- Typematic rate is set to 30 characters per second  $\pm 20\%$
- Typematic delay is set to 250 milliseconds
- The keyboard is enabled and is scanning for key entry to be sent in to the point-of-sale terminal.

If the default values are not acceptable, most of them can be overridden. For SIO keyboards, this can be done when your application opens the keyboard. For the system keyboard, this can be done via the ANPOS utility. See “Configuring the Alphanumeric Point of Sale Keyboard” on page 3-4 for more information on the ANPOS utility.

## Retail Alphanumeric Point of Sale Keyboard with Card Reader

The Retail Alphanumeric Point of Sale Keyboard with Card Reader functions as either a point-of-sale keyboard or a system keyboard.

This keyboard has the following characteristics:

- Four point-of-sale indicator lights
- Three standard indicator lights
- Internal speaker for audible feedback
- Pairs of keys can be treated as a single key by applications
- Optional manager keylock
- Alphanumeric key section similar to the PS/2 101-key enhanced keyboard
- Point of Sale section that contains programmable point-of-sale keys, similar to those on the 50-Key Modifiable Layout Keyboard
- Numeric keypad that is shared by the alphanumeric section and the point-of-sale section
- Built-in low-profile 3-track MSR
- Total of 117 keys
- Typematic rate and delay are settable (the typematic feature can be set on or off)
- The configuration of double keys can only be changed by using the ANPOS utility for system-attached keyboards

Key switch numbers are used to define double keys for the Retail Alphanumeric Point of Sale Keyboard with Card Reader. The assigned key switch numbers for this keyboard, and the scan codes generated by the key switches, are in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The default configuration provides 115 (U.S.) or 116 (world trade) keys because the numeric keypad zero key is a double key. This key can be defined as two single keys using the **PosNumpadZero** resource. Key switch 99 is inactive by default.

The **S1** and **S2** keys are secondary functions of key switch 135 and key switch 124. They are activated by pressing the **Ctrl** key (key switch 58 or 64) in combination with key 135 or 124. These keys cannot be part of a double key pair.

There are two possible locations for the numeric keypad. Both locations are shaded in the illustration of this keypad in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book. The default location for the numeric keypad is the right-most shaded area.

When the keyboard is a point-of-sale keyboard, the numeric keypad can be moved to the left-most shaded area. The location of the numeric keypad is specified by the **PosNumpadLocation** resource. This feature is not available when the keyboard is attached as the system keyboard port.

**Default State:** The keyboard hardware initializes itself to the following default state:

- All typematic functions are off
- Typematic rate is set to 10.9 characters per second  $\pm 20\%$
- Typematic delay is set to 500 milliseconds
- “Fat-finger” timeout is set to 30 milliseconds
- Key “click” is off
- Wait and OffLine indicator lights are on
- MSR is disabled if attached

When the Retail Alphanumeric Point of Sale Keyboard with Card Reader device handler is initialized and has established communications with the keyboard, commands are sent to the keyboard to set the following default state:

- NumLock is OFF
- All indicator lights are OFF
- Typematic function is ON
- Typematic rate is set to 30 characters per second  $\pm 20\%$
- Typematic delay is set to 250 milliseconds
- The keyboard is enabled and is scanning for key entry to be sent in to the point-of-sale terminal.

If the default values are not acceptable, most of them can be overridden. For point-of-sale keyboards, this can be done when your application opens the keyboard. For the system keyboard, this can be done via the ANPOS utility. See “Configuring the Alphanumeric Point of Sale Keyboard” on page 3-4 for more information on the ANPOS utility.

## PC Point of Sale Keyboard (ANKPOS Keyboard)

The PC Point of Sale Keyboard, also known as ANKPOS keyboard, can function as either a point-of-sale keyboard or as a system keyboard.

**Note:** This keyboard is not supported on the IBM Point of Sale Subsystem for Linux.

This keyboard has the following characteristics:

- Four point-of-sale indicator lights. Viewed from the keyboard left to right, these lights are Ready, Wait, Offline, and Message Pending.
- Internal speaker for audible feedback
- Four-position key lock that controls the keyboard mode. Viewed from the keyboard clockwise, these positions are System, Inactive, Operator, Manager
- Alphanumeric key section similar to the PS/55 5576-002 or 5576-A01 keyboards

- Point of Sale section that contains programmable point-of-sale keys, similar to those on the 50-Key Modifiable Layout Keyboard
- Numeric keypad, shared by the alphanumeric section and the point-of-sale section
- 12 function keys
- Three standard indicator lights
- Built-in two-head JIS-II, JIS-I track 2 Magnetic Stripe Reader (MSR)
- Total of 121 keys
- Typematic rate and delay are settable (the typematic feature can be set on or off)
- Double keys can only be defined when the keyboard is functioning as the system keyboard. The ANPOS utility allows you to configure the double keys.

The assigned key switch numbers for this keyboard, and the scan codes generated by the key switches, are in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The numeric keypad zero key is key switch 94. Key switch 99 is not part of the numeric keypad zero key and therefore can be assigned another function by the application. The numeric keypad location is shaded in the illustration of this keypad in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The S1 and S2 keys are secondary functions of key switch 135 and key switch 124. They are activated by pressing the **Ctrl** key in combination with key 135 or 124.

**Default State:** The keyboard hardware initializes itself to the following default state:

- All typematic functions off
- Typematic rate is set to 30 characters per second  $\pm 20\%$
- Typematic delay is set to 250 milliseconds
- “Fat-finger” timeout is set to 30 milliseconds
- Key “click” off
- Wait and OffLine indicator lights are on
- Magnetic Stripe Reader is disabled (if attached)

When the PC Point of Sale Keyboard device handler is initialized and has established communications with the keyboard, commands are sent to the keyboard to set the following default state:

- NumLock is OFF
- All indicator lights are OFF
- Typematic function is ON
- Typematic rate is set to 30 characters per second  $\pm 20\%$
- Typematic delay is set to 250 milliseconds
- The keyboard is enabled and is scanning for key entry to be sent in to the point-of-sale terminal

If the default values are not acceptable, most of them can be overridden. For point-of-sale keyboards, this can be done when your application opens the keyboard. For the system keyboard, this can be done via the ANPOS utility. See “Configuring the Alphanumeric Point of Sale Keyboard” on page 3-4 for more information on the ANPOS utility.

## Point of Sale Keyboard V

The Point of Sale Keyboard V functions only as a point-of-sale keyboard.

**Note:** This keyboard is not supported on the IBM Point of Sale Subsystem for Linux.

This keyboard has the following characteristics:

- Four point-of-sale indicator lights
- Internal speaker for audible feedback
- Four-position key lock that controls the keyboard mode
- Built-in two-head JIS-II, JIS-I track 2 Magnetic Stripe Reader
- Total of 67 keys in the default configuration
- 11-key numeric keypad
- The configuration of double keys cannot be changed
- A tone sounds whenever any key is pressed as long as the keyboard is online and the keylock is not in the “inactive” position.

The *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book illustrates the layout of the key switches for the Point of Sale Keyboard V and the scan codes generated by the key switches. There are no assigned key switch numbers for the Point of Sale Keyboard V. Key switch numbers are not used for this keyboard in this document.

There are two possible locations for the numeric keypad. Both locations are shaded in the illustration of this keypad in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book. The default location for the numeric keypad is the right-most shaded area. The location of the numeric keypad is specified by the **PosNumpadLocation** resource.

The S1 and S2 keys are only active when the keylock is in the “system” position.

## PLU Keyboard and Display-III

The PLU Keyboard and Display-III functions as a point-of-sale keyboard.

**Note:** This keyboard is not supported on the IBM Point of Sale Subsystem for Linux.

The PLU keyboard has the following characteristics:

- Four point-of-sale indicator lights
- Internal speaker for audible feedback
- Total of 120 keys in the default configuration
- 11-key numeric keypad
- The configuration of double keys cannot be changed
- A tone sounds whenever any key is pressed as long as the keyboard is online and the keylock is not in the “inactive” position.

The *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book illustrates the layout of the key switches and the scan codes for the PLU Keyboard and Display-III. There are no assigned key switch numbers for the PLU Keyboard and Display-III. Key switch numbers are not used for this keyboard in this document.

There are two possible locations for the numeric keypad. Both locations are shaded in the illustration of this keypad in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book. The default location for the numeric keypad is the right-most shaded area. The location of the numeric keypad is specified by the **PosNumpadLocation** resource.

The S1 and S2 keys are only active when the keylock is in the “system” position.

## 4685 Point of Sale Keyboard Model K01

The 4685 Point of Sale Keyboard Model K01 functions as an SIO keyboard.

**Note:** This keyboard is not supported on the IBM Point of Sale Subsystem for Linux.

This keyboard has the following characteristics:

- Four point-of-sale indicator lights
- Internal speaker for audible feedback
- Four-position key lock that controls the keyboard mode
- Built-in two-head JIS-II, JIS-I track 2 magnetic stripe reader (MSR)
- Total of 67 keys in the default configuration
- 11-key numeric keypad
- The configuration of double keys cannot be changed
- A tone sounds whenever any key is pressed as long as the keyboard is online and the keylock is not in the “inactive” position.

The *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book illustrates the layout of the key switches for the 4685 Point of Sale Keyboard Model K01 and the scan codes generated by the key switches. There are no assigned key switch numbers for the 4685 Point of Sale Keyboard Model K01. Key switch numbers are not used for this keyboard in this document.

The location for the numeric keypad is shaded in the illustration of the 4685 Point of Sale Keyboard Model K01 shown in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book. The default location for the numeric keypad is the shaded area on the right. The location of the numeric keypad is specified by the **PosNumpadLocation** resource.

The S1 and S2 keys are only active when the keylock is in the “system” position.

## IBM 4820 SurePoint Solution Keypad

This keyboard can only function as an SIO keyboard.

The IBM 4820 SurePoint Solution Keypad has the following characteristics:

- Internal speaker for audible feedback
- Pairs of keys can be treated as a single key by applications
- Optional manager keylock
- Total of 32 keys made up of:
  - 11-key numeric keypad
  - 2 system keys
  - 19 function keys

Key switch numbers are used to define double keys for these keyboards. The assigned key switch numbers for these keyboards, and the scan codes generated by the key switches, are in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The IBM Point of Sale Subsystem assigns ASCII characters to the numeric keypad keys. All other keys on the IBM 4820 SurePoint Solution Keypad are unassigned. Your application should use the generated scan codes to assign functions to key switches.

**Notes:**

1. The 4820 SurePoint Solution Keypad will come online even if there is no keypad physically attached to the 4820 Display. When the 4820 SurePoint Solution Keypad is attached to the display, the reported subtype will be PosKBD\_SUBTYPE\_4820\_KEYPAD; when the 4820 SurePoint Solution Keypad is not physically attached to the display, the reported subtype will be PosKBD\_SUBTYPE\_4820\_NO\_KEYS.
2. The 4820 SurePoint Solution keypad comes online even if a physical keypad is not attached. This behavior affects the enumeration of attached POS keyboards depending on their USB port location. For example, attaching a POS Keyboard downstream from the 4820 display enumerates the keyboard as a secondary POS keyboard. Attaching a POS Keyboard upstream from the 4820 display enumerates the keyboard as a primary POS keyboard.
3. This keyboard is not supported on the IBM Point of Sale Subsystem for Linux.

**Defining Keys**

The *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* contains the complete list of keyboard scan codes for both system and point-of-sale keyboards. For point-of-sale keyboards, your IBM Point of Sale Subsystem application will receive scan codes from the point-of-sale scan code set. For system keyboards, your IBM Point of Sale Subsystem application will receive scan codes from the the PS/2 scan code set 1. Some keyboards can function as either a point-of-sale keyboard or system keyboard.

Key meanings are assigned by the operating system or by the application:

**Assigned by the Operating System:**

For the system keyboard, the operating system is involved in processing the keystrokes. It performs the scan code-to-ASCII character code translations, keeps track of shift states, provides hot keys, and provides special function keys like **Pause**.

The following keyboards have additional point-of-sale-unique keys that are not known to the operating system:

- ANPOS Keyboard
- PC Point of Sale Keyboard
- Retail Alphanumeric Point of Sale Keyboard with Card Reader

The ASCII character code for these point-of-sale-unique keys will be hexadecimal zero (0x00) in the keyboard event message. Your application must check the scan code field in the keyboard event message in order to detect that these keys have been pressed.

**Assigned by the Application:**

For a point-of-sale keyboard, the keyboard device handler calls the operating system to perform the scan code-to-ASCII character code translations and to keep track of shift states. Hot key sequences and special function keys must be handled by your application.

The keystroke event message received by your application provides the scan code corresponding to the key switch. If the scan code can be translated to an ASCII character code, the PosKC\_CHAR flag is set, and the ASCII character code is also put in the event message.

**Notes:**

1. All standard PS/2 101-key or 102-key enhanced keyboard or PS/55 5576-002 or 5576-A01 keyboard key definitions for a process are taken from the operating system for the code page, country and subcountry at the time the *PosOpen()* subroutine is called by that process.
2. See “Getting Input Messages” on page 5-2 for more information about the input queue and receiving event messages.

## Restriction of the Keyboard Device Handler

The keyboard device handler does not support the *PosWrite()* subroutine. In addition, it does not support the *PosRead()* subroutine for the device descriptor that you receive from the *PosOpen()* subroutine. The reason is that all keyboard input is received from the presentation facility input queue (for example, from the Presentation Manager event queue) or by issuing a *PosRead()* subroutine for the IBM Point of Sale Subsystem input queue (device descriptor zero).

---

## Functions Your Application Performs

Your application can perform the following functions with the keyboard:

- Read keyboard data
- Use the manager key lock
- Use the keyboard tone
- Use the keyboard point-of-sale-unique keys
- Control the keyboard click
- Control the Num Lock and Scroll Lock keys
- Use the point-of-sale-unique keys
- Control the system hot keys
- Control the keyboard typematic function
- Specify the numeric keypad style
- Specify the numeric keypad location

Before your application program can access the point-of-sale keyboard or the point-of-sale portion of the system keyboard, you must open the keyboard (see “Opening Your Device” on page 5-4) and acquire exclusive use of it (see “acquire” on page 5-6).

## Reading Keyboard Data

The keyboard can be in a locked state or an unlocked state. Data cannot be read when the keyboard is in the locked state. The keyboard is initially in the unlocked state. Your application reads all keyboard input from the presentation facility input queue or by issuing *PosRead()* for the IBM Point of Sale Subsystem input queue (device descriptor zero). All of the information about a keystroke is contained in the event message. No other *PosRead()* call is required to get additional information. See “Getting Input Messages” on page 5-2 for more information.

The keyboard device descriptor obtained from the *PosOpen()* subroutine should not be used on the *PosRead()* subroutine. If it is, the 311 `POSERR_SYS_FUNCTION_NOT_SUPPORTED` error is set in *errno*.

**Note:** All keyboard *PosIOctl()* requests use the device descriptor obtained from the *PosOpen()* subroutine.

### System Keyboard Considerations

Applications have access to the system keyboard through the keyboard interface provided by the operating system. The keystrokes from the system

keyboard are delivered by way of the presentation facility input event queue or the operating system's keyboard *DosDevIOctl()* calls. The system keyboard keystrokes are never delivered by way of the IBM Point of Sale Subsystem input queue.

The `POS_SYS_ACQUIRE_DEVICE PosIOctl()` request and the `POS_SYS_RELEASE_DEVICE PosIOctl()` request have no effect on the receipt of keystrokes. Keystrokes are still handled by the operating system. These requests only control which application is allowed to issue *PosIOctl()* requests that affect the point-of-sale functions of the system keyboard.

#### Point of Sale Keyboard Considerations

The point-of-sale keyboard event messages are sent to the application that has the device connection acquired. If the connection is not acquired, the event messages are discarded.

### Using the Manager Keyboard Lock

The application that has acquired the keyboard device receives the `POSM_KBD_STATUS_CHANGE` event message on its message queue whenever the position of the manager keyboard lock changes. The message contains the key lock position.

To determine the current state of the keyboard lock at other times, use the `POS_SYS_GET_VALUES` request of the *PosIOctl()* subroutine to retrieve the value of the **PosNkeyLock** resource. See "PosNkeyLock" on page 21-25 for more information.

### Using the Keyboard Tone

The internal keyboard speaker is sounded by issuing the `POS_KBD_SOUND_TONE PosIOctl()` request. The frequency, duration, and volume of the tone are determined by the values of the **PosNtoneFreq**, **PosNtoneDuration**, and **PosNtoneVolume** resources. Use any of the following methods to specify values for these resources:

- Specify the value in the resource file.
- Set the value using the `POS_SYS_SET_VALUES PosIOctl()` request.
- Specify the value using the *args* and *nargs* parameters of the `POS_KBD_SOUND_TONE PosIOctl()` request.

If your application sounds the tone with the **PosNtoneDuration** resource set to **PosON**, it must issue the `POS_KBD_SILENCE_TONE PosIOctl()` request to silence the tone.

See "POS\_KBD\_SOUND\_TONE" on page 19-20 and "POS\_KBD\_SILENCE\_TONE" on page 19-19 for more information.

### Using the Keyboard Point of Sale Lights

The four point-of-sale lights are arranged in one row. You can control these lights with the `POS_SYS_SET_VALUES PosIOctl()` request. The four lights are controlled by the bit settings of the low-order four bit positions of the **PosNkeyboardLightsOn** resource byte. Bit 3 controls the left-most light, bit 2 controls the middle-left light, bit 1 controls the middle-right light and bit 0 controls the right-most light.

Set the point-of-sale lights for the keyboard by specifying the value for the **PosNkeyboardLightsOn** resource. Use any of the following methods to change the settings:

- Specify the value in the resource file.
- Use the *args* and *nargs* parameters of the `POS_SYS_SET_VALUES PosIOctl()` request.
- Use the *args* and *nargs* parameters of the `PosOpen()` subroutine call.

The keyboard lights are turned on or off when the **PosNkeyboardLightsOn** resource of the acquired keyboard connection is changed.

See “PosNkeyboardLightsOn” on page 21-26 for more information.

## Controlling the Keyboard Click

The keyboard click is off by default. You can change this setting by changing the value of the **PosNkeyboardClick** resource. Use any of the following methods to change the setting:

- Specify the value in the resource file.
- Use the *args* and *nargs* parameters of the `POS_SYS_SET_VALUES PosIOctl()` request.
- Use the *args* and *nargs* parameters of the `PosOpen()` subroutine call.

The keyboard click is turned on or off when the **PosNkeyboardClick** resource of the acquired keyboard connection is changed.

See “PosNkeyboardClick” on page 21-25 for more information.

## Controlling the Num Lock Key

The `POS_KBD_ENABLE_NUM_LOCK` and `POS_KBD_DISABLE_NUM_LOCK PosIOctl()` requests enable and disable the use of the **Num Lock** key on the alphanumeric point-of-sale keyboards. The state of the **Num Lock** key can be set by the `POS_KBD_SET_NUM_LOCK_ON` request and the `POS_KBD_SET_NUM_LOCK_OFF` request using the `PosIOctl()` subroutine. If you want to keep the operator from using the **Num Lock** key, your application should disable the key and set the key state.

## Controlling the Scroll Lock key

The `POS_KBD_ENABLE_SCROLL_LOCK` and `POS_KBD_DISABLE_SCROLL_LOCK PosIOctl()` requests enable and disable the use of the **Scroll Lock** key on the alphanumeric point-of-sale keyboards. The state of the **Scroll Lock** key can be set by the `POS_KBD_SET_SCROLL_LOCK_ON` request and the `POS_KBD_SET_SCROLL_LOCK_OFF` request using the `PosIOctl()` subroutine. If you want to keep the operator from using the **Scroll Lock** key, your application should disable the key and set the key state.

## Controlling the Point of Sale-Unique Keys

**Presentation Manager and Microsoft Windows Programming Considerations:**  
Because all Presentation Manager applications and all applications for the Microsoft Windows run in the same session, your application **must** use these requests. Your application must acquire the keyboard to enable scanning of the point-of-sale keys when it gains focus and it must release the keyboard to disable scanning of the point-of-sale keys before it loses focus. This must be done to ensure that non-point-of-sale applications do not receive unexpected input from the point-of-sale keys.

## Controlling the System Hot Keys

The `POS_KBD_ENABLE_HOT_KEYS` and `POS_KBD_DISABLE_HOT_KEYS` *PosIOctl()* requests enable and disable the use of the system hot keys for the point-of-sale keyboard when it is attached as the system keyboard port.

## Controlling the Keyboard Typematic Function

The `POS_KBD_SET_TYEMATIC_OFF` and `POS_KBD_SET_TYEMATIC_ON` keyboard *PosIOctl()* requests control the keyboard typematic function. The typematic rate and delay are controlled by the **PosNtypematicDelay** resource and the **PosNtypematicFreq** resource. See “PosKeyboard Resource Set” on page 21-22 for more information about these resources.

## Specifying the Numeric Keypad Style

The default numeric keypad style for point-of-sale keyboards is a calculator layout with the bottom right key defined as the slash (/). This style can be changed for Point of Sale keyboards by specifying a value for the **PosNnumpadStyle** resource in the resource file or on the *PosOpen()* subroutine call. The numeric keypad style cannot be changed for point-of-sale system keyboards.

## Specifying the Numeric Keypad Location

The default location for the numeric keypad is on the right. The Modifiable Layout Keyboard with Card Reader and the Retail Alphanumeric Point of Sale Keyboard with Card Reader allow the numeric keypad location to be changed when the keyboard is a point-of-sale keyboard. This can be done by specifying a value for the **PosNnumpadLocation** resource in the resource file or on the *PosOpen()* subroutine call. The numeric keypad location cannot be changed for Point of Sale system keyboards.

---

## Related Information

Additional information about keyboard programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOctl() Requests
- Chapter 20. Event Messages
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with Keyboard

See Chapter 18. Application Programming Interface:

```
PosClose()
PosIOctl()
PosOpen()
PosRead()
```

## Keyboard PosIOctl() Control Requests

See Chapter 19. PosIOctl() Requests:

```
POS_KBD_DISABLE_HOT_KEYS
POS_KBD_DISABLE_NUM_LOCK
POS_KBD_DISABLE_SCROLL_LOCK
POS_KBD_ENABLE_HOT_KEYS
POS_KBD_ENABLE_NUM_LOCK
POS_KBD_ENABLE_SCROLL_LOCK
```

- POS\_KBD\_SET\_NUM\_LOCK\_OFF
- POS\_KBD\_SET\_NUM\_LOCK\_ON
- POS\_KBD\_SET\_SCROLL\_LOCK\_OFF
- POS\_KBD\_SET\_SCROLL\_LOCK\_ON
- POS\_KBD\_SET\_TYPEMATIC\_OFF
- POS\_KBD\_SET\_TYPEMATIC\_ON
- POS\_KBD\_SILENCE\_TONE
- POS\_KBD\_SOUND\_TONE
- POS\_SYS\_ACQUIRE\_DEVICE
- POS\_SYS\_GET\_VALUES
- POS\_SYS\_LOCK\_DEVICE
- POS\_SYS\_RELEASE\_DEVICE
- POS\_SYS\_SET\_VALUES
- POS\_SYS\_UNLOCK\_DEVICE

## Keyboard Event Messages

See Chapter 20. Event Messages:

- POSM\_KBD\_STATUS\_CHANGE
- POSM\_KBD\_WM\_CHAR
- WM\_CHAR

**Note:** The WM\_CHAR event message is generated by OS/2 Presentation Manager or the Microsoft Windows operating system for the point-of-sale-unique scan codes as well as for the standard PS/2 101-key and 102-key enhanced keyboard keys, or the PS/55 5576-002 and 5576-A01 keyboard keys.

## Keyboard Resources

See Chapter 21. Resource Sets:

- PosNdoubleKey01 - PosNdoubleKey60
- PosNfatFingerTimeOut
- PosNkeyboardClick
- PosNkeyLock
- PosNkeyboardLightsOn
- PosNnumpadLocation
- PosNnumpadStyle
- PosNnumpadZero
- PosNtoneDuration
- PosNtoneFreq
- PosNtoneVolume
- PosNtypematicDelay
- PosNtypematicFreq

## Keyboard Error Codes

See Appendix D. Error Codes:

- 4705 POSERR\_KBD\_INVALID\_DOUBLE\_KEY
- 4702 POSERR\_KBD\_INVALID\_DURATION
- 4706 POSERR\_KBD\_INVALID\_FAT\_FINGER\_TIMEOUT
- 4701 POSERR\_KBD\_INVALID\_FREQUENCY
- 4708 POSERR\_KBD\_INVALID\_KEYBOARD\_CLICK
- 4713 POSERR\_KBD\_INVALID\_NUMPAD\_LOCATION
- 4709 POSERR\_KBD\_INVALID\_NUMPAD\_STYLE
- 4710 POSERR\_KBD\_INVALID\_NUMPAD\_ZERO
- 4711 POSERR\_KBD\_INVALID\_TYPEMATIC\_DELAY
- 4712 POSERR\_KBD\_INVALID\_TYPEMATIC\_FREQ

created on October 2, 2001

## Keyboard Programming

4703 POSERR\_KBD\_INVALID\_VOLUME



---

## Chapter 10. Magnetic Stripe Reader Programming

The magnetic stripe reader (MSR) device handler is software that manages communication between your application and the MSR. An application can only access the device through the device handler.

---

### Characteristics of the MSR

The magnetic stripe reader (MSR) appears to the system to be an independent input/output (I/O) device, regardless of whether it is attached to the system unit or the keyboard.

The MSR is unlocked and locked under control of the IBM Point of Sale Subsystem. The MSR should only be unlocked immediately prior to the time it is used.

All IBM Point of Sale Subsystem MSRs have the following common characteristics:

- The MSR power-on state is locked.
- The MSR, when it is unlocked for long periods of time, can be affected by electromagnetic fields emitted by other equipment. This can trigger an input event message with data that is not valid, so it is best to unlock the MSR only when input is expected. If an input request is fulfilled by means other than the MSR (for example, keyboard input), the application is responsible for locking the MSR to prevent unintentional input.
- The MSR performs parity and longitudinal redundancy checks (LRC) before sending the data to the system. However, modulo checking is not performed by the MSR or the MSR device handler. The application code must perform modulo checking.

### One-Track Magnetic Stripe Reader

The one-track magnetic stripe reader (MSR) attaches to the top of the 50-Key Modifiable Keyboard. It connects by a cable to socket 6 on the back of the keyboard.

The one-track MSR has the following characteristics:

- Data encoded on a card (such as a credit card) can be read as the card is passed through the slot in the reader. The one-track MSR reads track 2 data as specified in the following standards:
  - *The ANSI Specifications for Magnetic-Stripe Encoding for Credit Cards*, ANSI X4.16-1983
  - *The ANSI Specifications for Credit Cards*, ANSI X4.13
- The ANSI standards specify that the data available to your application consists of the account number field, a separator character (X'0d') and a discretionary data field. The account number can be a maximum of 19 characters. The discretionary data field can be a maximum of 36 characters. The total number of characters occupied by the account number field, the separator character, and the discretionary data field cannot exceed 37 characters. The total number of characters, including the begin and end of field indicators and LRC characters, is 40.
- Track 2 data is numeric only, and is in BCD format.

#### Notes:

1. The hardware only detects the MSR presence when the keyboard is reset.

2. The one-track MSR is not supported on the IBM Point of Sale Subsystem for Linux.

### Dual-Track Magnetic Stripe Reader

The dual-track magnetic stripe reader comes in two styles that are functionally equivalent:

- Standard Profile
- Low Profile

**Note:** The Dual-Track MSR is not supported on the IBM Point of Sale Subsystem for Linux.

The standard profile dual-track MSR attaches to the top of the 50-Key Modifiable Keyboard and connects to socket 5A or 5B on the back of the point-of-sale terminal. The low profile dual-track MSR is optionally integrated into the ANPOS keyboards. It is connected to a socket internal to the keyboard.

The dual-track MSR has the following characteristics:

- There are two models available, one reads tracks 1 and 2, and the other reads tracks 2 and 3
- Track 1 data is formatted as alphanumeric, using only the lower 6 bits of each byte
- The maximum amount of data on track 1 is 79 characters
- Track 2 data has the same format and size as on the one-track MSR
- Track 3 data is numeric only, and has the same format as track 2 data (BCD)
- The maximum amount of data on track 3 is 107 characters
- If MSR data is read while the MSR is waiting for a response from the system, the error flag for that track is set

### Three-Track Magnetic Stripe Reader

The three-track magnetic stripe reader comes integrated in the following keyboards:

- Retail Point of Sale Keyboard with Card Reader
- Retail Point of Sale Keyboard with Card Reader and Display
- Modifiable Layout Keyboard with Card Reader
- Retail Alphanumeric Point of Sale Keyboard with Card Reader

The three-track magnetic stripe reader is an optional feature for the following:

- All IBM 4695 Point of Sale Distributed Touch Terminal Models
- All IBM 4695 Point of Sale Integrated Touch Terminal Models
- SurePoint Monochrome Touch Screen
- SurePoint Color Touch Screen
- IBM 4820 SurePoint Solution

The three-track MSR has the following characteristics:

- Track 1 data is formatted as alphanumeric, using only the lower 6 bits of each byte
- The maximum amount of data on track 1 is 79 characters
- Track 2 data has the same format and size as on the one-track MSR
- Track 3 data is numeric only, and has the same format as track 2 data (BCD)
- The maximum amount of data on track 3 is 107 characters
- If MSR data is read while the MSR is waiting for a response from the system, the error flag for that track is set

## Two-Head/Two-Sided Magnetic Stripe Reader

The two-head/two-sided magnetic stripe reader comes integrated in the following keyboards:

- PC Point of Sale Keyboard
- Point of Sale Keyboard V
- Point of Sale Keyboard VI
- 4685 Point of Sale Keyboard Model K01

**Note:** The two-head/two-sided MSR is not supported on the IBM Point of Sale Subsystem for Linux.

The two-head/two-sided MSR is an optional feature for the following:

- All IBM 4695 Point of Sale Distributed Touch Terminal Models
- All IBM 4695 Point of Sale Integrated Touch Terminal Models
- IBM 4820 SurePoint Solution

The two-head/two-sided MSR has the following characteristics:

- One side reads ABA/ISO/JIS-I track 2 data. The head in the other side of the slot reads the JUCC/JIS-II track.
- Track 2 data has the same format and size as the one-track MSR.
- The maximum amount of data on the JUCC/JIS-II track is 88 characters.

## Restriction of the MSR Device Handler

The MSR device handler does not support the *PosWrite()* subroutine.

---

## Functions Your Application Performs

Your application can perform the following functions with the MSR:

- Unlock the MSR
- Read MSR data
- Lock the MSR

Before your application program can access the MSR, it must open the MSR (see “Opening Your Device” on page 5-4) and acquire exclusive use of the MSR.

## Unlocking the MSR

The MSR has two states: locked and unlocked. In the locked state, data cannot be received. Initially, the MSR is in the locked state. Your application must issue a `POS_SYS_UNLOCK_DEVICE PosIOctl()` request to put the MSR in the unlocked state before it is able to read card data.

The MSR changes from unlocked to locked state when one of the following occurs:

- Data is read from a card.
- The application issues the `POS_SYS_LOCK_DEVICE PosIOctl()` request.
- The application issues the `POS_SYS_RELEASE_DEVICE PosIOctl()` request. Although the MSR is locked when your application releases it, the state is remembered. So, the next time your application acquires the MSR, it is unlocked automatically for you.
- The application issues the `PosClose()` request if your application has the MSR acquired at the time of this call.

Because a successful card read results in the MSR being locked, your application must issue the `POS_SYS_UNLOCK_DEVICE PosIOctl()` request each time it

prepares to read data from the MSR. The MSR should be left in the locked state any time that MSR data is not expected to prevent input that is not valid.

### Reading MSR Data

After your application has unlocked the MSR, it receives a `POSM_MSR_DATA_AVAIL` event message on its input queue when MSR data is available. See “Getting Input Messages” on page 5-2 for more information. Your application should then call the `PosRead()` subroutine using the MSR device descriptor to read the data. This subroutine reads data from all tracks of the MSR. The data is placed in the application’s buffer that was specified on this subroutine call.

Your application specifies the read buffer on the `PosRead()` subroutine using the `buf` and `nbyte` parameters. The number of `nbytes` is returned in the `POSM_MSR_DATA_AVAIL` event message. The buffer length value set in `nbyte` must be big enough to hold the maximum amount of data from all tracks plus each of the 8 byte headers (one per track).

A value of 0 (zero) for the buffer length indicates that no data is to be read. If the value of `nbyte` of `PosRead()` specifies a value too small for the record being read, the 312 `POSERR_SYS_BUFFER_TOO_SMALL` error is returned and data is not put into the application’s buffer.

`PosRead()` returns to your application immediately with either MSR data or an error code. If no data is available and no error occurs, the read completes successfully with a length of 0 (zero) returned.

If your application issues a `POS_SYS_UNLOCK_DEVICE PosIOctl()` request when data is available for it, the data is discarded to ensure that previous unprocessed MSR card data is not used in error.

The 335 `POSERR_SYS_LOCKED_NO_DATA_READ` error code is returned when the MSR is in the locked state and no data is available. Any data available prior to the MSR being locked will be returned by a read request before this error code is returned.

See “PosRead()” on page 18-16 for the syntax of this subroutine call.

#### MSR Read Buffer Format

Once your application successfully reads the MSR card data into its buffer, it is ready to interrogate the data. The format of the MSR data is described in Figure 10-1 on page 10-5.

There is a *header/data* pair for each unlocked track that contains data. The header portion of the returned data is defined in the structure `PosMsrDataHdr` (see the header file `C:\POS\INCLUDE\POS\MSR.H`). If there is no data and no error condition on a particular track, no header or data is returned to the application.

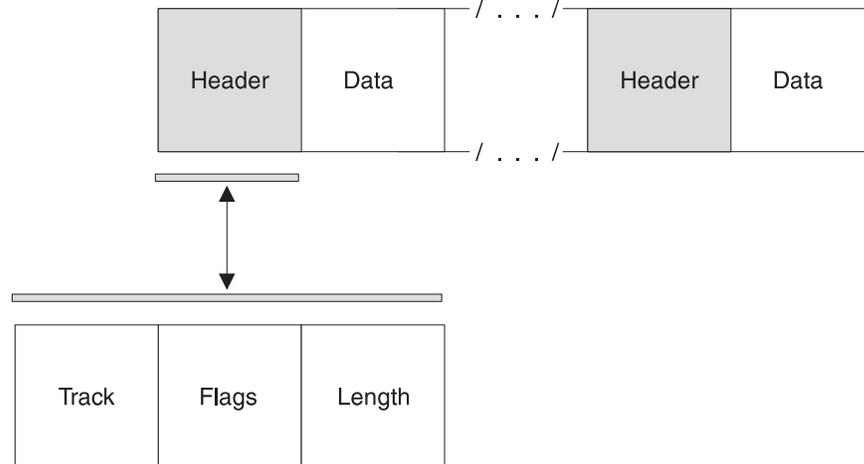


Figure 10-1. MSR Read Buffer Format

The format of each track is described by the following fields. The header is made up of the first three fields.

**Track** (*unsigned long*)

Identifies the MSR track.

Value	Meaning
<b>PosTRACK_1</b>	Track 1 data
<b>PosTRACK_2</b>	Track 2 data
<b>PosTRACK_3</b>	Track 3 data
<b>PosTRACK_J</b>	Track JUCC/JIS-II data

**Flags** (*unsigned short*)

Indicates the status of the data. These indicators should be checked on a track-by-track basis.

Value	Meaning
<b>POS_MSR_TRK_ERROR</b>	Error reading track

**Length** (*unsigned short*)

The length (0-65535) of the data read. This field is 0 (zero) if there was no data read for this track or if the flag POS\_MSR\_TRK\_ERROR is set.

**Data** (*unsigned char[ ]*)

The data that is read from the track on the card.

## Locking the MSR

The application can issue the POS\_SYS\_LOCK\_DEVICE *PosIOctl()* request to prevent the MSR from receiving card data. If your application issues a POS\_SYS\_LOCK\_DEVICE *PosIOctl()* request when data is available, the data is not discarded. This data is available to be read by the application until the application issues another POS\_SYS\_UNLOCK\_DEVICE *PosIOctl()* request.

---

## Related Information

Additional information about MSR programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOctl() Requests
- Chapter 20. Event Messages

- Appendix D. Error Codes

### Subroutines Used with MSR

See Chapter 18. Application Programming Interface:

PosClose()  
PosIOctl()  
PosOpen()  
PosRead()

### MSR PosIOctl() Control Requests

See Chapter 19. PosIOctl() Requests:

POS\_SYS\_ACQUIRE\_DEVICE  
POS\_SYS\_GET\_VALUES  
POS\_SYS\_LOCK\_DEVICE  
POS\_SYS\_RELEASE\_DEVICE  
POS\_SYS\_SET\_VALUES  
POS\_SYS\_UNLOCK\_DEVICE

### MSR Event Messages

See Chapter 20. Event Messages:

POSM\_MSR\_DATA\_AVAIL

### MSR Error Codes

There are no error codes that are specific to the MSR.

---

## Chapter 11. Non-Volatile Random Access Memory Programming

The non-volatile random access memory (NVRAM) device handler is software that manages communication between your application and the NVRAM. An application can only access the device through the device handler.

**Note:** For Windows 3.1 users, the IBM Point of Sale Subsystem for Windows does not support the NVRAM device on a satellite point-of-sale terminal.

---

### Characteristics of the NVRAM Device

The following is a list of characteristics of the NVRAM device:

- NVRAM provides battery-protected static storage. This area is available for your application to store data that needs to be saved if power is lost. For example, an application could use NVRAM to retain information about totals.
- An application can use the storage area in much the same way as it uses a direct file or a sequential file.
- For direct or sequential mode access, four overhead bytes of information are added to each record written:
  - 2 bytes for the length
  - 2 bytes for the cyclic redundancy check (CRC)

You must include this overhead when calculating record addresses or space requirements.

In sequential mode, there is an additional overhead of four bytes for an end-of-file (EOF) marker. When your application writes a record in sequential mode, an EOF marker is appended to the file after each record is written. The previous EOF marker is overlaid by each sequential write operation.

The NVRAM device handler automatically adds the required overhead bytes when you write a record. The overhead bytes are not passed to your application when you read a record.

- The maximum amount of data that can be read or written in a single operation is dependent on the type of point-of-sale terminal and on the mode of access. The following table shows the maximum amount of data that can be written or read at a time for each supported point-of-sale terminal, for direct and sequential mode.

*Table 11-1. Maximum Block Size for NVRAM Read and NVRAM Write Operations*

<b>Point of Sale Terminal</b>	<b>Direct Mode</b>	<b>Sequential Mode</b>
IBM 4683-x02	60	56
IBM 4683-421	1020	1016
IBM 4684-300	1020	1016
IBM 4693 (all models)	1020	1016
IBM 4694 (all models)	1020	1016
IBM 4695 (all integrated models)	1020	1016
IBM 4800-xxx	1020	1016
IBM 4695 Point of Sale Adapter	1020	1016
IBM 4695 Point of Sale Adapter/A	1020	1016

Table 11-1. Maximum Block Size for NVRAM Read and NVRAM Write Operations (continued)

Point of Sale Terminal	Direct Mode	Sequential Mode
IBM 7497-001 Attachment Adapter	1020	1016

An “x” in Table 11-1 on page 11-1 indicates that a number or alphabetic character can be substituted.

- The application address space is dependent on the type of point-of-sale terminal. The list below contains the NVRAM application address space for each supported IBM point-of-sale terminal.

4683-x02	0-1023
4683-421	0-28671
4684-300	0-4095
4693-all models	0-28671
4694-all models	0-28671
4695-all integrated models	0-126975
4800-xxx	0-111270
4695 Point of Sale Adapter	0-126975
4695 Point of Sale Adapter/A	0-126975
7497-001	0-28671

## Functions Your Application Performs

Your application can perform the following functions with the NVRAM device:

- Get the NVRAM application address space size
- Change the cursor position
- Open NVRAM in direct mode or sequential mode
- Read data in direct mode or sequential mode
- Write data in direct mode or sequential mode

Before your application program can access the NVRAM, it must open the NVRAM (see Opening Your Device) and acquire exclusive use of the NVRAM.

**Note:** The NVRAM device handler ensures that each read or write operation completes in its entirety before another read or write operation begins. This ensures that data is not interleaved if two write operations occur at the same time to the same area, but it does not protect one thread’s data from another thread, or one process’s data from another process.

## Available NVRAM Application Address Space

The available NVRAM application address space size can be queried by getting the value of the **PosNvramSize** resource with a `POS_SYS_GET_VALUES PosIOctl()` request.

## The Cursor Position

The cursor position specifies the NVRAM address of the start of the next record to be read or written. The cursor position is set by changing the **PosNvramCursor** resource with a `POS_SYS_SET_VALUES PosIOctl()` request. The cursor position can be queried by getting the value of the **PosNvramCursor** resource with a `POS_SYS_GET_VALUES PosIOctl()` request.

Following each successful read or write operation, the cursor position is advanced to the first byte of the next record. If an error occurs on either the `PosRead()` or the `PosWrite()` subroutine call, the cursor position is not advanced.

Note that the placement of the cursor position at or near the end of the NVRAM application address space does not cause an error. An error is returned on the next *PosRead()* or *PosWrite()* subroutine call.

## Opening NVRAM in Direct Mode or Sequential Mode

To open NVRAM, set the **PosNnvramMode** resource to `PosMODE_DIRECT` or `PosMODE_SEQUENTIAL` by specifying it as an argument on the *PosOpen()* subroutine call. The NVRAM connection stays in the mode you specified until the application issues a *PosClose()* subroutine call.

The default is `PosMODE_DIRECT`.

## Reading Data in Direct Mode or Sequential Mode

Set the cursor position to the address that you want to read. If you do not know the current position of the cursor, issue a `POS_SYS_GET_VALUES PosIOctl()` request for the **PosNnvramCursor** resource. If the current cursor position is not set to where you want it, issue a `POS_SYS_SET_VALUES PosIOctl()` for the **PosNnvramCursor** resource.

To read a record in direct mode or sequential mode, issue a *PosRead()* subroutine call. If the data is read successfully, the data (without the overhead bytes) is placed in the buffer specified on your read request. The cursor position is advanced to point to the first byte of the next record. Ensure that the buffer you use is large enough to hold the record being read. The read request returns the number of bytes actually read (copied) to your buffer. If your buffer is not large enough, the read operation completes with an error code of 312 `POSERR_SYS_BUFFER_TOO_SMALL`, and data is not put in the application's buffer. If any error occurs, the cursor position is not advanced. This allows your application an opportunity to get a larger buffer and reissue the *PosRead()* call without setting the cursor position again.

**Sequential Mode Only:** If the cursor is positioned at the EOF marker, the error code 4103 `POSERR_NVRAM_EOF` is returned on the next *PosRead()* call, and zero bytes are returned, indicating end-of-file. The cursor position is not advanced in this case, so all subsequent *PosRead()* calls continue to return zero bytes read until the cursor position is changed by setting the **PosNnvramCursor** resource to a valid address in the NVRAM application address space.

## Writing Data in Direct Mode or Sequential Mode

Set the cursor position to the address to which you want to write. If you do not know the current position of the cursor, issue a `POS_SYS_GET_VALUES PosIOctl()` request for the **PosNnvramCursor** resource. If the current cursor position is not set to where you want it, issue a `POS_SYS_SET_VALUES PosIOctl()` for the **PosNnvramCursor** resource. To write a record in sequential mode, issue a *PosWrite()* subroutine call. If the data is written successfully, the cursor position is advanced to point to the first byte of the next record (the EOF marker that marks the end of the current sequential stream). Writing another record overwrites the previous EOF and replaces it with the new record and the new EOF marker.

If the current cursor position plus the length of the data to be written extends beyond the end of the NVRAM application address space, no data is written and an error code of 4101 `POSERR_NVRAM_NOT_ENOUGH_ROOM` is returned on the *PosWrite()* subroutine call.

**Sequential Mode Only:** You can stop writing sequentially after any record, knowing that there is an EOF marker in place to mark the end of the sequential stream. The *PosWrite()* subroutine returns the number of bytes actually written from your buffer. If an error occurs, the cursor position is not advanced. If you position the cursor to a new address and then start writing sequentially again, it is possible to create multiple EOF markers in a single NVRAM application address space.

**Direct Mode Only:** To write a record in direct mode, issue a *PosWrite()* subroutine call. If the data is written successfully, the cursor position is advanced to point to the first byte of the next record. The write returns the number of bytes actually written (copied) from your buffer. If an error occurs, the cursor position is not advanced.

---

## Related Information

Additional information about NVRAM programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOCtl() Requests
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with NVRAM

See Chapter 18. Application Programming Interface:

PosClose()  
PosIOCtl()  
PosOpen()  
PosRead()  
PosWrite()

## NVRAM PosIOCtl() Control Requests

See Chapter 19. PosIOCtl() Requests:

POS\_SYS\_ACQUIRE\_DEVICE  
POS\_SYS\_GET\_VALUES  
POS\_SYS\_RELEASE\_DEVICE  
POS\_SYS\_SET\_VALUES

## NVRAM Resources

See Chapter 21. Resource Sets:

PosNnvramCursor  
PosNnvramMode  
PosNnvramSize

## NVRAM Error Codes

See Appendix D. Error Codes:

4103 POSERR\_NVRAM\_EOF  
4102 POSERR\_NVRAM\_INVALID\_CURSOR  
4106 POSERR\_NVRAM\_INVALID\_DATA\_CRC  
4105 POSERR\_NVRAM\_INVALID\_LENGTH\_RECORD  
4104 POSERR\_NVRAM\_INVALID\_MODE  
4101 POSERR\_NVRAM\_NOT\_ENOUGH\_ROOM

---

## Chapter 12. Printer Programming

The printer device handler is software that manages communication between your application and the printer. An application can only access the device through the device handler and only after it has successfully opened a connection to the device.

---

### Characteristics of the Printers

The IBM Point of Sale Subsystem supports the following point-of-sale printers:

- IBM Model 2 Printer
- IBM Model 3 Printer
- IBM Model 3F Fiscal Printer
- IBM Model 3R Printer
- IBM Model 4 Printer
- IBM Model 4A Printer
- IBM Model 4R Printer
- IBM 4689 Point of Sale Printer Models:  
001, 002, 301, 3G1, 3M1, TD5
- IBM 4610 SureMark Point of Sale Printer Models:  
TI1, TI2, TI3, TI4, TI5, TF6, TF7, TM6, TM7, TN3, TN4
- IBM 4610 SureMark Point of Sale Fiscal Printers

Several of the IBM point-of-sale printers are two-station printers that provide up to three functions:

- Customer receipt (CR) function in one station
- Summary journal (SJ) function in a second station
- Document insert (DI) function in the same CR and SJ stations

The CR function is used to provide a hard copy of the transaction. It can also function as a general output device to provide data to the user. The SJ function is used to record data on every transaction for an audit trail, or to perform any other printing function the application program dictates. The DI function is provided to allow the insertion of a form, document, or check directly into the printer from the front or from the top. This is used for a variety of print functions, such as:

- Printing store charge account forms
- Printing checks
- Endorsing checks

For the remainder of this book, the customer receipt function is referred to as the CR station, the summary journal is referred to as the SJ station, and the document insert referred to as the DI station.

### IBM Model 2 Printer

The IBM Model 2 printer has the following characteristics:

- Eight-wire bidirectional print mechanism
- Variable line spacing
- Emphasized print capability

**Note:** The IBM Model 2 Printer is not supported on the IBM Point of Sale Subsystem for Linux.

#### Print Mechanism

The printer mechanism is an 8-wire dot matrix, bidirectional, all-points-addressable device.

**Fonts**

The CR, SJ and DI stations each print up to 38 characters per line (CPL) at 15 characters per inch (CPI). Double-strike printing is available on all stations for emphasis.

Logo (all-points-addressable) (APA) printing is supported at the CR and DI stations.

**Line Length**

The maximum number of printable characters per line (CPL) for each font type and station type is shown in the following table:

*Table 12-1. Maximum Number of Characters for the IBM Model 2 Printer*

Station	15 CPI	LOGO
Customer Receipt	38 CPL	300 bytes
Summary Journal	38 CPL	n/a
Document Insert	38 CPL	300 bytes

**Note:** Each font control character pair inserted in the *PosWrite()* data field increases the size of the field by 2 bytes.

**Line Spacing**

The default line spacing for the CR and DI stations is 6 lines per inch. The default line spacing for the SJ station is 8 lines per inch. The line spacing for the CR and DI stations can be changed by the *PosIOctl()* subroutine call with the request of POS\_SYS\_SET\_VALUES and a resource of **PosNlineFeedCR** or **PosNlineFeedDI**.

Similarly, the current line spacing is returned by the POS\_SYS\_GET\_VALUES request of the *PosIOctl()* subroutine call. Refer to "POS\_SYS\_GET\_VALUES" on page 19-42 and "POS\_SYS\_SET\_VALUES" on page 19-45 for additional information.

**Emphasized Printing**

Emphasized printing is primarily intended to be used for printing on thick multi-part forms. A second strike of the print head wires on most forms results in the copies having darker, more easily read print. An application program puts the printer in emphasized mode by imbedding a double-density escape character sequence in the print data sent when calling the *PosWrite()* subroutine. Emphasized printing is available with all printer fonts at all the print stations.

**Performance**

The speed for printing on the CR station at 6 lines per inch with the normal font (15 CPI) can be up to 150 lines per minute, depending upon the application.

Logo printing is supported in the printer at approximately half the non-logo speed because of unidirectional printing.

Emphasized, or double-strike printing, is available in all printer stations at a speed that is one-third the normal printing speed. In emphasized print mode, the line is printed once, the print head is returned to the starting position, and the line is printed a second time. Every line of print requires three passes of the print head.

**IBM Model 3 and IBM Model 4 Printers**

The IBM Model 3 and Model 4 printers have the following characteristics in common:

- Nine-wire bidirectional print mechanism

- Multiple fonts
- Variable line lengths
- Variable line spacing
- Emphasized print capability

**Note:** These printers are not supported on the IBM Point of Sale Subsystem for Linux.

**Print Mechanism**

The printer mechanism is a 9-wire dot matrix, bidirectional, all-points-addressable high-speed device. The number 9 print wire can be used for an underscore or for descenders.

**Fonts**

The CR and SJ stations each print up to 38 characters per line at 15 characters per inch (CPI). The DI station prints up to 86 characters at 15 CPI. Other fonts are stored in the printer including 12 CPI, 17 CPI, and 7.5 CPI. The 7.5 CPI characters can also be printed double-high. Lower-case characters can be printed in addition to upper-case characters. Double-strike (emphasized) printing is available in all stations.

Logo (all-points-addressable) printing is supported at the CR and DI stations.

**Line Length**

The maximum number of printable characters per line (CPL) for each font type and station type is shown in the following table:

*Table 12-2. Maximum Number of Characters for IBM Model 3/3F/4/4A Printers*

Station	15 CPI	12 CPI	17 CPI*	Double-Wide 7.5 CPI	LOGO
Customer Receipt	38 CPL	30 CPL	42 CPL	19 CPL	380 bytes
Summary Journal	38 CPL	30 CPL	42 CPL	19 CPL	n/a
Document Insert Wide	86 CPL	68 CPL	94 CPL	43 CPL	880 bytes
Document Insert Narrow	38 CPL	30 CPL	42 CPL	19 CPL	380 bytes

\* Only supported on Model 4 (EC level 0X23) and Model 4R (EC level 0X3B and 0X3C)

**Note:** Each font control character pair inserted in the *PosWrite()* data field increases the size of the field by 2 bytes.

**Line Spacing**

The default line spacing for the CR and DI stations is 6 lines per inch. The default line spacing for the SJ station is 8 lines per inch. The line spacing for each station can be changed by the *PosIOctl()* subroutine call with the request of POS\_SYS\_SET\_VALUES and a resource of **PosNlineFeedCR**, **PosNlineFeedDI**, or **PosNlineFeedSJ**.

Similarly, the current line spacing is returned by the POS\_SYS\_GET\_VALUES request of the *PosIOctl()* subroutine call. Refer to “POS\_SYS\_GET\_VALUES” on page 19-42 and “POS\_SYS\_SET\_VALUES” on page 19-45 for additional information.

### Emphasized Printing

Emphasized printing is primarily intended to be used for printing on thick multi-part forms. A second strike of the print head wires on most forms results in the copies having darker, more easily read print. An application program puts the printer in emphasized mode by imbedding a double-density escape character sequence in the print data sent when calling the *PosWrite()* subroutine. Emphasized printing is available with all printer fonts at all the print stations.

### Font Specification

The pitch of the default font is 15 CPI. A font change can be specified by imbedding an escape control character, followed by a character designating the font type desired, in the data stream sent to the printer by the *PosWrite()* subroutine.

After receiving the escape character sequence denoting a font change, the printer continues to print in that font until another font change escape character sequence is received.

### Performance

The speed for printing on the CR station at 6 lines per inch with the normal font (15 CPI) can be up to 240 lines per minute, depending upon the application.

Logo printing is supported in the printer at approximately half the non-logo speed because of unidirectional printing. Double-high fonts are also printed at a lower speed.

Emphasized, or double-strike printing, is available in all printer stations at a speed that is one-third the normal printing speed. In emphasized print mode, the line is printed once, the print head is returned to the starting position, and the line is printed a second time. Every line of print requires three passes of the print head.

## IBM Model 3R and Model 4R Printers

The IBM Model 3R and the IBM Model 4R printers have the same characteristics as the IBM Model 3 and IBM Model 4 printers. However, the IBM Model 3R and the IBM Model 4R printers also have a magnetic ink character recognition (MICR) reader installed. The MICR reader allows application programs to read the account information on customer checks in the DI station.

**Note:** These printers are not supported on the IBM Point of Sale Subsystem for Linux.

## IBM Model 3F Fiscal Printer

The IBM Point of Sale Subsystem printer device handler supports several countries on the IBM Model 3F Fiscal Printer.

The IBM Model 3F Fiscal Printer has many of the characteristics of the IBM Model 3 Printer and the IBM Model 4 Printer. Where there are differences, it is because the fiscal printer is following the applicable point-of-sale fiscal laws of the supported countries. Some of the differences are:

- The fiscal printer supports only 2 lines per inch (LPI) values: 6 LPI and 8 LPI.
- The fiscal printer does not support logo printing.

### IBM Model 3F Fiscal Printer Restrictions

Refer to the country-specific printer documentation for printer restrictions

## IBM Model 4A Printer

The IBM Model 4A printer has the same characteristics as the IBM Model 3 and IBM Model 4 printers. However, the Model 4A printer allows fonts to be downloaded to the printer instead of having a fixed font in the ROM on the printer.

**Note:** The IBM Model 4A printer is not supported on the IBM Point of Sale Subsystem for Linux.

With the capability of defining new characters, the maximum number of characters per line can vary depending upon the font downloaded to the printer, the current font, and the height of the font. The line length can be calculated by dividing the number of *Addressable Print Positions Per Line* by the number of *Addressable Print Positions Per Character* for the current station, character set, font, and height.

### Addressable Print Positions Per Line

The maximum number of addressable print positions (dot columns) per line for each font type and station type is shown in the following table

*Table 12-3. Maximum Number of Addressable Print Positions Per Line for the IBM Model 4A Printer*

Station	15 CPI	12 CPI
Customer Receipt	380 Points	300 Points
Summary Journal	380 Points	300 Points
Document Insert Wide	860 Points	680 Points
Document Insert Narrow	380 Points	300 Points

### Addressable Print Positions Per Character

The maximum number of addressable print positions (dot columns) per character for each font type and station type is shown in the following table:

*Table 12-4. Maximum Number of Addressable Print Positions per Character for the IBM Model 4A Printer*

Character Set	Font	Height	Addressable Print Positions per Character
Single-Byte	Normal	9	10 Points
Single-Byte	Double-Wide and Double-High	9	20 Points
Double-Byte	Normal	9	WIDTH + SPACE
Double-Byte	Double-Wide	9	2 • (WIDTH + SPACE)
Double-Byte	Double-High	9	2 • (WIDTH + SPACE)
Double-Byte	Normal	16	2 • (WIDTH + SPACE)
Double-Byte	Narrow	16	WIDTH + SPACE

The values for WIDTH and SPACE in the previous table are the defined double-byte character sets font width and height. For more information about the WIDTH and SPACE keywords, see Appendix E. IBM Model 4A Font Download.

## IBM 4689-001 and IBM 4689-002 Printer

The IBM 4689-001 and IBM 4689-002 printers have the following characteristics:

- Nine-wire bidirectional print mechanism
- Multiple fonts
- Variable line lengths
- Variable line spacing

**Note:** These printers are not supported on the IBM Point of Sale Subsystem for Linux.

### Print Mechanism

The printer mechanism is an 18-wire (9 dot X 2 parallel) dot matrix.

### Fonts

The CR and SJ stations each print up to 30 characters per line. The DI station prints up to 70 characters per line. Another font is stored in the printer that can print up to 25 characters per line for the CR and SJ stations and up to 58 characters per line on the DI station. Lower-case characters can be printed in addition to upper-case characters.

### Line Length

The maximum number of printable characters per line (CPL) for each font type and station type is shown in the following table:

*Table 12-5. Maximum Number of Characters for IBM 4689-001/002 Printers*

Station	SBCS 25 CPL	SBCS 30 CPL	DBCS 25 CPL	DBCS 30 CPL
Customer Receipt	25 CPL	30 CPL	12 CPL	15 CPL
Summary Journal	25 CPL	30 CPL	12 CPL	15 CPL
Document Insert	58 CPL	70 CPL	29 CPL	35 CPL

### Line Spacing

The line spacing for the CR and DI stations is 6 lines per inch. The line spacing for the SJ station is either 6 lines per inch or 7.2 lines per inch. The line spacing for the SJ station is controlled by a hardware switch called the Journal Spacing switch. The line spacing for the CR and DI stations cannot be changed.

### Font Specification

There are two fonts available on the IBM 4689-001. One is 25 characters per line (CPL) and the other is 30 characters per line. The font specification(25 CPL or 30 CPL) is controlled by a hardware switch called the Character Spacing switch.

There is only one font (25 CPL) available on the IBM 4689-002 printer.

### Performance

The speed for printing on the CR station at 6 lines per inch can be up to 200 lines per minute, depending upon the application.

### Restrictions

If a document is inserted in the printer when the cover is opened, the document should be removed before the cover is closed.

## IBM 4689 Point of Sale Printer Model 301, 3G1, 3M1, and TD5

The IBM 4689-3x1 and TD5 printers consist of two stations (CR and SJ print stations only) and have the following characteristics:

- Thermal print mechanism
- Multiple fonts
- Multiple print directions
- Variable line lengths
- Variable line spacing
- Ruled lines (Keisen)
- Multiple contrast (Amikake) levels

**Note:** These printers are not supported on the IBM Point of Sale Subsystem for Linux.

### Print Mechanism

The printer mechanism is a thermal print head.

### Fonts

There are two font sizes in the 4689-3x1 and TD5 printers:

- 24x24 dot matrix (normal font)
- 16x16 dot matrix (small font)

Both the normal font and the small font support the full Kanji (JIS-I and JIS-II) character sets, the single-byte character set (Hankaku), and Kana characters. The size of the characters are dependent on both the font and the type of character:

- Normal font
  - 12x24 dot matrix for single-byte characters (Hankaku).
  - 24x24 dot matrix for double-byte characters (Zankaku).
- Small font
  - 8x16 dot matrix for single-byte characters (Hankaku).
  - 16x16 dot matrix for double-byte characters (Zankaku).

In addition to the two installed fonts, there is also the capability for 64 user-defined characters (UDC) using the 24x24 dot matrix font.

Lower-case characters can be printed in addition to upper-case characters.

The size of the characters can also be changed to:

- Yokobai (double-wide)
- Tatebai (double-high)
- Yonbai (double-wide and double-wide)

### Line Length

Both the CR station and the SJ station are able to print on the 58 mm wide thermal paper. Using the 24x24 dot matrix font (normal font), up to 32 single-byte characters per line can be printed. Using the 16x16 dot matrix font (small font), up to 42 single-byte characters per line can be printed. The maximum number of printable characters per line (CPL) for each font type and station type is shown in the following table:

*Table 12-6. Maximum Number of Characters for IBM 4689-3x1 and TD5 Printers*

Station	SBCS 32 CPL	SBCS 42 CPL	DBCS 32 CPL	DBCS 42 CPL
Customer Receipt	32 CPL	42 CPL	16 CPL	21 CPL

Table 12-6. Maximum Number of Characters for IBM 4689-3x1 and TD5 Printers (continued)

Station	SBCS 32 CPL	SBCS 42 CPL	DBCS 32 CPL	DBCS 42 CPL
Customer Receipt Rotated	14 CPL	19 CPL	14 CPL	19 CPL
Summary Journal	32 CPL	42 CPL	16 CPL	21 CPL

### Line Spacing

The line spacing for the CR station defaults to 6.78 lines per inch. The line spacing for the SJ station defaults to 9.24 lines per inch.

### Print Direction

There are two directions characters can be printed:

- Yokogaki (English direction)
- Tategaki (90 degrees clockwise rotated)

**Note:** The Tategaki direction can only be printed on the CR station using the normal (24x24) font.

### Contrast (Amikake)

Three levels of contrast (Amikake) can be specified for the characters:

- None (the default)
- Light
- Dark

### Line Characters (Keisen)

Line characters can be printed on the 4689 Model 3x1 and TD5 printers. These characters can be either solid lines (jissen) or dashed lines (tensen).

### Performance

The speed for printing on the CR and SJ station using the 24x24 dot matrix font can be up to 20 lines per second. Using the 16x16 dot matrix font, the speed can be up to 27 lines per second.

### Restrictions

During normal print operations, if the printer cover is opened while the printer is idle, no status change messages indicating the cover is open will be sent to the application.

## IBM 4610 SureMark Point of Sale Printer

The IBM 4610 SureMark Printer models TI1, TI2, TI3, TI4, and TI5 have the following characteristics in common:

- Separate CR and DI print mechanisms (no SJ station)
- Multiple fonts
- Variable line lengths
- Variable line spacing
- Multiple text print attributes
- Pre-defined messages and logos
- Barcode printing

The IBM 4610 SureMark Printer model TN3 has a separate thermal SJ station. The TN3 printer is supported in the IBM Point of Sale printer device handler as a Model 4 printer.

**Notes:**

1. Only 4610 Models TI1, TI2, TI3, and TI4 are supported on IBM Point of Sale Subsystem for OS/2.
2. Only 4610 Models TI1, TI2, TI3, TI4, TM6, TF6, TF7, and TM7 are supported on the IBM Point of Sale Subsystem for Linux.

**Print Mechanism**

The CR station uses a thermal print head and the DI station uses a 9-wire impact print head. There is no SJ station.

**Fonts**

The CR station can print up to 44 characters per line (CPL) at 15 characters per inch (CPI). The DI station can print up to 47 CPL at 15 CPI. Other fonts that are stored in the printer include 12 CPI, 17 CPI and 20 CPI. Lower-case characters can be printed in addition to upper-case characters.

Logo (all-points-addressable) printing is supported at the CR and at the DI station when the DI station is oriented for portrait printing (see “PosNdiOrientation” on page 21-35 resource for more information).

**Note:** The format of the logo data is dependent on the station. See Writing Data in Logo Mode for more information.

**Scaling Text, Text Attributes and Double High/Wide font:**

1. Scaling remains on until either it is turned off (with \x1b\x23\x00) or it is temporarily turned off by enabling double high/wide with the text attributes escape sequence.
2. If you are in double high/wide text attribute mode and you turn on scaling, text attributes are disabled until scaling is turned off.
3. Changing the font removes double high/wide text attributes and scaling.

**Line Length**

The maximum number of printable characters per line (CPL) for each font type and station type is shown in the following table:

*Table 12-7. Maximum Number of Characters for IBM 4610 SureMark Point of Sale Printer Printers*

<b>Station</b>	<b>15 CPI</b>	<b>12 CPI</b>	<b>17 CPI</b>	<b>20 CPI</b>	<b>LOGO</b>
Customer Receipt 80mm paper	44 CPL	34 CPL	48 CPL	57 CPL	576 dots
Customer Receipt 58mm paper	30 CPL	24 CPL	33 CPL	39 CPL	400 dots
Document Insert Wide	47 CPL	37 CPL	52 CPL	N/A	474 dots
Document Insert Narrow	38 CPL	30 CPL	42 CPL	N/A	384 dots

**Notes:**

1. For DI station printing, the wide and narrow widths listed (see Table 12-7) apply only when the DI station is oriented for portrait printing. When the printing at the DI station is oriented for landscape printing, the length of the print line is dependent on the length of the inserted document.

2. When the double-high font, the double-wide font, or the double-wide attribute is selected, (see Text Print Attributes) the characters per line will be half of what is listed (see Table 12-7 on page 12-9) .
3. For the CR station, the setting of the **PosNleftMarginCR** resource affects the number of characters which can be printed on a line (depending on whether you have 58 or 80-mm paper).

**Note:** Each font control character pair inserted in the *PosWrite()* data field increases the size of the field by 2 bytes.

### Line Spacing

The default line spacing for the CR and DI stations is 6 lines per inch. The line spacing for each station can be changed by the *PosIOCtl()* subroutine call with the request of POS\_SYS\_SET\_VALUES and a resource of **PosNlineFeedCR** or **PosNlineFeedDI**.

Similarly, the current line spacing is returned by the POS\_SYS\_GET\_VALUES request of the *PosIOCtl()* subroutine call. Refer to “POS\_SYS\_GET\_VALUES” on page 19-42 and “POS\_SYS\_SET\_VALUES” on page 19-45 for additional information.

### Text Print Attributes

The following table shows the print attributes applied to each font in a given station:

*Table 12-8. Text Print Attributes for the IBM 4610 SureMark Printer models.*

Attribute	CR Station	DI Station
Double-high	yes	portrait only
Double-wide	yes	yes
Underlined	yes	no
Overlined	yes	no
Inverted	yes	no
Rotated	yes	no

### Barcode Printing

The IBM 4610 SureMark Point of Sale Printer models have the capability of printing barcodes on the CR station.

### Pre-defined Messages and Logos

Commonly used messages and logos can be downloaded to the printer for later printing with an escape character sequence. Up to 25 messages (a maximum of 8K bytes total) and up to 40 logos (a maximum of 64K bytes total) can be pre-defined. These pre-defined messages and logos are stored in Flash EPROM on the printer and only need to be downloaded once.

### Font Specification

The pitch of the default font is 15 CPI. A font change can be specified by imbedding an escape control character, followed by a character designating the font type desired, in the data stream sent to the printer by the *PosWrite()* subroutine.

After receiving the escape character sequence denoting a font change, the printer continues to print in that font until another font change escape character sequence is received.

### Performance

For the 4610 SureMark Point of Sale Printer models TI1 and TI2, the speed for printing on the CR station can be up to 26 lines per second, and up to 4.35 lines per second on the DI station, depending on the application.

For the 4610 SureMark Point of Sale Printer models TI3, TI4, and TI5, the speed for printing on the CR station can be up to 52 lines per second, and up to 4.35 lines per second on the DI station, depending on the application.

If the CR station is not kept busy with multiple print lines outstanding, the maximum speed of the printer is not obtained.

Logo printing at the CR station is the same speed as text printing.

### MICR Recognition and Check Flipping

The IBM 4610 SureMark Models TI2 and TI4 printers have a built-in magnetic ink character recognition (MICR) reader and a built-in check flipping mechanism. The MICR reader allows application programs to read the account information on customer checks in the DI station. The check flipping mechanism allows an application program to automatically flip a check that is inserted in the DI station. This allows the reading of MICR information on the front of the check, and franking on the back of the check without any operator handling.

### IBM 4610 SureMark Point of Sale Fiscal Printer

The IBM Point of Sale Subsystem printer device handler supports the 4610 SureMark Point of Sale Fiscal Printers. The fiscal printers conform to fiscal laws of the supported countries. Refer to country-specific documentation for the country-specific information and for printer restrictions.

---

## Functions Your Application Performs

Your application can perform the following printer functions:

- Use different code pages
- Read data from the printer
- Write data to the printer
- Control the printed text with control characters
- Modify the printed text with escape character sequences
- Control, query, and set the printer with input/output control requests
- Change the printer configuration through resources
- Receive printer event messages

Before your application program can access the printer, it must open the printer device (see “Opening Your Device” on page 5-4) and acquire exclusive use of the printer.

## Code Page Support

The code page of the process at the time the *PosOpen()* subroutine call is issued is queried and saved by the IBM Point of Sale Subsystem.

### Notes:

1. Code pages are defined for the printer, but due to the printer’s fixed internal character set, not all characters are available in each code page. See the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book for the available characters for each code page. Code page 858 will be used for any code page not listed in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

2. There are three sets of characters for downloading to the Model 4A printer. One is for the single-byte characters, one is for Japanese characters, and the last one is for Korean characters. The first time the IBM Point of Sale Subsystem uses the Model 4A printer, it will download to the printer, the code page corresponding to the current country code defined for the system. If a different code page is needed, you may download the new code page with the Model 4A printer font download program. For more information, see Appendix E. IBM Model 4A Font Download.
3. The IBM 4689-001 and the IBM 4689-002 printers use the code pages located in the ROM on the printer. The code page resource is not supported on the 4689 family of printers.
4. See “IBM 4689 SurePOS Receipt Journal Printer” on page F-2 for information about loading fonts for this printer.
5. Code pages are not supported on USB fiscal printers. Fiscal printers that are set to fiscal mode (with *PosIOctl()* POS\_PRN\_ENABLE\_FISCAL\_PRINTING) do not use code pages to translate characters that are sent to the printer. The use of resource PosNcodePage will not have any affect when using the 4689 printer.

## Reading Data from the Printer

The IBM Point of Sale Subsystem device handler supports reading:

- MICR data
- Fiscal data

### Reading MICR Data

The IBM Model 3R printer, the IBM Model 4R printer, and the IBM 4610 SureMark models TI2 and TI4 printers have a MICR reader installed. When the IBM Point of Sale Subsystem printer device handler detects that the MICR reader is present, it enables applications to read data from the printer. If no MICR reader is detected by the printer device handler, a 311 POSERR\_SYS\_FUNCTION\_NOT\_SUPPORTED error is returned to the application program on the *PosRead()* subroutine call.

The application receives a POSM\_PRN\_DATA\_AVAIL event message on the IBM Point of Sale Subsystem point-of-sale input queue when data is available from the MICR reader. See “Getting Input Messages” on page 5-2 for more information about the IBM Point of Sale Subsystem point-of-sale input queue.

After your application receives a POSM\_PRN\_DATA\_AVAIL event message on its input queue, your application should call the *PosRead()* subroutine using the printer device descriptor to read the data. This subroutine reads the MICR data from the printer. The data is placed in the application’s buffer that was specified on the *PosRead()* subroutine call.

Your application specifies the read buffer on the *PosRead()* subroutine using the *buf* and *nbyte* parameters. The number of *nbytes* is returned in the POSM\_PRN\_DATA\_AVAIL event message. The buffer length value set in *nbyte* must be big enough to hold the maximum amount of data from the MICR reader. A value of 0 (zero) for the buffer length indicates that no data is to be read. If the value of *nbyte* of *PosRead()* specifies a value too small for the record being read, the 312 POSERR\_SYS\_BUFFER\_TOO\_SMALL error is returned and data is not put into the application’s buffer.

*PosRead()* returns to your application immediately with either MICR data or an error code. If no data is available, the read completes successfully with a length of 0 (zero) returned.

If your application issues a `POS_SYS_RELEASE_DEVICE PosIOctl()` request when MICR data is available for it, the data is discarded. This is done to ensure that previous unread MICR data is not used in error.

See “PosRead()” on page 18-16 for the syntax of the read subroutine calls.

### MICR Information

The MICR information is represented as ASCII characters:

MICR Information	ASCII Characters
0 - 9	ASCII '0' through '9' (0x30 through 0x39)
Transit character	ASCII 'T' (0x54)
On Us character	ASCII 'A' (0x41)
Dash	ASCII '-' (0x2D)
Unreadable character	ASCII '?' (0x3F)
Amount character	ASCII '\$' (0x24)
Blank	ASCII ' ' (0x20)
Special Character 1	ASCII 'a' (0x61)
Special Character 2	ASCII 'b' (0x62)
Special Character 1	ASCII 'c' (0x63)
Special Character 4	ASCII 'd' (0x64)
Special Character 5	ASCII 'e' (0x65)

### Reading Fiscal Data

Refer to the country specific “IBM Model 3F Fiscal Technical Specification” for the definition of the fiscal commands that return information to the application.

The application receives a `POSM_PRN_DATA_AVAIL` event message on the IBM Point of Sale Subsystem point-of-sale input queue when data is available from the fiscal printer. See “Getting Input Messages” on page 5-2 for more information about the IBM Point of Sale Subsystem point-of-sale input queue.

After your application receives a `POSM_PRN_DATA_AVAIL` event message on its input queue, your application should call the `PosRead()` subroutine using the printer device descriptor to read the data. This subroutine reads the fiscal data from the printer. The data is placed in the application’s buffer that was specified on the `PosRead()` subroutine call.

Your application specifies the read buffer on the `PosRead()` subroutine using the `buf` and `nbyte` parameters. The number of `nbytes` is returned in the `POSM_PRN_DATA_AVAIL` event message. The buffer length value set in `nbyte` must be big enough to hold the maximum amount of data from the fiscal printer. A value of 0 (zero) for the buffer length indicates that no data is to be read. If the value of `nbyte` of `PosRead()` specifies a value too small for the record being read, the 312 `POSERR_SYS_BUFFER_TOO_SMALL` error is returned and data is not put into the application’s buffer.

`PosRead()` returns to your application immediately with either fiscal data or an error code. If no data is available, the read completes successfully with a length of 0 (zero) returned.

If your application issues a `POS_SYS_RELEASE_DEVICE PosIOctl()` request when fiscal data is available for it, the data is discarded. This is done to ensure that previous unprocessed fiscal data is not used in error.

See “PosRead()” on page 18-16 for the syntax of the read subroutine calls.

## Writing Data to the Printer

There are five modes for writing data to the printer device handler:

- Normal mode
- Logo mode
- Fiscal mode (fiscal printer only)
- Load pre-defined messages mode (4610 SureMark printers only)
- Load pre-defined logos mode (4610 SureMark printers and 4689 Model 3x1 and TD5 printers only)

The mode of printing is set by the **PosNprintMode** resource on a `POS_SYS_SET_VALUES PosIOctl()` request. The only values this resource can have are:

- PosMODE\_NORMAL
- PosMODE\_LOGO
- PosMODE\_FISCAL (fiscal printer only)
- PosMODE\_LOAD\_MESSAGE (4610 SureMark printers only)
- PosMODE\_LOAD\_LOGO (4610 SureMark printers and 4689 model 3x1 and TD5 only)

Any other value results in a 4905 POSERR\_PRN\_INVALID\_MODE error code from the `POS_SYS_SET_VALUES` request of the `PosIOctl()` subroutine call.

**Note:** For fiscal printers and IBM 4689 printers, PosMODE\_LOGO can be specified. However, logo data will not be printed and no error will be returned to the application program.

The `PosWrite()` subroutine is used to print text strings, logo data, and fiscal commands. Text strings, which include ASCII characters, control characters, and escape character sequences, are printed when the **PosNprintMode** resource is set to PosMODE\_NORMAL.

See “PosWrite()” on page 18-19 for the syntax of the `PosWrite()` subroutine call.

## Writing Data in Normal Mode

Normal mode is the default mode for the non-fiscal printers. In normal mode, print data consists of ASCII characters, double-byte characters, control characters, and escape character sequences. To switch back to normal mode from logo mode or fiscal mode, set the **PosNprintMode** resource to PosMODE\_NORMAL with a `POS_SYS_SET_VALUES` request on the `PosIOctl()` subroutine call.

The table below shows the maximum number of bytes of data allowed on one `PosWrite()` call. If the number of bytes to be written exceeds this value, the subroutine returns the 318 POSERR\_SYS\_INVALID\_LENGTH error code and the data is not printed.

Table 12-9. Maximum Number of Characters for `PosWrite()`

Printer	Maximum Number of Characters
Model 2 Printer	1024
Model 3 Printer	1024
Model 4 Printer	1024
4689 Point of Sale Printer (3x1 Models)	8192
4689 Point of Sale Printer (except 3x1 Models)	1024

Table 12-9. Maximum Number of Characters for PosWrite() (continued)

Printer	Maximum Number of Characters
4610 SureMark Printer Models T11, T12, T13, T14, T15, TF6, TF7, TM6, TM7	16384

### Printing a Line of Text at the Printer

There are three print queues, one for each print station. If a print line cannot be printed immediately, it is queued in the appropriate station's queue. The application controls the sequence of printing by:

- Enabling and disabling DI station printing
- Setting the CR and SJ interleave value
- On the fiscal printer, enabling and disabling fiscal printing

**Enabling and disabling the DI station:** When the DI station is enabled, it is the only active station. Messages to the CR or SJ stations are queued, but are not printed. When the DI station is disabled, only the CR and SJ stations are active. Messages to the DI station are queued but are not printed.

**Note:** Enabling and disabling the DI station is not supported on the IBM 4689 Model 3x1 and TD5 printers.

**Interleaved printing:** The printer resource **PosNinterleaved**, when set to `PosINTERLEAVED_TRUE`, causes data for the CR or SJ stations to be printed in the order received from the application. When it is set to `PosINTERLEAVED_FALSE`, the printer device handler prints the lines in whatever order gives the best throughput.

**Note:** Interleaved printing is not available on the fiscal printer, the 4689 printers, and the IBM 4610 SureMark printers.

**Enabling and disabling fiscal printing:** When fiscal printing is enabled, all non-fiscal data is queued and fiscal commands are sent to the printer. When fiscal printing is disabled, all fiscal commands are queued and non-fiscal data is sent to the printer.

While fiscal printing is enabled, the application controls the fiscal printer. All commands for printing to the different stations are under the control of the application program. This includes registering and de-registering documents in the DI station.

#### Notes:

1. Enabling and disabling fiscal printing is available only on the fiscal printer.
2. Fiscal printing is always enabled for the IBM 4610 SureMark Point of Sale Fiscal USB printers. These printers do not support non-fiscal printing.

### Printer Errors

An error in any of the print stations causes printing to stop. After the problem is corrected, the *PosIOctl()* request `POS_PRN_RESUME_PRINTING` or `POS_PRN_RETRY_PRINTING` should be issued to restart printing. See "POSM\_PRN\_PRINTER\_ERROR" on page 20-11 for more information.

### Changing the Print Characteristics

The control characters and escape character sequences are not printed, but are used to change the print style or to control the operation of the printer. A list of the control characters and escape character sequences recognized by the printer

device handler follows.

Table 12-10. Printer Control Characters and Escape Character Sequences

Special characters	Meaning	Values
20 CPI Font	20 characters per inch	27 (0x1B), 59 (0x3E)
Advance paper to tear bar	Advance paper to tear bar	27 (0x1B), 79 (0x4F)
Carriage return	Move print position to the left margin	13 (0x0D)
Chase	Notify application of event completion	27 (0x1B), 71 (0x47)
Cut paper	Cut paper at the CR station	27 (0x1B), 80 (0x50)
Cut paper and stamp	Cut paper and stamp paper at the CR station	27 (0x1B), 81 (0x51)
Double-density	Start double density printing	27 (0x1B), 69 (0x45)
Double-high font	Set font to double-high	27 (0x1B), 23 (0x17)
Double-wide font	Set font to double-wide	27 (0x1B), 14 (0x0E)
Escape	Enter command mode	27 (0x1B)
Fix Font Width	Line up numbers when using proportional fonts	27 (0x1B), 63 (0x3F)
Flip check	Flip the check in the DI station	27 (0x1B), 53 (0x35)
Inline logo	Insert a logo into the print line	27 (0x1B), 115 (0x73)
Line feed	Feed one line	10 (0x0A)
Move to Tab	Print at the next tab stop	09 (0x09)
Normal density	Stop double density printing	27 (0x1B), 70 (0x46)
Normal font	Set font to 15 CPI (24x24 dot matrix on 4689-301)	27 (0x1B), 59 (0x3B)
Print Barcode	Print a barcode	27 (0x1B), 107 (0x6B)
Print PDF417 Barcode	PDF417 Barcode	27 (0x1B), 108 (0x6C), 32 (0x50)
Print Pre-defined Message	Print a pre-defined message	27 (0x1B), 112 (0x71)
Print Pre-defined Logo	Print a pre-defined logo	27 (0x1B), 113 (0x70)
Read MICR data	Read MICR data	27 (0x1B), 82 (0x52)
Register document	Feed in document at the DI station	27 (0x1B), 76 (0x4C)
Release document	Eject document from the DI station	27 (0x1B), 77 (0x4D)
Right column align	Align printing to the right column	27 (0x1B), 116 (0x74)
Scalable font	Print text larger than the default	27 (0x1B), 35 (0x23)
Select color printing	Full color printing	27 (0x1B), 34 (0x22)
Select PDF417 aspect ratio	Vary the ratios of width to height of the barcode	27 (0x1B), 108 (0x6C), 35 (0x53)
Select PDF417 ECC level	Specify security or error correction codewords	27 (0x1B), 108 (0x6C), 34 (0x52)
Select PDF417 truncation	Enable or Disable barcode truncation	27 (0x1B), 108 (0x6C), 36 (0x54)
Small font	Change to the small font (16x16 dot matrix on 4689-301). Set font to 17CPI	27 (0x1B), 60 (0x3C)
Spread font	Set font to 12 CPI	27 (0x1B), 58 (0x3A)

Table 12-10. Printer Control Characters and Escape Character Sequences (continued)

Special characters	Meaning	Values
Text attributes	Change the text print attributes	27 (0x1B), 33 (0x21)
User-defined fonts	Enable customized user fonts	27 (0x1B), 36 (0x24)

### Line Buffering

Except for the IBM 4610 SureMark printers and the IBM 4689 Model 3x1 and TD5 printers, the printer device handler internally buffers print data until one of the following occurs:

- A line feed control character is processed
- A carriage return control character is processed
- An escape character sequence is processed, which results in a change to the current print characteristics of the printer
- One of the *PosIOctl()* subroutine calls is issued, which causes one of the print characteristics of the printer to change
- The internal buffer of 128 bytes in the printer device handler is full

For the IBM 4610 SureMark Point of Sale printers and the IBM 4689 Model 3x1 and TD5 printers, the printer device handler sends the data to the printer when either the internal buffer is full or all the data on the write request is processed.

The printer device handler allows the application program to print up to 1024 bytes for the IBM Models 2, Model 3, and Model 4 printers; 16,384 bytes for the IBM 4610 SureMark Point of Sale printers; and 8,192 for the IBM 4689 Models 3x1 and TD5 printers with one *PosWrite()* subroutine call. It is not always necessary for the application to explicitly insert a carriage return and line feed into the data stream. If the data stream exceeds the maximum allowed for the station at the current font, the printer device handler (or the printer) wraps the data to the next line.

### Determining When Printing is Complete

If you have print lines queued and want them printed before your application continues, use the chase escape character sequence in the data stream. This character sequence is put in the print queue along with other ASCII characters, control characters, and escape character sequences. When the chase escape character sequence is encountered as the print lines are sent to the printer, the POSM\_PRN\_CHASE\_COMPLETE event message is put in the application input queue. Once the application receives the chase event message, it is assured that the printing of all data up to the chase escape character sequence has been completed.

### Font Interactions

Because the IBM Point of Sale Subsystem printer device handler allows the application program to combine all the different types of fonts on a single print line, there are some interactions between the different fonts you need to be aware of:

- Fonts with different widths on the same line can cause characters to be overwritten or can cause more than the expected number of blank spaces between characters to appear.
- Switching from double-high font to any other font size, or vice versa, on the same print line can cause the characters in the new font to be offset downward as compared to the double-high font characters.
- A chase escape character sequence within a print line at the double-high font size causes the characters to be offset downward as compared to the chase escape character sequence.

**Note:** Mixing different size fonts on the same print line is not recommended.

The IBM Model 2 and the IBM 4689-00x printers do not support changing fonts.

## Writing Data in Logo Mode

Logo printing can be done only at the CR and DI stations. If an application program attempts to write logo data to the SJ station, the 4904 POSERR\_PRN\_INVALID\_STATION error code is returned from the *PosWrite()* subroutine call. The logo data to be printed is specified in the *buffer* parameter of the *PosWrite()* subroutine call.

### IBM Model 2, Model 3, Model 4, and Model 4A Printers

On the IBM Model 2, the IBM Model 3, the IBM Model 4, and the IBM Model 4A printer, each byte of logo data controls the printing of one 8-dot column. The first byte in the buffer corresponds to the left-most print position. Bit zero (the least significant bit) of each byte corresponds to the top-most dot of each dot column. If a bit is set (equal to 1), the corresponding dot is printed.

On the IBM Model 3, the IBM Model 4, and the IBM Model 4A printer, the maximum amount of data per write statement printing is 380 bytes for the CR station or the narrow DI station and 880 bytes for the wide DI station. The maximum amount of data for logo printing on the IBM Model 2 printer is 300 bytes. This prints one line of data and does an automatic line feed and carriage return. If the data is less than the maximum, it is padded with blanks. If the data is more than the maximum, it is truncated.

### IBM 4610 SureMark Printer

On the IBM 4610 SureMark printers the first three bytes of data in the buffer indicate the following:

- Byte 0:** Dot Density. (0 - normal print, 1 - double-wide print, 2 - double-wide and double-high print)
- Byte 1:** Width. One-eighth the number of dots in the horizontal direction. For the CR station, this field has a valid range from 1 to 72. For the DI station, this field has a valid range from 1 to 59.
- Byte 2:** Height. One-eighth the number of dots in the vertical direction. For the CR station, this field has a valid range from 1 to 255. For the DI station, this field has a valid range from 1 to 5.

The actual logo data follows these three bytes. For the CR station, the logo data is interpreted as one-dot-high rows (horizontal slices), with bit zero of each byte corresponding to the right-most dot within that byte. For the DI station, the logo data is interpreted as eight-dot-high rows (vertical slices), with bit zero of each byte corresponding to the top-most dot within that byte.

The size of the logo data is limited to 3400 bytes for the 4610 T11 and 4610 T12 printers. For other printers in the 4610 printer family, the logo data is limited to 4000 bytes. If the print position is not at the first character position of the print line, logo printing forces a new line before printing the logo.

#### Notes:

1. Logo printing is not valid when the DI station is oriented for landscape printing.
2. An invalid value for the dot density (byte 0) will result in a 4914 POSERR\_PRN\_INCORRECT\_DATA\_FORMAT error.

- Exceeding the allowed logo data size will result in a 4914 POSERR\_PRN\_INCORRECT\_DATA\_FORMAT error.

### 4689 Models 3x1 and TD5 Printers

On the IBM 4689 Models 3x1 and TD5 Printers, there are two types of logo data that can be printed:

- Image character
- Sliced image

The image character logo format is used for printed characters that are not defined in the printer. The sliced image character logo format is used to print a barcode on the printer.

**Image character logo format:** The format for defining an image character logo is:

- Byte 0:** Escape character (0x1B)
- Byte 1:** Image character indicator (0x5E)
- Byte 2:** High byte of image length
- Byte 3:** Low byte of image length

Following byte 3 is the data for the character. The number of bytes specified in bytes 2 and 3 must be supplied. The format of the logo data is the same as the format used for defining a user-defined character. Multiple image character logo format commands can be printed at once.

**Sliced image logo format:** The format for defining a sliced image logo is:

- Byte 0:** Escape character (0x1B)
- Byte 1:** Sliced image indicator (0x6E)
- Byte 2:** Length of the data

Following byte 2 is the data for the sliced image. The number of bytes specified in byte 2 must be supplied. Only one line is defined. The printer will duplicate this line on each row of the current print line. Depending on the data supplied, a barcode can be created.

### Printer Errors

An error in any of the print stations causes printing to stop. After the problem is corrected, the *PosIOctl()* request POS\_PRN\_RESUME\_PRINTING or POS\_PRN\_RETRY\_PRINTING should be issued to restart printing.

## Writing Data in Download Message Mode (4610 SureMark Printers Only)

On the IBM 4610 SureMark Printers, predefined messages can be downloaded to the printer using PosMODE\_LOAD\_MESSAGE mode. Up to 25 messages (a maximum of 8K bytes) can be predefined. Once a message is downloaded, that message cannot be re-defined without erasing all the predefined messages already stored. All messages are erased by requesting to download message number zero.

**Note:** Predefined messages must use characters from the IBM 4610 Printer generic code page. See the *IBM 4610 SureMark Point of Sale Printer: User Guide*, GA27-4151 for details.

The data format for downloading messages (one per write request) is:

- Byte 0:** Message number. (Writing message number zero erases all currently defined messages in the printer.)

The message data follows the message number byte.

## Writing Data in Download Logo Mode

The data format for downloading a logo is dependent on the printer. The printers that support downloading logos are:

- IBM 4610 SureMark Point of Sale Printer models TI1, TI2, TI3, TI4, TI5, TF6, TF7, TM6, and TM7
- IBM 4689 Models 3x1 and TD5 Printers

### IBM 4610 SureMark Point of Sale Printers

On the IBM 4610 SureMark Point of Sale printers, predefined logos can be downloaded to the printer using PosMODE\_LOAD\_LOGO mode. Up to 40 logos (a maximum of 64K bytes) can be predefined. Once a logo is downloaded, that logo cannot be redefined without erasing all the predefined logos already stored. All logos are erased by requesting to download logo number zero.

The data format for downloading logos (one per write request) is:

- Byte 0:** Logo number. (Writing logo number zero erases all the currently defined logos in the printer).
- Byte 1:** Width. One-eighth the number of dots in the horizontal direction. For the CR station, this field has a valid range from 1 to 72. For the DI station, this field has a valid range from 1 to 59.
- Byte 2:** Height. One-eighth the number of dots in the vertical direction. For the CR station, this field has a valid range from 1 to 255. For the DI station, this field has a valid range from 1 to 5.

The actual logo data follows these bytes. For the CR station, the logo data is interpreted as one-dot-high rows (horizontal slices) with bit zero of each byte corresponding to the right-most dot within that byte. For the DI station, the logo data is interpreted as eight-dot-high rows (vertical slices) with bit zero of each byte corresponding to the top-most dot within that byte.

### IBM 4689 Models 3x1 and TD5 Printers

On the IBM 4689 Models 3x1 and TD5 Printers, only one predefined logo can be downloaded to the printer using PosMODE\_LOAD\_LOGO mode. This logo is used when the cut paper and stamp escape character sequence is sent to the printer. If a logo already exists in the printer, the 4915 POSERR\_PRN\_LOGO\_EXISTS error code is returned on the *PosWrite()* subroutine call. To define a new logo, the keypad on the printer must be used to erase the existing logo. Restart the IBM Point of Sale Subsystem, and then retry the command.

The format of the logo data is a fixed size of 7,632 bytes. There are 53 bytes that define a horizontal dot row and a total of 144 dot rows in a logo. Each 53 bytes in the write buffer defines one horizontal dot row. This will print a logo that is 52 mm horizontally and 18 mm vertically.

### Printer Errors

An error in any of the print stations causes printing to stop. After the problem is corrected, the *PosIOctl()* request POS\_PRN\_RESUME\_PRINTING or POS\_PRN\_RETRY\_PRINTING should be issued to restart printing. See "POSM\_PRN\_PRINTER\_ERROR" on page 20-11 for more information.

## Writing Data in Fiscal Mode (Fiscal Printer Only)

In fiscal mode, each *PosWrite()* subroutine call must contain an entire fiscal command. Fiscal commands cannot be split across multiple *PosWrite()* subroutine

calls. The fiscal printer device handler does not modify the fiscal commands. All fiscal commands provided by your application are sent to the fiscal printer “as-is”.

Full details of the fiscal commands are provided in the “IBM Fiscal Printer Hardware and Software Supplement” manual for each supported country.

**Note:** The fiscal escape character sequence (0x1B, 0x66 or ESC f) should **not** be inserted by the application into commands sent to the IBM Point of Sale Subsystem fiscal printer device handler. This character string is inserted by the device handler.

### Printing a Line of Text at the Printer

The application controls the sequence of printing fiscal commands by enabling and disabling fiscal printing. When fiscal printing is enabled, data for the CR, SJ, and DI stations is queued, but is not printed. When fiscal printing is disabled, the CR, SJ, and DI stations are active. Fiscal commands are queued but are not printed.

The IBM 4610 SureMark Point of Sale Fiscal USB printers are always set to fiscal-printing-enabled. They do not support non-fiscal printing.

### Printer Errors

A fiscal error or an error in any of the print stations causes printing to stop. After the problem is corrected, issue a `POS_PRN_RETRY_PRINTING PosIOctl()` request to start printing again. For more information, see “`POSM_PRN_PRINTER_ERROR`” on page 20-11.

If the `POS_PRN_RETRY_PRINTING PosIOctl()` request is not successful, the application must issue a `POS_PRN_DISCARD_DATA PosIOctl()` request to discard all the queued fiscal commands. The application must then issue a `POS_PRN_RESUME_PRINTING PosIOctl()` request to start printing again.

**Note:** Fiscal commands reprinted by the fiscal printer are identified by the pound or number sign(#) in the first three character positions of the reprinted line.

**Power-on Errors:** After the fiscal printer comes online and your application successfully opens and acquires the fiscal printer, issue a `POS_SYS_GET_VALUES PosIOctl()` request to get the value of the **PosNfiscalPLDStatus** resource. If this is set to `PosFISCAL_PLD_TRUE`, a fiscal command was being processed by the fiscal printer when the power to it was interrupted.

## Control Characters

The control characters determine how a line of text should be processed and are interpreted only when the **PosNprintMode** resource value is set to `PosMODE_NORMAL`. The following control characters are recognized by the printer device handler:

### Line Feed (0x0A)

The line feed control character causes the printer device handler to flush out any buffered data to the printer, advances the paper one line, and leaves the current print position the same. A subsequent write in normal mode to the printer continues at the current position one line below the previously printed line. On the IBM 4610 SureMark Point of Sale printers, a line feed control character resets the print position to the beginning of the next line.

**Carriage Return (0x0D)**

The carriage return control character causes the printer device handler to flush out any buffered data to the printer, and to move the current print position to the first position of the line. On the IBM 4610 SureMark Point of Sale printers the carriage return control character is ignored.

**Escape (0x1B)**

The escape control character indicates to the printer device handler that the next character is to be interpreted as a command for the printer. See "Escape Character Sequences" for more information.

**Note:** For print data alignment, it is recommended that the application terminate each print line with a carriage return and line feed.

## Escape Character Sequences

The escape character sequences are the commands recognized by the printer device handler to:

- Advance and cut CR station paper
- Change font size
- Change print density
- Control document registration
- Obtain print confirmation
- Read MICR data
- Change text attributes
- Print a barcode
- Flip a check
- Print a pre-defined message
- Print a pre-defined logo

Unrecognized commands following an escape control character result in both characters being discarded by the printer device handler. The escape character sequences are interpreted only when the **PosNprintMode** resource value is set to **PosMODE\_NORMAL**.

For each escape character sequence that applies to a specific station, the escape command is recognized only if that station is selected at the time of the *PosWrite()* subroutine call. Otherwise, the escape character sequence is discarded. See "PosNprintStation" on page 21-45 for more information about selecting stations.

The escape character sequences for specific printer stations are:

**CR Station**

- Advance Paper to Tear Bar
- Cut Paper
- Cut Paper and Stamp
- Print Barcode

**DI Station**

- Register Document
- Release Document
- Read MICR Data
- Flip Check

Under normal conditions on all printers except the IBM 4610 SureMark Point of Sale printers, all text is buffered by the printer device handler until a carriage return or line feed control character is encountered, or until the printer device handler's internal buffer is full. However, when an escape character sequence results in a

change to the current print characteristics of the printer, any buffered data is sent to the printer before the command is processed by the printer device handler. On the IBM 4610 SureMark Point of Sale printers, text is only sent to the printer when the device handlers internal buffer is full or when the processing of the current write request is complete.

**20 CPI Font (0x1B, 0x3E)**

Switches printing to 20 characters per inch (CPI) mode.

**Note:** Not supported in SurePoint Model TI1/TI2 printers.

**Advance Paper to Tear Bar (0x1B, 0x4F)**

The advance paper to tear bar escape character sequence advances the CR station paper forward until the last printed line is above the tear bar.

This command is recognized only if the CR station has been selected at the time of the *PosWrite()* subroutine call. Otherwise, the advance paper to tear bar escape character sequence is discarded.

**Chase (0x1B, 0x47)**

The chase escape character sequence indicates to the printer device handler to send a confirmation event message to the application program once the data has been successfully printed on the selected stations. A confirmation event message is generated for each selected station. If no stations are selected at the time of the *PosWrite()* subroutine call, the chase escape character sequence is discarded.

**Note:** Extensive use of the chase escape character sequence on the IBM 4610 SureMark Point of Sale printers can negatively impact the performance of printing.

**Cut Paper (0x1B, 0x50)**

The cut paper escape character sequence causes the paper at the CR station to be cut. See "Receipt Paper Cutter" on page 12-42 for more information.

This command is recognized only if the CR station has been selected at the time of the *PosWrite()* subroutine call. Otherwise, the cut paper escape character sequence is discarded.

**Notes:**

1. On the 4689-301, 3G1, 3M1, and TD5 printers, the paper cut will be a partial paper cut.
2. This escape character sequence will be ignored on the IBM Model 2 printer and the IBM 4689-00x printers.

**Cut Paper and Stamp (0x1B, 0x51)**

The cut paper and stamp escape character sequence causes the paper at the CR station to be cut, the paper to be fed several lines, and a stamp to be placed on the paper. See "Receipt Paper Cutter" on page 12-42 for more information.

**Notes:**

1. This command is recognized only on the IBM 4689 printers and only if the CR station has been selected at the time of the *PosWrite()* subroutine call. Otherwise, the cut paper and stamp escape character sequence is discarded.
2. On the 4689-301, 3G1, 3M1, and TD5 printers, the paper cut will be a full paper cut.

**Double Density (0x1B, 0x45)**

The double density escape character sequence results in the printer device handler printing the text twice on the printer for all subsequent writes to each station. If the current density is double density, the printer device handler discards this request.

**Note:** The double density escape character sequence is ignored on the IBM 4610 SureMark Point of Sale printers at the DI station when the orientation is landscape. However, when this escape character sequence is processed, all future data is printed in double density until the normal density escape sequence is processed.

**Double-High Font (0x1B, 0x17)**

The double-high font escape character sequence sets the printer to 7.5 CPI and the character height equal to 18 dot-rows for all subsequent writes to each station. If the current font is double-high font, the printer device handler discards this request.

**Notes:**

1. When scalable fonts are used with this escape character sequence enabled, the print is defaulted to scaling a normal size font.
2. The double-high font escape character sequence is not recognized on the IBM Model 2 printer and the IBM 4689 printers.
3. The double-high font escape character sequence is ignored on the IBM 4610 SureMark Point of Sale printers when the print station is set to the DI station and the DI station is in landscape orientation.

**Double-Wide Font (0x1B, 0x0E)**

The double-wide font escape character sequence sets the printer to 7.5 CPI for all subsequent writes to each station. If the current font is double-wide font, the printer device handler discards this request.

**Notes:**

1. When scalable fonts are used with this escape character sequence enabled, the print is defaulted to scaling a normal size font.
2. The double-wide font escape character sequence is not recognized on the IBM Model 2 printer and the IBM 4689 printers.

**Fix Font Width (0x1B, 0x3F, *n*)**

Fix the width of proportional fonts. Use this command to line up numbers.

**Notes:**

1. If the defined character is larger than the width that is defined, the right side of the character will be truncated when the width of the character is scaled using the scalable fonts; otherwise, the character will overlap the next character.
2. If the character is smaller than the defined width, the character will be centered in the space. (+ or - one dot).
3. If the width is set to zero, the fix width function is disabled and the characters are printed at their defined widths.
4. Only valid on proportional user-defined fonts (ignored on non-proportional fonts).
5. Not supported in T11/T12 printers.

**Flip Check (0x1B, 0x35)**

The flip check escape character sequence will flip the check inserted in the

DI station. This command is typically used before printing on the front of the check, after reading the MICR data, and franking the back of the check.

**Note:** The flip check escape character sequence is recognized only on the IBM 4610 SureMark Point of Sale printers, and only when the DI station has been selected at the time of the *PosWrite()* subroutine call.

**Inline Logo (0x1B, 0x73, *d*, *w1*, *w2*, *data*)**

This inline command inserts a logo in the print line, which embeds the logo into the print line.

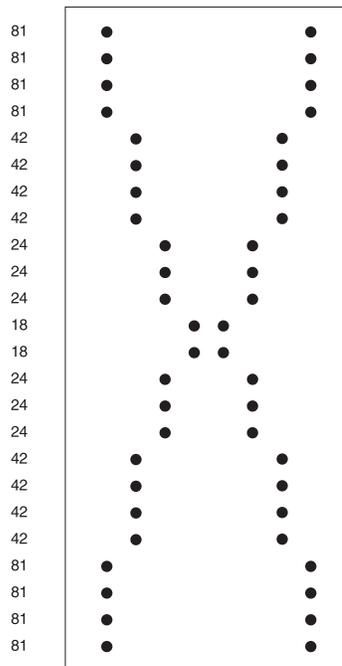
- d** Logo height:  
  - 00** single height printing = (24 dots high)
  - 01** double height printing = (48 dots high; however, it is scaled to this dot height by the hardware)
- w1, w2** Two-bytes that specify the width of the logo.
- data** Content of the logo (up to 65535 bytes)

**For Example:**

```

d w1w2 data
1B 73 00 0008 8181 8181 4242 4242 2424 2418 1824 2424
                4242 4242 8181 8181
    
```

- d** 00, which produces single height printing (24 dot height).
- w1w2** 0008, which indicates that the logo will be 8 dots wide.
- data** 81 81 81 . . . which are hexadecimal numbers (24 of them) that each represent lines of logo data. Each byte identifies the 8 dots that are on or off as shown in the sample output that follows:



**Notes:**

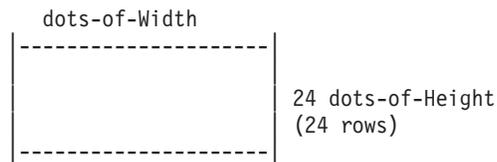
1. Logo data is represented from left to right. For a 16 dot (2-byte), logo the logo data is transmitted:

```

byte
1  2
3  4
5  6
7  8
.
.
.
47 48
    
```

2. The number of data bytes to form the image is calculated as follows:

$$\text{dots-of-width} / 8 * \text{dots-of-height}$$



**Note:** Dots-of-width / 8 must be a whole number (for example, 1.1 would be rounded up to 2).

Like user-defined characters, if the image is only 14 dots wide, the last two dots should be zero to complete the byte. When printed, the image will only be 14 dots wide.

3. See “User defined fonts” on page 12-31 and “Print pre-defined logo” on page 12-28 commands for image generation.
4. Use only in Thermal Stations.
5. If the data sent in PosWrite() for the inline logo command is too short, the logo and the following data will be truncated.
6. Not supported in T11/TI2 printers.

**Move to Tab (0x09)**

The move to tab control sequence moves printing to the next Tab stop.

Setting Tabs (see “PosNprintTabStops” on page 21-47) and Move to Tab are used with proportional space fonts to improve the format of the printed text.

**Notes:**

1. The tab control character is embedded in a print string.
2. Tabs are only valid when print alignment is set to the left.

**Normal Density (0x1B, 0x46)**

The default density for printer device handler is normal density. The normal density escape character sequence can be issued to return the printer to normal density after it has been modified by a Double Density escape character sequence. If the current density is normal density, the printer device handler discards this request.

**Note:** The normal density escape character sequence is not recognized on the IBM 4689 printers.

**Normal Font (0x1B, 0x3B)**

The default font for the printer device handler is normal font (15 CPI). Issue the normal font escape character sequence to return the font size to 15 CPI after it has been changed by any of these escape character sequences:

- Spread font
- Double-wide font
- Double-high font

If the current font is normal font, the printer device handler discards this request.

All subsequent writes to any station are in this font size until the font size is changed.

**Notes:**

1. On the 4610 TI5 printer in DBCS mode, this command turns OFF the Double-High and Double-Wide fonts.
2. On the IBM 4689 Models 3x1 and TD5 Printers, the normal font is the 24x24 dot matrix font.
3. The normal font escape character sequence is not recognized on the IBM Model 2 printer and the IBM 4689-00x printers.

**Print Barcode (0x1B, 0x6B, *type, width, height, HRI, HRI font, barcode digits*)**

The print barcode escape character sequence prints the *barcode digits*, which is an ASCII null terminated character string, as a bar code. The format of the data following the barcode escape character sequence is:

<b>Type</b>	The type of the barcode to be printed:
<b>00</b>	UPC-A
<b>01</b>	UPC-E
<b>02</b>	JAN13 (EAN)
<b>03</b>	JAN8 (EAN)
<b>04</b>	CODE39
<b>05</b>	ITF
<b>06</b>	CODABAR
<b>07</b>	CODE128 (c)
<b>08</b>	CODE93
<b>09</b>	CODE128 (a&b&c).

**Note:** Refer to the 4610 SureMark printer user's guide for additional information.

<b>Width</b>	Horizontal magnification of the line width (valid values are 2, 3, and 4)
<b>Height</b>	Dot height (valid values are between 1 and 255)
<b>HRI</b>	Print human readable information location:
<b>0</b>	None
<b>1</b>	Above the barcode
<b>2</b>	Below the barcode
<b>3</b>	Above and below the barcode
<b>HRI Font</b>	Human readable font size:
<b>0</b>	Normal font
<b>1</b>	Spread font

**Notes:**

1. If the current print position is not at the beginning of the print line, the print barcode escape character sequence will insert a new line character before printing the barcode.
2. If an invalid *type, width, height, HRI, or HRI font* is specified, the printer will default to using the last valid values used in a previous print barcode escape character sequence.

- The print barcode escape character sequence is recognized only on the CR station of the IBM 4610 SureMark Point of Sale printers.

The barcode escape sequence string can be terminated two ways:

- If the barcode string is terminated with a ASCII null (00x0), the length returned from the *PosWrite()* will be equal to the length sent.
- If the barcode string is not terminated with a ASCII null, one will be added and the length returned from the *PosWrite()* will be equal to the length sent plus 1.

#### **Print PDF417 Barcode: (0x1B, 0x6C, 0x50, *data*, 0x00)**

**Data** The information that is to be printed. The 0x00 at the end is the termination string of the barcode data.

The maximum number of bytes of barcode data that can be printed on the 4610:

- T11/T12: 250.
- T13/4/5: 1,000.
- SST/TF6, SST/TF7, SST/TM6, SST/TM7 printers: 1,000.

The **data** between the 0x50 marker and the 0x00 termination character is copied "as-is" from the application. No data transformations are applied to mapped code pages.

#### **Notes:**

- This command is only supported on the CR station. If another station is selected, this command will not be loaded.
- The POSS for Windows drivers will not check the valid size of the data.
- This command should be sent as a separate *PosWrite()*.

#### **Print Pre-defined Message (0x1B, 0x71, *message number*)**

The print pre-defined message escape character sequence prints the message number identified in the escape character sequence. If the message associated with *message number* was not previously stored in the printer, an error message will be sent to the application asynchronously.

**Note:** The print pre-defined message escape character sequence is recognized only on the IBM 4610 SureMark printers.

#### **Print Pre-defined logo (0x1B, 0x70, *font*, *logo number*)**

The print pre-defined logo escape character sequence prints the logo associated with *logo number* identified in the escape character sequence. If the logo number was not previously stored in the printer, an error message will be sent to the application asynchronously. In addition to the logo number, the desired font size must be specified. The font value to use is the same as the first byte in the regular logo printing command. For more information, see *Writing Data in Logo Mode*. If the current print position is not at the beginning of the print line, the print pre-defined logo escape character sequence will insert a new line character before printing the logo.

**Note:** The print pre-defined message escape character sequence is recognized only on the IBM 4610 SureMark Point of Sale printers on the CR station and on the DI station in portrait orientation.

#### **Read MICR Data (0x1B, 0x52)**

The read MICR data escape character sequence causes the printer device handler to read the account information on the check inserted in the DI station.



5	Width = 6X	5	Height = 6X
6	Width = 7X	6	Height = 7X
7	Width = 8X	7	Height = 8X

For example:

0x47 would produce a character width that is 5 times normal size and a character height that is 8 times normal size.

You can also use the **text attribute** escape sequences to select the character size (double high and wide). The command that is received last is the command that is used.

Characters are formed by expanding the character width proportionally in the printer.

The largest user-defined font is 32 dots high. Therefore, the largest character that can be printed is 32x8 dots high, or 256 dots high - 1.25 inches.

**Notes:**

1. This command is only valid on the Thermal Print Station.
2. Not supported on 4610 TI1/TI2 printers

**Select Color Printing (0x1B, 0x22, nn)**

Enabling and disabling of full character color is available within a print line.

**nn** The choices are:

- 00** Cancel Color Printing
- 01** Enable Full Character Color Printing
- 02** Enable Half Character Color Printing
- > 02** Reserved, do not use

**Notes:**

1. There is a limit of 8 enables and disables per line when full character color printing is used. (More than 8 will result in unpredictable behavior.)
2. Half character color will only be supported at the beginning of a print line.
3. This command is only valid if the enable color printing command has been issued prior to issuing this command enabling color printing .
4. Not supported on 4610 TI1/TI2 printers.

**Select PDF417 Aspect Ratio: (0x1B, 0x6C, 0x53, h, w)**

The select PDF417 aspect ratio enables you to vary the ratio of the width to the height. This enables the bar code to fit in wider or higher spaces.

- **h** is the height dimension for the ratio (default : 1 (0x31))
- **w** is the width dimension for the ratio (default : 2 (0x32))

The Range of both **h** and **w** is between 1 (0x31) and 9 (0x39) inclusively.

**Note:** If an invalid value is sent (for **h** or **w**), it will be ignored and the current value is retained.

**Select PDF417 ECC Level: (0x1B, 0x6C, 0x52, n1, n2)**

Select PDF417 ECC Level allows you to specify security or error correction codewords for error recovery. This error correction enables for the correction of missing or codewords that cannot be decoded, and incorrectly decoded codewords.

**n1** is the high order byte ECC level (default: 0x00 (0))

**n2** is the low order byte ECC level (default: 0x0F (15))

Range: 0 <= n1,n2 <= 400 (0x190)

**Note:** If an invalid value is sent (for *n1* or *n2*), it will be ignored and the current value is retained.

**Select PDF417 Truncation: (0x1B, 0x6C, 0x54, t)**

**t** is '1' (0x31) to enable truncation and all other values to disable truncation. The default is '0' (0x30)

**Small Font (0x1B, 0x3C)**

The small font escape character sequence sets the printer to smaller font defined in the printer and remains in affect for all subsequent writes to each station until changed. For the IBM 4610 SureMark Point of Sale printers, the font will be set to 17 CPI. For the IBM 4689-301, 3G1, 3M1, and TD5 printers, the font will be set to the 16x16 dot matrix font. If the current font is the small font, the printer device handler discards this request.

The small font is supported only when the Model 4 printer is at EC level 0x23 and the Model 4R printer is at EC level 0x3B or 0x3C. If the printer is at an unsupported EC level, the request will be ignored, the return code will be set to -1 and the error will be set to POSERR\_SYS\_FUNCTION\_NOT\_SUPPORTED

**Note:** The small font escape character sequence is recognized only on the IBM 4610 SureMark Point of Sale printers, the IBM 4689-301, 3G1, 3M1, and TD5 printers and model 4 and 4R printers at a supported EC level. Small font is not supported on the 4610 TI5 printer in DBCS mode, the Model 4A , or Model 3F Fiscal printer.

**Spread Font (0x1B, 0x3A)**

The spread font escape character sequence sets the printer to 12 CPI for all subsequent writes to each station. If the current font is spread font, the printer device handler discards this request.

**Note:** The spread font escape character sequence is not recognized on the IBM Model 2 printer and the 4610 TI5 printer in DBCS mode.

**User-defined Fonts (0x1B, 0x24 nn)**

The User-defined fonts escape character sequence enables (previously installed) customized/user-defined fonts to be used for printing.

**nn** The choices are:

- 00** Disable User-defined fonts
- 01** Select Character Set 1
- 02** Select Character Set 2
- 03** Select Character Set 3 (ignored on the DI station)
- 04** Select Character Set 4 (ignored on the DI station)

**Notes:**

1. The User-defined font must be downloaded to the printer before the device is Opened.
2. User-defined fonts also include Proportional Fonts. Proportional Fonts take up 2 character sets; therefore, they are valid only at memory location 1 or 3 of the CR stations.
3. User-defined and "standard" fonts may be used on the same line. Most likely the descriptions will be in User-defined fonts and amounts will be IBM resident font sets.

- Spread fonts, normal fonts, small fonts and 20 CPI fonts are not supported with User-defined fonts on the CR station (or DI station). When the above fonts are enabled and you print from the User-defined character sets, the font size does not change, but the character spacing may change.

**Text Attributes (0x1B, 0x21, attribute code)**

The text attributes escape character sequence enables and disables various attributes of text character printing. The IBM 4610 SureMark Point of Sale printers do not have an SJ station, and the IBM 4689-301, 3G1, 3M1, and TD5 printers do not have a DI station. Therefore, in the table that follows, these stations should be ignored on these printers. The following table defines each supported attribute that can be changed, and the *attribute code* used to enable or disable it:

Table 12-11. Text Attribute enable and disable codes for 4610 models and 4689 Model 3x1 and TD5 printers

Attribute	Enable Code	Disable Code	Supported in CR Station	Supported in SJ Station	Supported in DI Station	Notes
Double-high	0x01	0x81	yes	yes	portrait only	
Double-wide	0x02	0x82	yes	yes	yes	
Underlined	0x03	0x83	yes	n/a	no	
Overlined	0x04	0x84	yes	yes	no	Note 1
Inverted	0x05	0x85	yes	n/a	no	Note 1
Rotated	0x06	0x86	yes	no	no	Note 2
Dashed Lines	0x07	0x87	yes	yes	n/a	Note 3,4
Dark Contrast	0x08	0x88	yes	yes	n/a	Note 3,5
Line Characters	0x09	0x89	yes	yes	n/a	Note 3
Contrast On	0x0A	0x8A	yes	yes	n/a	Note 3

**Notes:**

- Not supported on the IBM 4689-301, 3G1, 3M1, and TD5 printers.
- On the IBM 4689-301, 3G1, 3M1, and TD5 printers, the rotated text attribute results in characters being printed in Tategaki mode.
- Not supported on the IBM 4610 SureMark Point of Sale printers.
- The default line type is solid lines.
- The default contrast is light contrast.

These attributes apply to the current font selected. The double-high and double-wide attributes should not be confused with the double-high font and double-wide font escape character sequences, which set the font to be either double-high or double-wide of the normal (15 CPI) font.

The *line characters* text attribute turns on the printing of line characters on the IBM 4689-301, 3G1, 3M1, and TD5 printers. Use the following character mappings to print a line character:

Code (Hex)	Printed Line Character
0x01	(⌈) Upper-left corner

<b>0x02</b>	(┌) Upper-right corner
<b>0x03</b>	(└) Lower-left corner
<b>0x04</b>	(┐) Lower-right corner
<b>0x05</b>	( ) Vertical line
<b>0x06</b>	(-) Horizontal line
<b>0x10</b>	(+) Center cross
<b>0x15</b>	(⊥) Bottom junction
<b>0x16</b>	(⊤) Top junction
<b>0x17</b>	(⊖) Right junction
<b>0x19</b>	(⊕) Left junction

## Printer Input/Output Control Requests (IOCtl)

The printer input/output control (IOCtl) requests enable an application to:

### Disable DI Station Printing

After your application is finished printing on a document in the DI station, it should disable printing to the DI station. This allows any data that is queued for either the CR station or the SJ station to be printed. While the DI station printing is disabled, all data for the DI station is queued.

To disable printing at the DI station, issue a `POS_PRN_DISABLE_DI_PRINTING PosIOCtl()` request.

#### Notes:

1. For the IBM 4610 SureMark Point of Sale printers, it is recommended that before disabling DI station printing, the print lines to the DI station are completed by using the chase escape character sequence. Otherwise, print lines (including lines that set print characteristics, such as setting the font size), may be printed out-of-order.
2. Not recognized on the IBM 4689-301, 3G1, 3M1, and TD5 printers.
3. Not recognized on the IBM 4610 SureMark Point of Sale Fiscal printers.

### Disable Fiscal Printing

Fiscal printing must be disabled for data to be printed on the CR, DI, and SJ stations. This allows any data that is queued for the CR, DI, and SJ stations to be printed. While fiscal printing is disabled, all fiscal commands are queued.

To disable fiscal printing, issue a `POS_PRN_DISABLE_FISCAL_PRINTING PosIOCtl()` request.

**Note:** This request is recognized only by the fiscal printers. Fiscal printing is always enabled and cannot be disabled for the USB versions of the IBM 4610 SureMark Point of Sale Fiscal printers.

### Discard Printer Data

Use the `POS_PRN_DISCARD_DATA PosIOCtl()` request to discard all queued printer data and fiscal commands after an error occurs.

### Enable DI Station Printing

For data to be printed on a document, printing must be enabled at the DI station by using a `POS_PRN_ENABLE_DI_PRINTING PosIOCtl()` request. Once this request is processed, any data for either the CR station or the SJ station is queued. Any data previously queued for the DI station is now printed.

**Notes:**

1. For the IBM 4610 SureMark Point of Sale printers, it is recommended that before enabling DI station printing, the print lines to the CR station are completed by using the chase escape character sequence. Otherwise, print lines (including lines that set print characteristics, such as setting the font size) may be printed out-of-order.
2. Not recognized on the IBM 4689-301, 3G1, 3M1, and TD5 printers.

**Enable Fiscal Printing**

For fiscal commands to be processed, fiscal printing must be enabled by using a `POS_PRN_ENABLE_FISCAL_PRINTING PosIOctl()` request. Once this request is processed, any data for the CR, DI, and SJ stations is queued. Any fiscal commands previously queued are now processed.

**Note:** This request is recognized only by the fiscal printers. Fiscal printing is always enabled and cannot be disabled for the USB versions of the IBM 4610 SureMark Point of Sale Fiscal printers.

**Hold Printing**

Use the `POS_PRN_HOLD_PRINTING PosIOctl()` request to halt printing temporarily.

**Release Printing**

Use the `POS_PRN_RELEASE_PRINTING PosIOctl()` request to resume printing.

**Reset Printer**

Use the `POS_PRN_RESET_PRINTER PosIOctl()` request to issue a power-on reset command to the printer.

**Resume Printing After an Error**

Use the `POS_PRN_RESUME_PRINTING PosIOctl()` request to restart printing after an error has occurred. This request starts printing without trying to reprint the error line. Some lines could be discarded as a result of this request. If the printer device handler detects that the error occurred before any print lines were printed, those lines will not be discarded.

The application program can provide an overlay string that will be printed before any printing is restarted. See “PosNresumeString” on page 21-50 for more information.

**Retry Printing After an Error**

Use the `POS_PRN_RETRY_PRINTING PosIOctl()` request to restart printing after an error has occurred. This request starts printing at the line the error occurred on.

The application program can provide an overlay string that will be printed before any printing is restarted. See “PosNretryString” on page 21-52 for more information.

For the 4610 printer, only the first character from the overlay string will be used. This character is only printed if the retry is for a print line that had a home error. In all other instances, the print line will print without a retry overly character in the first position.

**Setting Printer Resource Values**

When your application issues a `POS_SYS_SET_VALUES` request, the specified resources affect all `PosWrite()` subroutines issued after the request. The new values do not affect queued print operations resulting

from *PosWrite()* subroutines that were issued before the `POS_SYS_SET_VALUES PosIOctl()` request.

### Getting Printer Resource Values

Your application can issue a `POS_SYS_GET_VALUES` request to get the current value of any of the printer's resources. See "Printer Resources" for more information about the printer resources.

## Printer Resources

The printer resources control the operation of the printer device handler. These resources can be set with the `POS_SYS_SET_VALUES PosIOctl()` request or can be queried with the `POS_SYS_GET_VALUES PosIOctl()` requests. See "PosPrinter Resource Set" on page 21-32 for a list of resources.

## Printer Event Messages

Event messages are sent by the printer to the application. For a description of the POSS event messages, see "Chapter 20. Event Messages" on page 20-1. Printer-specific events are named `POSM-PRN-xxx`.

## Determining the Printer Status

Use the `POS_SYS_GET_VALUES PosIOctl()` request for the **PosNprintStatus** resource to determine printer status information. This function returns the current settings of the DI station control bits set by the `POS_SYS_SET_VALUES PosIOctl()` request and other printer status information. The printer status includes the following information:

- Paper status for the SJ station (out of paper or paper jam)
- Document insertion status (present or not present, top or front insert)
- Document station registered direction (from front or from top)
- Document insert station status (document ready or not ready)
- Printer head status (parked or not parked)
- Printer cover status (open or closed)
- MICR reader installed
- Whether the printer is online

Use the `POS_SYS_GET_VALUES PosIOctl()` request for the **PosNprintStatus2** resource to determine status information for the IBM 4610 SureMark Point of Sale Printers and the 4689-301, 3G1, 3M1, and TD5 printers.

**Note:** If the cover is opened while printing is in progress, the line in progress is completed. However, no additional printing can occur until the cover is closed. If an error occurred because of an open cover, the application program must issue either a `POS_PRN_RESUME_PRINTING` subroutine call or a `POS_PRN_RETRY_PRINTING PosIOctl()` subroutine call.

## Printer Queues

The SJ station and the CR station are considered blocked when a document is being printed. Simultaneous printing on the document and either the CR station or the SJ station is not permitted.

The `POS_PRN_ENABLE_DI_PRINTING PosIOctl()` and the `POS_PRN_DISABLE_DI_PRINTING PosIOctl()` requests are used to control the interactions between the DI station and the CR and SJ stations. To minimize the

impact to the application program, the printer device handler queues all data written to the CR and SJ station when the DI station is enabled, and queues all DI station data when the DI station is disabled.

The application program can control when the buffered data is printed by using the `POS_PRN_ENABLE_DI_PRINTING PosIOctl()` and the `POS_PRN_DISABLE_DI_PRINTING PosIOctl()` requests. The printer driver begins printing the buffered CR and SJ data as soon as the DI station becomes disabled. Therefore, the application should ensure that the user has completely removed the document from the printer before it disables the DI station.

**Note:** For the IBM 4610 SureMark printers, it is recommended that before enabling or disabling DI station printing, the application should ensure that the print lines to the current print station are completed by using the chase escape character sequence. Otherwise, print lines (including lines that set print characteristics, such as setting the font size), may be printed out-of-order.

## Document Insert Station

The DI station allows the insertion of forms, documents, or checks into the printer.

When a document is inserted into the DI station, a sensor is activated as the user inserts the document. Any change in sensor values causes an event message to be sent to the application.

The `POS_SYS_GET_VALUES PosIOctl()` request can be used by an application program to determine the presence of a document. The **PosNprintStatus** resource indicates the following conditions:

### **PosSTATUS\_DOCUMENT\_AT\_TOP**

Indicates when a document has been inserted in the top of the DI station or when a form has been inserted from the front far enough to reach the top sensor.

### **PosSTATUS\_DOCUMENT\_AT\_FRONT**

Indicates when a document has been inserted in the front of the DI station or when a form has been inserted from the top far enough to reach the front sensor.

### **PosSTATUS\_DOCUMENT\_READY**

Indicates that a document has been registered, either manually or automatically (under program control). The document must be registered before printing can begin at the DI station. Both `PosSTATUS_DOCUMENT_AT_TOP` and `PosSTATUS_DOCUMENT_AT_FRONT` can be true even though `PosSTATUS_DOCUMENT_READY` is false. This occurs when the document has been fed in by several line feed commands rather than with the register document control character.

### **PosINSERTED\_FORWARD**

Indicates from which direction a document was registered (front or top). Based on this determination, the application can order the print lines accordingly. For example, either printing the lines in reverse order for top inserted forms or standard order for forms inserted from the front.

## **IBM Model 2 Printer**

The DI station allows the insertion of forms, documents, or checks into the printer either from the front or from the side. The following functions are provided to allow application programs to work with the DI station:

- Front or side loading of documents
- Manual or automatic insertion of documents
- Document reinsertion
- Releasing the document

**Front or Side Loaded Documents:** Documents can be inserted from the front or side. A sensor is activated as the user inserts the document to the positive stop feed rollers.

**Note:** The Model 2 printer will always indicate that a document was inserted from the front (PosSTATUS\_DOCUMENT\_AT\_FRONT) and that it was inserted forward (PosSTATUS\_INSERTED\_FORWARD).

**Manual or Automatic Document Insertion:** Manual insertion is when the document is inserted to the feed rollers, and the station is activated using the DI station ready button on the top of the printer. Pressing the DI station ready button causes the document to be registered. If the document is inserted from the front, the document must be advanced forward approximately one inch before the first printable position is in the print field. The printer line feed buttons can be used to position the document, or the document can be advanced forward by the application.

Automatic insertion is when the document is inserted to the feed rollers, the DI station is enabled, the application writes a register document escape character sequence, and the document is positioned to the first print position by the application program.

The document can be advanced additional spaces by the application prior to printing the first line by issuing a *PosWrite()* subroutine with the desired number of line feed characters.

If a document is inserted from the side, it blocks the CR station when the form is stopped against the feed rollers. The user must not insert a form from the side until printing at the CR station is completed. Documents inserted from the front do not block the CR station until registered at the DI station using the DI ready button or a register document escape character sequence.

**Reinserting Documents for Printing:** Some transactions require that a document be inserted into the printer many times for printing at a different location each time, for example, documenting exception conditions or voiding transactions. These documents can be positioned for printing using the automatic method described in "Manual or Automatic Document Insertion", or they can be positioned using a manual method.

**Releasing the Document:** For removal of documents after printing, use the release document escape character sequence. When the printer receives this escape character sequence, the document is released from the printer and can be removed. The POS\_PRN\_DISABLE\_DI\_PRINTING *PosIOctl()* request should be sent after all document sensors are clear to prevent further DI station printing and to allow CR or SJ printing.

### **IBM Model 3, Model 3F, Model 3R, Model 4, Model 4A, and Model 4R Printers**

The DI station allows the insertion of forms, documents, or checks into the printer either from the top or from the front. The following functions are provided to allow application programs to work with the DI station:

- Top or front loading of documents

- Manual or automatic insertion of documents
- Document reinsertion
- Releasing the document
- Positioning the print head
- Reversible DI station motor
- Various DI station line lengths

**Top or Front Loaded Documents:** Documents can be inserted from the top or the front. A sensor is activated as the user inserts the document to the positive stop feed rollers and an LED on the front of the printer lights when the sensor is activated.

**Manual or Automatic Document Insertion:** Manual insertion is when the document is inserted to the feed rollers, and the station is activated using the DI station ready button on the front of the printer. Pressing the DI station ready button causes the document to be registered (fed into the printer until the first printable position is in the print field). If the document is inserted from the top, the first printable position is the bottom-most line of the document. If the document is inserted from the front, the first printable position is the top-most line of the document. If the user wants to start printing in a position other than the top or bottom of the form, the printer line feed buttons can be used to further position the document, or the document can be advanced (forward or backward) by the application.

Automatic insertion is when the document is inserted to the feed rollers, the DI station is enabled, and the document is positioned to the first print position when the application writes a register document control character.

For automatic registration, the first print position (top-most line or bottom-most line) is not determined by the top or front insertion, but by the value of the resource **PosNfeedDirection**. That is, if the direction is forward, (from front to top) the document is registered with the top line as the first print position, even though the document was inserted from the top.

Because users can insert documents from either the top or the front, mistakes can be made even if a message is displayed prompting the user on the method of insertion. The printer is designed to automatically interpret the intended insertion direction and allow the application program to correct for user errors.

The document is automatically positioned at the top of the document if an application program is designed for automatic front insertion of documents. That is, **PosNfeedDirection** is PosFEED\_FORWARD. If, however, the user inserts the document from the top, the printer automatically advances the document through the printer until the document is positioned at the top of the document. Because the application was designed for front insertion, the result of inserting the document from the top is exactly the same as inserting from the front.

The opposite insertion method is also automatic. If an application is designed for automatic top document insertion and the document is placed in the front opening, the printer automatically advances the form through the printer, positioned the document at the bottom of the form.

The document can be advanced additional spaces by the application prior to printing the first line by issuing a *PosWrite()* subroutine with the desired number of line feed characters.

If a document is inserted from the top, it blocks the CR and SJ stations when the form is stopped against the feed rollers. The user must not insert a form from the top until printing at both of these stations is completed. Documents inserted from the front do not block the CR or SJ stations until registered at the DI station using the DI ready button or a register document escape character sequence.

**Reinserting Documents for Printing:** Some transactions require that a document be inserted into the printer many times for printing at a different location each time, for example, documenting exception conditions or voiding transactions. These documents can be positioned for printing using the automatic method described in “Manual or Automatic Document Insertion” on page 12-38, or they can be positioned using a manual method. The steps required for reinserting a document using the manual method are shown in the following section:

1. Insert the document to the feed rollers and press the printer line advance button until the desired print location is positioned just above the printer cover.
2. Press the DI station ready button to reverse feed the document into the printer. The document is now correctly positioned for printing and the application can issue *PosWrite()* subroutines to the DI station.

**Releasing the Document:** The DI station of the printer holds documents tightly in place, preventing manual repositioning. This allows for more accurate positioning when advancing a document under program control, but it prevents a user from pulling a document out of the printer after printing has completed.

For fast removal of documents after printing, use the release document escape character sequence. When the printer receives this escape character sequence, the document is fed rapidly out of the printer until it is free from the document feed rollers. The direction the document is advanced out of the printer is controlled by the **PosNfeedDirection** resource. The `POS_PRN_DISABLE_DI_PRINTING` *PosIOctl()* request should be sent after all document sensors are clear to prevent further DI station printing and to allow CR or SJ printing.

**Positioning the Print Head:** Some forms are inserted easier with the print head positioned at the left home position. Other forms can be inserted easier with the head at center home position. When there is no more data to print, the head parks at one of these two positions depending on the **PosNheadParkedPosition** resource. Testing your application with the different print head locations is highly recommended.

**Reversible Document Station Motor:** The default document feed direction is the same as the one used to feed the receipt paper during printing (front to top). The document feed direction is determined by the **PosNfeedDirection** resource.

**DI Station Character Line Lengths:** The printer device handler can print either on the full width of the document or only on the right half. If **PosNDIWidth** is set to `PosDI_WIDE`, the first character position is near the left margin and there are 86 characters per line at 15 CPI. If only the left half is desired, use the new line and carriage return control characters imbedded in the data. If **PosNDIWidth** is set to `PosDI_NARROW`, the first character is near center home position and there are 38 characters per line at 15 CPI.

### **IBM 4689-001 and IBM 4689-002**

The DI station allows the insertion of forms, documents, or checks into the printer either from the top or from the front.

There are a maximum of 30 print lines for narrow paper printing and the first print position is 32.5 mm from the top of the form. For wide paper, the maximum number of print lines is 16. If the length of the wide paper is shorter than 117 mm, the first print position is set to 32.5 mm from the top of the form. Otherwise, the first print position is set to 84.5 mm from the bottom of the form.

The following functions are provided to allow application programs to work with the DI station:

- Top or front loading of documents
- Narrow document insertion
- Wide document insertion
- Releasing the document

**Top or Front Loaded Documents:** Narrow paper (slips) must be inserted from the front and wide paper (validation forms) must be inserted from the top. A sensor is activated as the user inserts the document to the positive stop feed rollers and an LED on the front of the printer lights when the sensor is activated. Any change in sensor values causes an event message to be sent to the application.

**Narrow Document Insertion:** Narrow paper must be inserted in the front of the DI station with the top edge of the form inserted first. After printing is complete, the document is ejected out the top side slot of the DI station. If the narrow paper is inserted in the top of the DI station, the printer will print in the wrong position on the narrow paper without any errors.

The recommended sequence for printing on narrow (slip) paper is:

1. The application enables DI station printing.
2. The application issues the register document escape character sequence to position the first print position 32.5 mm from the top of the form.
3. The document is inserted in the front of the DI station.
4. The application receives the event message indicating that a document has been inserted in the front.
5. The application writes up to 30 print lines.
6. The application issues the release document escape character sequence to eject the document out of the top side slot of the DI station.
7. The application disables DI station printing.

**Wide Document Insertion:** Wide paper must be inserted in the top of the DI station with the bottom edge of the form inserted first. After printing is complete, the document is ejected out the top side slot of the DI station.

The recommended sequence for printing on wide (slip) paper is:

1. The application enables DI station printing.
2. The application issues the register document escape character sequence to position the first print position 32.5 mm from the top of the form or 84.5 mm from the bottom of form.
3. The document is inserted in the top of the DI station.
4. The application receives the event message indicating that a document has been inserted.
5. The application writes up to 16 print lines.
6. The application issues the release document escape character sequence to eject the document out of the top side slot of the DI station.
7. The application disables DI station printing.

**Releasing the Document:** The DI station of the IBM 4689-00x printers holds documents tightly in place, preventing manual repositioning. This allows for more accurate positioning when advancing a document under program control, but it prevents a user from pulling a document out of the printer after printing has completed.

For fast removal of documents after printing, use the release document escape character sequence. When the printer receives this escape character sequence, the document is fed rapidly out of the printer until it is free from the document feed rollers.

### **IBM 4610 SureMark Point of Sale Printers**

The DI station allows the insertion of forms, documents, or checks into the printer either from the front or from the side. The following functions are provided to allow application programs to work with the DI station:

- Front or side loading of documents
- Manual or automatic insertion of documents
- Document reinsertion
- Releasing the document
- Various line lengths
- Portrait or landscape print orientation
- Flipping a check

**Front or Side Loaded Documents:** Documents can be inserted from the front or side. A sensor is activated as the user inserts the document to the positive stop feed rollers.

**Note:** The IBM 4610 SureMark Point of Sale printers always indicate that a document was inserted forward (PosSTATUS\_INSERTED\_FORWARD).

**Manual or Automatic Document Insertion:** Manual insertion is when the document is inserted to the feed rollers, and the station is activated using the DI station ready button on the top of the printer. Pressing the DI station ready button causes the document to be registered. If the document is inserted from the front, the document must be advanced forward approximately 1.5 inches before the first printable position is in the print field. The printer line feed buttons can be used to position the document, or the document can be advanced forward by the application.

Automatic insertion is when the document is inserted to the feed rollers, the DI station is enabled, the application writes a register document escape character sequence, and the document is positioned to the first print position by the application program.

The document can be advanced additional spaces by the application prior to printing the first line by issuing a *PosWrite()* subroutine with the desired number of line feed characters.

**Note:** For the IBM 4610 SureMark Point of Sale printers, documents can be inserted while printing on the CR station.

**Reinserting Documents for Printing:** Some transactions require that a document be inserted into the printer many times for printing at a different location each time, for example, documenting exception conditions or voiding transactions. These

documents can be positioned for printing using the automatic method described in “Manual or Automatic Document Insertion” on page 12-41, or they can be positioned using a manual method.

**Releasing the Document:** For removal of documents after printing, use the release document escape character sequence. When the printer receives this escape character sequence, the document is released from the printer and can be removed. The `POS_PRN_DISABLE_DI_PRINTING Pos/Oct()` request should be sent after all document sensors are clear to prevent further DI station printing and to allow CR printing.

**DI Station Character Line Lengths:** For printing in the portrait orientation, the printer device handler can print either on the full width of the DI station or only on a portion of the right side. If **PosNDIWidth** is set to `PosDI_WIDE`, there are 47 character positions per line at 15 CPI. If **PosNDIWidth** is set to `PosDI_NARROW`, there are 38 character positions per line at 15 CPI.

**Portrait or Landscape Oriented Printing:** Normal horizontal printing in the DI station is accomplished by setting the **PosNdiOrientation** resource to `PosPRINT_PORTRAIT`. When printing more than 47 characters per line, (as on the front of a check), set the **PosNdiOrientation** resource to `PosPRINT_LANDSCAPE`. When in landscape oriented printing, the document will be printed at 90 degrees counter-clockwise and print lines will be printed from the bottom of the document to the top of the document.

**Flipping a Check:** Both sides of a check can be printed without the operator manually removing the check from the printer. To accomplish this, send the Check Flip escape character sequence when it is time to flip the check from one side to the other. For example, the check could be franked on the back, the check flip escape character sequence is sent to the printer, and then the front of the check can be printed.

## Receipt Paper Cutter

The following printers contain a receipt paper cutter:

- IBM Model 3
- IBM Model 3F
- IBM Model 3R
- IBM Model 4
- IBM Model 4A
- IBM Model 4R
- IBM 4689-00x
- IBM 4689-301, 3G1, 3M1, and TD5
- IBM 4610 SureMark Point of Sale Printer Model TI1
- IBM 4610 SureMark Point of Sale Printer Model TI2
- IBM 4610 SureMark Point of Sale Printer Model TI3
- IBM 4610 SureMark Point of Sale Printer Model TI4
- IBM 4610 SureMark Point of Sale Printer Model TI5
- IBM 4610 SureMark Point of Sale Printer Model TF6
- IBM 4610 SureMark Point of Sale Printer Model TF7
- IBM 4610 SureMark Point of Sale Printer Model TM6
- IBM 4610 SureMark Point of Sale Printer Model TM7

The IBM Point of Sale Subsystem application program interface provides an escape character sequence for a partial cut of the paper, and for the IBM 4689 printers, an escape character sequence for stamping and cutting the paper. Because the tear

bar, on the printers with the receipt paper cutter, is only intended for occasional use (for example, when tearing off the excess paper after paper loading), applications should use the cutter at all times.

The application must advance the paper at the end of the transaction so that the last line clears the cutter. You can advance the paper in the following ways:

1. Write the advance paper to tear bar escape character sequence.
2. Write multiple line feeds.

Ten line feeds are needed to clear the tear bar when the CR station line spacing is set to 6 lines per inch. Twelve line feeds are required when the line spacing is set to 8 lines per inch.

For the IBM 4689-301, 3G1, 3M1, and TD5 printers, the logo that is stamped on a cut paper, and stamp escape character sequence, is logo that has been downloaded to the printer. For more information on downloading the logo, see "Writing Data in Download Logo Mode" on page 12-20.

## Printing Checks

Checks can be printed in normal mode in the DI station of the IBM Model 2 printer, the IBM Model 3 printer, the IBM Model 4 printer, the IBM Model 3F Fiscal printer, and the IBM 4610 SureMark Point of Sale printers. Checks can be inserted horizontally for printing on all printers except the IBM 4610 SureMark Point of Sale printers. For the IBM 4610 SureMark Point of Sale printers, the front of a check can be printed with the print orientation set for landscape printing.

## MICR Reader

The IBM Model 3R printer, the IBM Model 4R printer, and the IBM 4610 SureMark Point of Sale Printer models TI2 and TI4 have a MICR reader installed. When the IBM Point of Sale Subsystem printer device handler detects that the MICR reader is present, it enables application programs to read the MICR data from the printer.

Before a read MICR data escape character sequence is sent to the printer, a check should be inserted into the front of the DI station and aligned against the right wall of the DI station. If a document is not inserted in the DI station, a `POSM_PRN_PRINTER_ERROR` is sent to the application's input queue. If the check is inserted in the DI station but is not registered, it is automatically registered at the top of the document. The printer then advances the check forward until the front sensor is uncovered. The check is then fed in the opposite direction and the MICR information is read. After the MICR information is read, the check is fed to the first print position.

If the printer is unable to read any character on the first attempt, the check will be tried a second time automatically. Any character that could not be read on either attempt will be identified by a '?' (0x3F) character.

**Note:** Checks with magnetic ink printing within the first 1.2 inches of the inserted end of the check will not be read correctly.

Use the following steps to read the account information from customer checks:

1. Enable the DI station.
2. Set the **PosNprintStation** resource to `PosPRINT_STATION_DI`.
3. Insert the check into the front of the DI station and align it against the right wall of the DI station.

4. Send the register document escape character sequence with a *PosWrite()* subroutine call.
5. Send the read MICR escape character sequence with a *PosWrite()* subroutine call.
6. Wait for a POSM\_PRN\_DATA\_AVAIL event message on the point-of-sale input queue.
7. Issue a *PosRead()* subroutine call to read the MICR data.
8. Send the release document escape character sequence with a *PosWrite()* subroutine call.

**Note:** Because the check must be aligned against the right wall of the DI Station, only the last 20 characters are available for printing on the check.

---

## Fiscal Printing

The IBM Point of Sale Subsystem fiscal printer device handler allows an application to print all the fiscal commands of the supported IBM Fiscal printers. In addition, the fiscal device handler allows applications written for the IBM Model 2, IBM Model 3, IBM Model 4, and the IBM Model 4A printers to run on the IBM Model 3F Fiscal printer.

The IBM 4610 Fiscal printers are not supported in non-fiscal mode.

The default mode for the IBM Fiscal printer is PosMODE\_FISCAL with fiscal printing enabled.

---

## User-Defined Characters

The IBM 4689-001 and the IBM 4689-002 support the definition of new characters by your application program. The diagrams that follow show the layout for defining a new user-defined character (UDC).

In the diagrams, the first column indicates which print head wire will be turned on when the corresponding bit is set. Each numbered column defines the one printed dot column. Two bytes are needed to define the nine print head wires for each column.

**Note:** The first byte of the two byte column definition corresponds to the print head wire number 9 and the second byte corresponds to the upper eight print head wires.

The amount of data needed to define a character is dependent upon the number of characters per line (CPL). There are two characters-per-line lengths available on the 4689-00x printers:

- 25 CPL
- 30 CPL

### IBM 4689-00x in 25 CPL Mode

The following diagram shows the layout for defining a user-defined character on the IBM 4689-00x printers in 25 CPL mode.

The first column indicates which print head wire will be turned on when the corresponding bit is set. Each numbered column defines one printed dot column. Two bytes are needed to define the nine print head wires for each column.

**Note:** The first byte of the two byte column definition corresponds to the print head wire number 9 and the second byte defines the upper eight print head wires.

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21
Wire 1 (Bit 0)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 2 (Bit 1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 3 (Bit 2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 4 (Bit 3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 5 (Bit 4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 6 (Bit 5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 7 (Bit 6)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 8 (Bit 7)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Byte	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21
Wire 9 (Bit 0)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Byte	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41

**Note:** A period (.) in the previous table indicates that the position is not used.

### IBM 4689-00x in 30 CPL Mode

The following diagram shows the layout for defining a user-defined character on the IBM 4689-00x printers in 30 CPL mode.

The first column indicates which print head wire will be turned on when the corresponding bit is set. Each numbered column defines one printed dot column. Two bytes are needed to define the nine print head wires for each column.

**Note:** The first byte of the two byte column definition corresponds to the print head wire number 9 and the second byte defines the upper eight print head wires.

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
Wire 1 (Bit 0)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 2 (Bit 1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 3 (Bit 2)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 4 (Bit 3)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 5 (Bit 4)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 6 (Bit 5)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 7 (Bit 6)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Wire 8 (Bit 7)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Byte	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
Wire 9 (Bit 0)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Byte	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33

**Note:** A period (.) in the previous table indicates that the position is not used.

### IBM 4689-301, 3G1, 3M1, and TD5

There is a maximum of 64 characters which can be defined. The character is mapped into the Kanji Code area. However, the two-byte character code cannot duplicate an existing Kanji character code.

- For the 4689-3x1 printers, the two byte character code can be in the range of 0x8000 through 0x9FFF and 0xE000 through 0xFFFF.
- For the 4689-TD5, the range is 0x8100 through 0xFFFF.

The following diagram shows the layout for defining a user-defined character on the IBM 4689-301, 3G1, 3M1, and TD5 printers. This example shows the definition of the Kanji character "KAN". The first column indicates the byte offset within the character buffer, the second column is the bit pattern, and the third column is the dot row number.

Byte Offset	MSB		LSB		Dot Row
	76543210	76543210	76543210	76543210	
Byte 0x00, Byte 0x01, Byte 0x02	00000000	00000000	00000000	00000000	01
Byte 0x03, Byte 0x04, Byte 0x05	00000000	00111001	11000000		02
Byte 0x06, Byte 0x07, Byte 0x08	00111000	00110001	10001000		03
Byte 0x09, Byte 0x0A, Byte 0x0B	00011100	00110001	10011100		04
Byte 0x0C, Byte 0x0D, Byte 0x0E	00001101	11111111	11111100		05
Byte 0x0F, Byte 0x10, Byte 0x11	00000000	00110001	10000000		06
Byte 0x12, Byte 0x13, Byte 0x14	00000000	10000000	00110000		07
Byte 0x15, Byte 0x16, Byte 0x17	00000000	11111111	11111000		08
Byte 0x18, Byte 0x19, Byte 0x1A	01110000	11000110	00110000		09
Byte 0x1B, Byte 0x1C, Byte 0x1D	00111010	11000110	00110000		10
Byte 0x1E, Byte 0x1F, Byte 0x20	00010010	11111111	11110000		11
Byte 0x21, Byte 0x22, Byte 0x23	00000110	11000110	00000000		12
Byte 0x24, Byte 0x25, Byte 0x26	00000100	00000110	00110000		13
Byte 0x27, Byte 0x28, Byte 0x29	00001100	11111111	11111000		14
Byte 0x2A, Byte 0x2B, Byte 0x2C	00001000	00000110	00001000		15
Byte 0x2D, Byte 0x2E, Byte 0x2F	00011000	00000110	00011100		16
Byte 0x30, Byte 0x31, Byte 0x32	01111000	11111111	11111100		17
Byte 0x33, Byte 0x34, Byte 0x35	00011000	00001101	00000000		18
Byte 0x36, Byte 0x37, Byte 0x38	00011000	00011101	10000000		19
Byte 0x39, Byte 0x3A, Byte 0x3B	00011000	00011000	11000000		20
Byte 0x3C, Byte 0x3D, Byte 0x3E	00011000	00110000	01110000		21
Byte 0x3F, Byte 0x40, Byte 0x41	00011000	11100000	00111110		22
Byte 0x42, Byte 0x43, Byte 0x44	00011111	10000000	00001100		23
Byte 0x45, Byte 0x46, Byte 0x47	00000000	00000000	00000000		24

**Note:** The image pattern is the actual printed data. Character spacing and Line spacing is automatically inserted by the printer.

---

## Performance Considerations

This section applies to printers with both a CR station and a SJ station.

There is a margin to the left of the CR station and a margin to the right of the SJ station. When printing is complete at a station, the print head moves from center home to the station margin or from the margin to center home, depending on where the head was at the end of the previous print. If the print head is in the margin of one station and the next print request is for the other station, the print head must move back to the center home position before it reaches the station at which the print is requested. In this case, the print head travels across one station without printing.

Your application can print so that print head travel time is minimized. Your application has two options.

1. Set the **PosNinterleaved** resource to PosINTERLEAVED\_TRUE and print an even number of times at one station before printing at the other station. In some cases your application prints the same data at both the CR station and the SJ station. In such cases, you should alternate which station is printed at first (CR, SJ, SJ, CR, CR, SJ, and so on.). This results in an even number of prints at one station before moving to the other.
2. Set the **PosNinterleaved** resource to PosINTERLEAVED\_FALSE and allow the device handler to optimize print throughput. The data for each station is printed in the order sent by the application, but lines printed on different stations can be printed out of sequence with each other.

For logo printing, the print head must travel across the print line either one or three times. Only one pass is required if you compose the logo by using every other column of dots. To maximize performance, you should either use all odd dot columns or all even dot columns.

---

## Related Information

Additional information about printer programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOctl() Requests
- Chapter 20. Event Messages (POSM\_PRN . . .)
- Chapter 21. Resource Sets
- Appendix D. Error Codes (POSERR\_PRN. . .)

## Subroutines Used with the Printer

See Chapter 18. Application Programming Interface:

PosClose()  
PosIOctl()  
PosOpen()  
PosRead()  
PosWrite()

## Printer PosIOctl Control Requests

See Chapter 19. PosIOctl() Requests:

POS\_PRN\_DEFINE\_CHARACTERS  
POS\_PRN\_DISABLE\_DI\_PRINTING  
POS\_PRN\_DISABLE\_FISCAL\_PRINTING  
POS\_PRN\_DISCARD\_DATA  
POS\_PRN\_ENABLE\_DI\_PRINTING  
POS\_PRN\_ENABLE\_FISCAL\_PRINTING  
POS\_PRN\_RESET\_PRINTER  
POS\_PRN\_RESUME\_PRINTING  
POS\_PRN\_RETRY\_PRINTING  
POS\_PRN\_SILENCE\_TONE  
POS\_PRN\_SOUND\_TONE  
POS\_SYS\_ACQUIRE\_DEVICE  
POS\_SYS\_GET\_VALUES  
POS\_SYS\_RELEASE\_DEVICE  
POS\_SYS\_SET\_VALUES

## Printer Event Messages

See Chapter 20. Event Messages:

POSM\_PRN\_CHASE\_COMPLETE  
POSM\_PRN\_DATA\_AVAIL  
POSM\_PRN\_FISCAL\_ERROR  
POSM\_PRN\_FISCAL\_STATUS  
POSM\_PRN\_PRINTER\_ERROR  
POSM\_PRN\_STATUS\_CHANGE

## Printer Resources

See Chapter 21. Resource Sets:

PosNcodePage  
PosNCRWidth  
PosNdiOrientation  
PosNDIWidth  
PosNfeedDirection  
PosNfiscalCountry  
PosNfiscalNotify

PosNfiscalPLDStatus  
 PosNfiscalVersion  
 PosNheadParkedPosition  
 PosNinterleaved  
 PosNleftMarginCR  
 PosNlineFeedCR  
 PosNlineFeedDI  
 PosNlineFeedSJ  
 PosNprintAlignment  
 PosNprintCRCharSetx  
 PosNprintDICharSetx  
 PosNprintFeatures  
 PosNprintMode  
 PosNprintQualityMode  
 PosNprintStation  
 PosNprintStatus  
 PosNprintStatus2  
 PosNprintTabStops  
 PosNprintToneDuration  
 PosNprintToneFrequency  
 PosNprintToneNote  
 PosNprintToneOctave  
 PosNprintToneVolume  
 PosNprintUpsideDown  
 PosNrawPrintStatus  
 PosNresumeString  
 PosNretryString

## Printer Error Codes

See Appendix D. Error Codes:

4914 POSERR\_PRN\_INCORRECT\_DATA\_FORMAT  
 4906 POSERR\_PRN\_INVALID\_CR\_LF\_DISTANCE  
 4908 POSERR\_PRN\_INVALID\_DI\_LF\_DISTANCE  
 4911 POSERR\_PRN\_INVALID\_DI\_ORIENTATION  
 4901 POSERR\_PRN\_INVALID\_DI\_WIDTH  
 4909 POSERR\_PRN\_INVALID\_FEED\_DIRECTION  
 4910 POSERR\_PRN\_INVALID\_FISCAL\_NOTIFY  
 4903 POSERR\_PRN\_INVALID\_HEAD\_PARKED\_POSITION  
 4902 POSERR\_PRN\_INVALID\_INTERLEAVED\_VALUE  
 4912 POSERR\_PRN\_INVALID\_LEFT\_MARGIN\_CR  
 4905 POSERR\_PRN\_INVALID\_MODE  
 4913 POSERR\_PRN\_INVALID\_PRINT\_ALIGNMENT  
 4907 POSERR\_PRN\_INVALID\_SJ\_LF\_DISTANCE  
 4904 POSERR\_PRN\_INVALID\_STATION  
 4915 POSERR\_PRN\_LOGO\_EXISTS  
 4916 POSERR\_PRN\_UDC\_CHARACTER\_EXISTS  
 4918 POSERR\_PRN\_INVALID\_QUALITY\_MODE  
 4919 POSERR\_PRN\_INVALID\_UPSIDE\_DOWN\_MODE  
 4920 POSERR\_PRN\_INVALID\_TABSTOPS\_FORMAT  
 4921 POSERR\_PRN\_INVALID\_COLOR\_MODE  
 4922 POSERR\_PRN\_INVALID\_TONE\_VOLUME  
 4923 POSERR\_PRN\_INVALID\_TONE\_DURATION  
 4924 POSERR\_PRN\_INVALID\_TONE\_NOTE  
 4925 POSERR\_PRN\_INVALID\_TONE\_OCTAVE  
 4926 POSERR\_PRN\_INVALID\_TONE\_FREQUENCY

4927 POSERR\_PRN\_INVALID\_CODE\_PAGE

---

## Chapter 13. Programmable Power Programming

The programmable power device handler is software that manages communication between your application and the programmable power device. An application can access the device only through the device handler.

### Notes:

1. The programmable power device is not supported by the IBM Point of Sale Subsystem for Windows on the Microsoft Windows 3.1 operating system for local PosDEVICE\_POWER\_LOCAL devices.
2. The Programmable Power device is not supported on the IBM Point of Sale Subsystem for Linux.

---

### Characteristics of the Programmable Power Device

The programmable power device allows an application to control the power supply in any of the following point-of-sale terminals:

- IBM 4693-2x2
- IBM 4693-3x1 (OS/2 only)
- IBM 4693-4x1 (OS/2 only)
- IBM 4693-5x1 (OS/2 only)
- IBM 4693-7x1 (OS/2 only)

**Note:** The “x” in the above list indicates that a number or alphabetic character can be substituted.

The characteristics of the programmable power device are:

- A 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 terminal can issue programmable power commands to turn itself off.
- A 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 terminal can issue programmable power commands to control the power supply of an attached 4693-2x2.
- The programmable power device does not prevent the use of the power switch, nor does the power switch prevent the programmable power device from functioning. Only disconnecting the power cord plug can completely remove power from a terminal.
- The programmable power device in the 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 provides a way for an application to specify the day of month and time that power should be turned back on. The **PosNpowerAlarm** resource is used to set the day of month and time of day.

---

### Functions Your Application Performs

Your application can perform the following functions with the programmable power device:

- Turn on and off power to a 4693-2x2
- Turn off power to a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1
- Set the day of month and time at which power is to be restored to a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1
- Query the resource that indicates when power is to be turned back on for a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1

Before your application program can access the programmable power device, it must open the programmable power device (see *Opening Your Device*) and acquire exclusive use of it.

### Turning Power Off to a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1

An application running on a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 can use the programmable power device to turn off its own power supply.

To turn power off to a terminal and have it turned back on at a certain day of month and time of day, an application can modify the **PosNpowerAlarm** resource using the `POS_POWER_SET_ALARM PosIOctl()` request. If the value of the **PosNpowerAlarm** resource is 0 (zero) when the application issues the `POS_POWER_OFF PosIOctl()` request is issued, power is turned off and will not turn back on unless the power switch is pressed. To clear the alarm setting, an application can specify 00 (two zeros) for the day of month.

A 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 that is already turned off with a time specified for turning power back on cannot respond to any commands. If the **PosNpowerAlarm** resource is set using the `POS_POWER_SET_ALARM PosIOctl()` request, and the `POS_POWER_OFF PosIOctl()` request is issued, the alarm setting cannot be changed. When the set day of month and time arrive, power at the terminal is turned on. The maximum duration that a terminal can be set to wait before power is turned on is one month. You can, however, use the power switch to turn power on at the terminal before the specified time arrives.

On OS/2, when the `POS_POWER_OFF PosIOctl()` request is issued, the programmable power device handler calls `DosShutdown()` to prepare for turning power off to the terminal. `DosShutdown()` locks out changes to all file systems, and writes system buffers to the hard disk in preparation for turning off power. When power to the terminal is turned back on, OS/2 will not need to run `CHKDSK.EXE` to detect files that the system could not save completely, which is especially useful if you are using the High Performance File System. If other applications are running when the `POS_POWER_OFF PosIOctl()` request is issued, the `DosShutdown()` command ends them. When power to the terminal is turned back on, these applications must be restarted if you wish to use them.

If an application turns off power to a 4693-3x1, 4693-4x1, 4693-5x1, or a 4693-7x1, the attached 4693-2x2 is also turned off. Once the 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 is turned on, power to the attached 4693-2x2 will also be restored if the alternating current is present.

### Turning Power On and Off to the 4693-2x2

An application running on a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1 point-of-sale terminal can use the programmable power device to turn off the power supply to an attached 4693-2x2. If the 4693-2x2 is turned off using the programmable power device, its non-volatile random access memory (NVRAM) device can still run using +5LM (logic memory) power. NVRAM commands, including `PosWrite()` and `PosRead()`, can be processed when the main power is off but the alternating current is present.

To turn power off at the 4693-2x2, an application can issue the `POS_POWER_OFF PosIOctl()` request. To issue a `POS_POWER_OFF PosIOctl()` request, an application must have an acquired device connection to the programmable power device.

To turn power back on at the 4693-2x2, an application can issue the `POS_POWER_ON PosIOctl()` request. To issue a `POS_POWER_ON PosIOctl()` request, an application must have an acquired device connection to the programmable power device.

The following sequence of events illustrate a possible use of the programmable power device:

1. Application A opens the programmable power device
2. Application A acquires the programmable power device
3. Application B opens the programmable power device
4. Application A turns off power to the 4693-2x2
5. Application A releases the programmable power device
6. Application B acquires the programmable power device
7. Application B restores power to the 4693-2x2

Note that if application A turned off power to the 4693-2x2 before application B opened the programmable power device, application B could not have opened and acquired the programmable power device and restored power to the 4693-2x2.

Another way to turn power on is to press the power switch on the front panel of the 4693-2x2.

If an application turns off power to a 4693-3x1, 4693-4x1, 4693-5x1, or 4693-7x1, power to the attached 4693-2x2 is also turned off.

The 4693-2x2 point-of-sale terminal does not have an alarm. The **PosNpowerAlarm** resource cannot be used for the 4693-2x2.

## Querying the Time that Power Is to Be Turned On

An application can query the **PosNpowerAlarm** resource using the `POS_SYS_GET_VALUES PosIOctl()` request.

---

## Related Information

Additional information about programmable power device programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOctl() Requests
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with Programmable Power Subsystem

See Chapter 18. Application Programming Interface:

`PosClose()`  
`PosIOctl()`  
`PosOpen()`

## Programmable Power Subsystem PosIOctl() Control Requests

See Chapter 19. PosIOctl() Requests:

`POS_SYS_ACQUIRE_DEVICE`  
`POS_SYS_GET_VALUES`  
`POS_SYS_RELEASE_DEVICE`  
`POS_POWER_OFF`

## **Programmable Power Programming**

created on October 2, 2001

POS\_POWER\_ON  
POS\_POWER\_SET\_ALARM

## **Programmable Power Resources**

See Chapter 21. Resource Sets:  
PosNpowerAlarm

## **Programmable Power Device Error Codes**

See Appendix D. Error Codes:  
6401 POSERR\_POWER\_INVALID\_DAY  
6402 POSERR\_POWER\_INVALID\_HOUR  
6403 POSERR\_POWER\_INVALID\_MINUTES

---

## Chapter 14. RS-232C Programming

The RS-232C device handler is software that manages communication between your application and SIO-attached RS-232C devices. These RS-232C devices can be attached to a Feature E expansion card (in an IBM 4683-x02), attached to an IBM 4693 Point of Sale Terminal Models 2x2, or attached directly to the SIO channel on an IBM Point of Sale Terminal model. There are also some USB devices that are managed by this device handler. An application can only access such devices through this device handler.

### Notes:

1. The “x” in the paragraph above and the lists below indicates that a number or alphabetic character can be substituted.
2. The RS-232C device handler supports:
  - devices attached to the RS-232C serial ports labeled 23 and 25 on the back of an E expansion card for the IBM 4683-x02
  - devices attached to the RS-232C serial ports on the IBM 4693-2x2
  - RS-232C devices that have been modified by their manufacturers to attach directly to an SIO channel
  - USB devices that conform to the IBM USB Pseudo-RS232 Interface
3. The RS-232C device handler does not support devices attached to the RS-232C ports on any Personal Computer or on any of the following IBM Point of Sale Terminal models:
  - IBM 4683
  - IBM 4684-300
  - IBM 4693 (all models except 2x2)
  - IBM 4694 (all models)
  - IBM 4695 (all Integrated models)
4. The RS-232C device handler is not supported on the IBM Point of Sale Subsystem for Linux.

---

## Characteristics of the RS-232C Port

The following is a list of characteristics for the RS-232C port:

- Two independent full-duplex asynchronous channels
- Two independent baud rate generators
- For each channel:
  - Five through eight data bits per character
  - Break detection and generation
  - One, one and a half, or two stop bits
  - False start bit detection
  - Parity bit detection and generation: odd, even or none
  - Overrun and framing detection
  - Double buffering of data
  - Transmission rates of 110, 300, 1200, 2400, 4800, and 9600 bits per second (bps)

---

## Functions Your Application Performs

Your application can perform the following functions with the RS-232C port:

- Control the RS-232C port
- Read data received from the RS-232C port
- Write data to the RS-232C port
- Get RS-232C port status

Before your application program can access the RS-232C port, it must open the RS-232C port and acquire exclusive use of it.

**Note:** When your application closes the RS-232C port and data that has not been read is queued, the close will complete with an error code of `POSERR_SYS_DATA_DISCARDED`, and any unread data will be discarded.

## Controlling the RS-232C Port

Your application can control the RS-232C port by issuing the *PosIOctl()* subroutine. The following functions can be performed:

- Send a break
- Enable and disable the receiver
- Assert and negate DSR and DTR
- Assert and negate CTS and RTS

All 4 control lines (DSR, DTR, CTS, and RTS) can be asserted or negated as a result of the different configuration of the supported hardware. The ports on a Feature E expansion card act like data communication equipment (DCE) to any device that connects to it, while the ports on the IBM 4693 Point of Sale Terminal Models 2x2 act like data terminal equipment (DTE). For example, a serial printer attaches directly to IBM 4693 Point of Sale Terminal Models 2x2, but requires a null modem cable to attach to a Feature E card.

**Note:** When you use a null modem cable or connector to attach equipment, be sure that all serial communications lines are passed through correctly. Some null modem connectors tie the DSR and DTR lines and CTS and RTS lines together. In this situation, the equipment can go off-line without the application detecting it.

As a result of the different device configuration, the acronyms DSR, DTR, CTS, and RTS will not be used. See the **PosNlineMode** resource and the

**PosNrs232Status** resource for the names and meanings of the constants that will be used.

When the RS-232C port is disabled, it cannot receive data. By default, when the device is acquired, the port is enabled and the appropriate control lines are asserted to indicate that the port is ready to communicate. The value of the **PosNlineMode** resource is used to control the lines. Your application can change the value of the **PosNlineMode** resource on the *PosOpen()* subroutine call.

You can also obtain the values of the RS-232C port parameters by issuing the `POS_SYS_GET_VALUES PosIOctl()` request. For detailed information about these resources, see “PosRs232c Resource Set” on page 21-53.

## Reading RS-232C Data

The *PosRead()* subroutine calls are used by an application to read data that has been received from the RS-232C port. Your application receives a `POSM_RS232_DATA_AVAIL` event message on the application input queue when data is available from the RS-232C port. See “Getting Input Messages” on page 5-2 for more information about the input queue.

After your application receives a `POSM_RS232_DATA_AVAIL` event message on its input queue, your application should then call the *PosRead()* subroutine using the RS-232C device descriptor to read the data. This subroutine reads the data from the RS-232C port. The data is placed in the application buffer that was specified on the read subroutine call.

Specify the read buffer on the *PosRead()* subroutine by using the *buf* and *nbyte* parameters. The buffer length value set in *nbyte* must be big enough to hold the maximum amount of data from the RS-232C port. A value of 0 (zero) for the buffer length indicates that no data is to be read. If the value of *nbyte* of *PosRead()* specifies a value too small for the record being read, the `POSERR_SYS_BUFFER_TOO_SMALL` error is returned and data is not put into the application buffer.

*PosRead()* returns to your application immediately with either RS-232C data or an error code. If no data is available, the read completes successfully with a length of 0 (zero) returned.

When the *PosRead()* subroutine returns successfully, the first 2 bytes of the application buffer are used to indicate whether errors have occurred on receiving data. The bit assignments for the errors are defined in the file `C:\POS\INCLUDE\POS\RS232C.H`.

### Error Definitions

Error	Definition
Parity error	<code>POS_RS232_PARITY_ERROR (0x01)</code>
Overrun error	<code>POS_RS232_OVERRUN_ERROR (0x02)</code>
Framing error	<code>POS_RS232_FRAMING_ERROR (0x04)</code>

If any of these errors occur, you must decide how you want your application to regard any data in the buffer. If no error occurred on receiving data, the first 2 bytes of the application buffer are set to 0 (zero). The second 2 bytes of the application buffer are set to the number of bytes of data following this length field. The maximum length of data to be read is 251 bytes, including the first 4 bytes of overhead (4 bytes overhead plus 247 bytes maximum of user data).

See “PosRead()” on page 18-16 for the syntax of the *PosRead()* subroutine.

## Writing RS-232C Data

The *PosWrite()* subroutine call is used by an application to write data to be sent to the RS-232C port.

Specify the write buffer on the *PosWrite()* subroutine by using the *buf* and *nbyte* parameters. The buffer length value set in *nbyte* must not exceed the size of sending buffer (247 bytes). If the value of *nbyte* of *PosWrite()* specifies a value greater than 247 bytes, the POSERR\_SYS\_INVALID\_LENGTH error is returned. A value of 0 (zero) for the buffer length indicates that no data is to be sent.

The RS-232C device handler allows data for only one *PosWrite()* subroutine to be outstanding at a time. Any subsequent *PosWrite()* subroutine call will fail with error code POSERR\_RS232\_PREV\_NOT\_COMPLETE until the previous *PosWrite()* subroutine call has completed. Your application is notified that the previous *PosWrite()* subroutine call has completed by a POSM\_RS232\_XMIT\_COMPLETE event message on the application input queue. If some condition prevents or interrupts the data transmission, including closing the device, your application is notified that the previous *PosWrite()* subroutine call has failed by a POSM\_RS232\_XMIT\_ABORT event message on the point-of-sale input queue.

See “PosWrite()” on page 18-19 for the syntax of the *PosWrite()* subroutine call.

### Notes:

1. By default, the “ready to receive data” line from the RS-232C device must be asserted for the IBM Point of Sale Subsystem to transmit data. Use the **PosNlineMode** resource to change this behavior.
2. If either of the control lines sent from the attached device are dropped while the port is actively transmitting data, data might be lost or repeated.

## Getting RS-232C Port Status

The current status of the RS-232C port can be determined by issuing the POS\_SYS\_GET\_VALUES *PosIOctl()* request for the **PosNrs232Status** resource. You can get the RS-232C status regardless of whether your application has the device acquired. The following status information can be obtained:

- Status of the two control lines output to the remote device
- Status of the two control lines input from the remote device
- Transmit buffer is empty
- Receiver is enabled

---

## Related Information

Additional information about RS-232C programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOctl() Requests
- Chapter 20. Event Messages
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with RS-232C

See Chapter 18. Application Programming Interface:  
PosClose()

PosIOctl()  
PosOpen()  
PosRead()  
PosWrite()

## RS-232C PosIOctl() Control Requests

See Chapter 19. PosIOctl() Requests:

POS\_RS232\_SEND\_BREAK  
POS\_SYS\_ACQUIRE\_DEVICE  
POS\_SYS\_GET\_VALUES  
POS\_SYS\_RELEASE\_DEVICE  
POS\_SYS\_SET\_VALUES

## RS-232C Resources

See Chapter 21. Resource Sets:

PosNbaudRate  
PosNdataBits  
PosNlineMode  
PosNparity  
PosNrs232Status  
PosNstopBits  
PosNtimeoutChar

## RS-232C Event Messages

See Chapter 20. Event Messages:

POSM\_RS232\_BREAK\_DETECTED  
POSM\_RS232\_DATA\_AVAIL  
POSM\_RS232\_XMIT\_ABORT  
POSM\_RS232\_XMIT\_COMPLETE

## RS-232C Error Codes

See Appendix D. Error Codes:

5901 POSERR\_RS232\_INVALID\_BAUD\_RATE  
5904 POSERR\_RS232\_INVALID\_DATA\_BITS  
5903 POSERR\_RS232\_INVALID\_PARITY  
5902 POSERR\_RS232\_INVALID\_STOP\_BITS  
5905 POSERR\_RS232\_INVALID\_TIMEOUT\_CHAR  
5907 POSERR\_RS232\_PREV\_NOT\_COMPLETE



---

## Chapter 15. Scale Programming

The scale device handler manages communication between your application and the scale. An application can only access a scale device through the scale device handler.

**Note:** The scale devices are not supported by the IBM Point of Sale Subsystem for Windows on the Microsoft Windows 3.1 operating system.

---

### Characteristics of the Scale Devices

The scale device handler supports the following devices:

- IBM 4687 Point of Sale Scanner Model 2
- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4698 Point of Sale Scanner Model 2
- IBM USB Scale Interface

The supported scales share some common characteristics:

- Each supported scale allows the application to request that the weight be returned in either English (Avoirdupois) or Metric units. (A hardware adjustment to the IBM 4687 Model 2 is necessary to switch from one unit of weight to the other.)
- The supported scales have similar weighing specifications. For example, each scale has a weight capacity of 30 pounds (15 kilograms).

### IBM 4687 Point of Sale Sale Scanner Model 2

The IBM 4687 Point of Sale Scanner Scale Model 2 is a flat-top scanner combined with a rapid settling scale that attaches to IBM point-of-sale terminals. The IBM 4687 Model 2 scale allows the user to weigh items on a flat surface without lifting the items from the counter top.

**Note:** The IBM 4687 Point of Sale Scanner Model 2 is not supported on the IBM Point of Sale Subsystem for Linux

Operator controls include a zero light button which illuminates when the scale has no items on its top surface and the IBM 4687 is stable. This button can also be used to zero the scale. A remote display unit is included with the IBM 4687 Point of Sale Scanner Model 2.

See Chapter 16. Scanner Programming for information about communications between an application and the IBM 4687 scanner.

### IBM 4696 Point of Sale Scanner Scale Model 1

The IBM 4696 Scanner Scale Model 1 is a high-performance horizontal scanner with an integrated retail scale. Operator scale controls are top-mounted and include a scale zero switch and a scale zero indicator. Many of the scale features can be configured by the user. The IBM 4696 also supports an optional remote display unit for the scale.

**Note:** The IBM 4696 Point of Sale Scanner Model 1 is not supported on the IBM Point of Sale Subsystem for Linux

See Chapter 16. Scanner Programming for information about communications between an application and the IBM 4696 scanner.

## IBM 4698 Point of Sale Scanner Model 2

The IBM 4698 Scanner Model 2 combines the IBM 4698 Scanner Model 1 with a fast-settling, high-reliability scale. Operator scale controls include a scale zero switch and a scale zero indicator. Many of the scale features can be configured by the user. The IBM 4698 also supports an optional remote display unit for the scale.

See Chapter 16. Scanner Programming for information about communications between an application and the IBM 4698 scanner.

## IBM USB Scale Interface

Scale devices that conform to the IBM USB Scale Interface specification are similar in feature to the scale in the IBM 4698 Scanner Model 2. Properly written IBM Point of Sale Subsystem applications which were developed for any of the SIO-attached scales supported by this device handler, should run correctly (unmodified) with USB scales that conform to the IBM USB Scale Interface specification.

**Note:** The IBM USB Scale Interface is not supported on the IBM Point of Sale Subsystem for Linux

---

## Functions Your Application Performs

Your application can perform the following functions with a scale:

- Read scale data
- Configure the scale
- Zero the scale
- Clear the scale display

Before your application program can access the scale, it must open the scale and acquire exclusive use of the scale.

## Reading Scale Data

The *PosRead()* subroutine call is used by an application to request data from the scale. The *PosRead()* request returns to your application immediately with scale data or with an error code. See “PosRead()” on page 18-16 for the syntax of the read subroutine call.

Scale data is placed in the application’s buffer that is specified on the *PosRead()* subroutine call. The format of the buffer returned by the scale device handler is specified in the structure *PosScaleData* (see the header file `C:\POS\INCLUDE\POS\SCALE.H`), and is shown in Figure 15-1.

WtType	Flags	WtVal
--------	-------	-------

Figure 15-1. Scale Data Buffer Format

The following sections contain descriptions of these fields:

**WtType** (unsigned short)

A value indicating which unit of weight the data represents, English or metric. Possible values are PosENGLISH (0x00) or PosMETRIC (0x01).

**Flags** (unsigned short)

Flags are used to indicate the status of the scale data. The flag values below are bitwise ORed together to form the *Flags* value. A weight value (*WtVal*) is not returned if any of these flags are set. The following flag values can be returned by all supported scales:

**POS\_SCALE\_IN\_MOTION (0x0001)**

This flag is set when there is no weight data being returned to the scale data buffer. It will be set whenever any of the other flags are set. When this flag is the only flag set, it can mean that the scale has detected scale motion. Scale motion is a normal occurrence in scale operation. However, if scale motion is continually detected, it could mean that the scale requires calibration.

**POS\_SCALE\_HW\_ERROR (0x0040)**

This flag is set when the scale detects a hardware error.

The following flag values can be returned by the IBM 4696 Point of Sale Scanner Scale Model 1, the IBM 4698 Point of Sale Scanner Model 2, and the IBM USB Scale Interface.

**POS\_SCALE\_UNDER\_ZERO (0x0002)**

This flag is set when the scale registers a weight less than zero.

**POS\_SCALE\_OVER\_CAPACITY (0x0004)**

This flag is set when an item weighing more than 30 pounds (15 kilograms) is placed on the scale.

**POS\_SCALE\_REQUIRES\_ZEROING (0x0008)**

This flag is set when the **PosNzeroRetState** is set to PosENABLE and one of the following occurs:

- An item has been left on the scale for more than four minutes. The item must be removed to allow the scale to return to zero, before weight requests will be answered.
- A negative weight value is registered by the scale prior to the item being placed on the scale. The scale must be zero-adjusted (reset to zero) before weight requests will be answered.

In either case, the scale display shows -0- after *PosRead()* is attempted.

**POS\_SCALE\_NOT\_READY (0x0010)**

This flag is set when the scale device is not yet ready to receive weight requests. An example of when this flag might be set is when a *PosRead()* request is attempted while the scale display shows a moving dash (–) character. When your scale display shows the moving dash character, the scale has attempted to reset while an item was on the scale. The scale will not return a weight value while the moving dash character is displayed. Removing the item from the scale allows the scale to complete its reset.

**POS\_SCALE\_DUPLICATE\_WEIGHT (0x0020)**

This flag is set when the scale is operating in PosUK mode and has received a weight request when the non-zero weight on the scale

has not changed more than 40 scale increments or the scale has not returned to a weight of zero prior to the weight request.

The following flag value can only be returned by the IBM USB Scale Interface:

**POS\_SCALE\_FIVE\_DIGIT\_WEIGHT (0x0080)**

This flag is set when the returned weight contains five digits. Four weight digits are returned when this flag is not set.

**WtVal** (long)

The weight read by the scale. Weights can be given in hundredths of a pound, thousandths of a pound, or in thousandths of a kilogram (grams). For example, a weight of 0.5 pounds can be returned as 50 hundredths of a pound or 500 thousandths of a pound. A weight of 0.5 kilograms is returned as 500 thousandths of a kilogram.

## Configuring the Scale

A scale can be configured using two methods:

1. Change the resource file entries for the scale. See “The Resource File” on page 3-1 for information about changing the resource file.
2. Pass overriding arguments on the *PosOpen()* function call. See “PosOpen()” on page 18-10 for the syntax of this subroutine call.

“PosDevice Resource Set” on page 21-7 and “PosScale Resource Set” on page 21-56 list the resources that are configurable for scale devices. “Scale Default Values” on page 15-4 lists the default configuration for each supported scale.

## Zeroing the Scale

An application can zero the scale by issuing the `POS_SCALE_ZERO_SCALE PosIOctl()` request. This request is not valid for the IBM 4687 Point of Sale Scanner Model 2.

See “POS\_SCALE\_ZERO\_SCALE” on page 19-39 for more information about zeroing the scale.

## Clearing the Scale Display

An application can clear the remote scale display by issuing the `POS_SCALE_CLEAR_SCREEN PosIOctl()` request. This request is not valid for the IBM 4687 Point of Sale Scanner Model 2.

See “POS\_SCALE\_CLEAR\_SCREEN” on page 19-38 for more information about clearing the remote scale display.

---

## Scale Default Values

This section lists the default values for all resources used by each of the supported scales. The default value for a particular resource is used when a value for that resource is not specified in one of the following ways:

- By the application in the argument list on the scale *PosOpen()* call
- In the resource file for the application

If a value for the resource is specified in the argument list and in the resource file, the value in the argument list is used.

This section can be used as a cross-reference to determine which of the resources are configurable for each scale. Any resources not listed for a particular scale are not used by that scale.

See “PosScale Resource Set” on page 21-56 for information about the resources supported by the scales and the valid values for each resource.

## IBM 4687 Point of Sale Scanner Model 2

The default scale resource values for the IBM 4687 Point of Sale Scanner Model 2 are:

Resource	Default Value
PosNweightMode	PosENGLISH

## IBM 4696 Point of Sale Scanner Scale Model 1 and IBM 4698 Point of Sale Scanner Model 2

The default scale resource values for the IBM 4696 Point of Sale Scanner Scale Model 1 and the IBM 4698 Point of Sale Scanner Model 2 are:

Resource	Default Value
PosNoperMode	PosUSCANADA
PosNdisplayRequired	PosENABLE
PosNvibrationFilter	PosLOW
PosNweightMode	PosENGLISH
PosNzeroIndState	PosENABLE
PosNzeroRetState	PosDISABLE

## IBM USB Scale Interface

The default scale resource values for the IBM USB Scale Interface are:

Resource	Default Value
PosNoperMode	PosUSCANADA
PosNdisplayRequired	PosENABLE
PosNnumWeightDigits	PosFOUR_WEIGHT_DIGITS
PosNvibrationFilter	PosLOW
PosNweightMode	PosENGLISH
PosNzeroIndState	PosENABLE
PosNzeroRetState	PosDISABLE

---

## Related Information

Additional information about scale programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOctl() Requests
- Chapter 20. Event Messages
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with Scale

See Chapter 18. Application Programming Interface:

```
PosClose()
PosIOctl()
PosOpen()
PosRead()
```

## Scale PosIOctl() Control Requests

See Chapter 19. PosIOctl() Requests:

- POS\_SCALE\_CLEAR\_SCREEN
- POS\_SCALE\_ZERO\_SCALE
- POS\_SYS\_ACQUIRE\_DEVICE
- POS\_SYS\_GET\_VALUES
- POS\_SYS\_RELEASE\_DEVICE

## Scale Resources

See Chapter 21. Resource Sets:

- PosNoperMode
- PosNdisplayRequired
- PosNnumWeightDigits
- PosNvibrationFilter
- PosNweightMode
- PosNzeroIndState
- PosNzeroRetState

## Scale Error Codes

See Appendix D. Error Codes:

- 6209 POSERR\_SCALE\_CONFIGURATION\_ERROR
- 6208 POSERR\_SCALE\_INVALID\_CLEAR\_SCREEN\_REQUEST
- 6211 POSERR\_SCALE\_INVALID\_NUM\_WEIGHT\_DIGITS
- 6201 POSERR\_SCALE\_INVALID\_OPERATIONS\_MODE
- 6202 POSERR\_SCALE\_INVALID\_REMOTE\_DISPLAY\_STATE
- 6203 POSERR\_SCALE\_INVALID\_VIBRATION\_FILTER
- 6204 POSERR\_SCALE\_INVALID\_WEIGHT\_MODE
- 6205 POSERR\_SCALE\_INVALID\_ZERO\_INDICATOR\_STATE
- 6206 POSERR\_SCALE\_INVALID\_ZERO\_RETURN\_STATE
- 6207 POSERR\_SCALE\_ZERO\_SCALE\_FAILED

---

## Chapter 16. Scanner Programming

The scanner device handler manages communication between your application and a scanner. An application can only access a scanner device through the scanner device handler.

---

### Characteristics of the Scanners

The scanner device handler supports the following devices:

- IBM Hand-Held Bar Code Reader Model 1 (HHBCR-1)
- IBM Hand-Held Bar Code Reader Model 2 (HHBCR-2)
- IBM 1520 Hand-Held Scanner Model A02
- IBM 4685 Hand-Held Bar Code Reader Model 001 (4685-1)
- IBM 4685 Hand-Held Bar Code Reader Model K01 (4685-K01)
- IBM 4686 Retail Point of Sale Scanner Models 1 and 2 (4686-1 and 4686-2)
- IBM 4686 Retail Point of Sale Scanner Models 3 and 4 (4686-3 and 4686-4)
- IBM 4687 Point of Sale Scanner Model 1 (4687-1)
- IBM 4687 Point of Sale Scanner Model 2 (4687-2)
- IBM 4696 Point of Sale Scanner Scale Model 1 (4696-1)
- IBM 4697 Point of Sale Scanner Model 1 (4697-1)
- IBM 4698 Point of Sale Scanner Model 1 (4698-1)
- IBM 4698 Point of Sale Scanner Model 2 (4698-2)
- IBM USB Scanner Interface

The supported scanners share some common characteristics:

- The state of all scanners when power is turned on is locked (disabled).
- Any devices attached to the scanner have unique device names and are controlled by a device driver specific to that device. Requests sent to those devices' drivers do not affect the scanner. For example, the IBM 4687 Point of Sale Scanner Model 2 and the IBM 4696 Point of Sale Scanner Scale Model 1 both include a scale. A scale device driver is provided to handle communications between the scale and an application.
- The only data validation (such as CRC checking) that is done is performed by the scanner hardware. Bar codes such as UPC and EAN have check characters built in. Scanners read and check this code before returning the data.
- All of the supported scanners allow the application to unlock and lock (enable and disable) the device and to enable and disable the beeper.

### Hand-Held Bar Code Readers

The Feature Code 4500 Hand-Held Bar Code Reader (HHBCR-1) reads bar codes and can be attached to IBM point-of-sale terminals. Feature Code 4501 is an improved version of the HHBCR-1, and is referred to as the Hand-Held Bar Code Reader Model 2 (HHBCR-2).

**Note:** The Hand-Held Bar Code Reader Model 1 is not supported on the IBM Point of Sale Subsystem for Linux.

The IBM 4685 Hand-Held Bar Code Reader Model 001 and Model K01 are similar to the HHBCR-2. The HHBCR-2 is backward compatible with the HHBCR-1. The major difference between the two models is that the HHBCR-2 reads a greater variety of bar code types. The IBM 4685 Hand-Held Bar Code Readers are supported as a HHBCR-2.

See “PosNbarCodes1” on page 21-62 for a list of the bar code types read by HHBCR-1 and HHBCR-2.

## IBM 1520 Hand-Held Scanner Model A02

The IBM 1520 Hand-Held Scanner Model A02 is a hand-held laser scanner. The IBM 1520-A02 Scanner can be configured to read any combination of the four bar code types it supports. The bar codes to be recognized by the IBM 1520-A02 are specified by the scanner resources **PosNbarCodes1**, **PosNbarCodes2**, **PosNbarCodes3**, and **PosNbarCodes4**.

See “PosNbarCodes1” on page 21-62 for a list of the bar code types that can be read by the IBM 1520-A02 Scanner.

## IBM 4686 Retail Point of Sale Scanner

The IBM 4686 Retail Point of Sale Scanner is a compact bar code scanner that attaches to most IBM point-of-sale terminals. The scanner provides fast, accurate reading performance for retail applications. Many aspects of this scanner can be configured. There are four different models of the IBM 4686 Retail Point of Sale Scanner.

**Note:** The IBM 4686 Retail Point of Sale Scanner is not supported on the IBM Point of Sale Subsystem for Linux.

See “PosNbarCodes1” on page 21-62 and “PosNbarCodes2” on page 21-64 for a list of the bar code types read by the IBM 4686 Retail Point of Sale Scanners.

## IBM 4687 Point of Sale Scanner

The IBM 4687 Point of Sale Scanner Model 1 is a flat-top scanner that attaches to IBM point-of-sale terminals. This scanner can be installed horizontally or vertically with equal performance.

The IBM 4687 Point of Sale Scanner Model 2 is the IBM 4687 Point of Sale Scanner Model 1 combined with a rapid settling scale. For information about communications between an application and the IBM 4687 scale, see “Chapter 15. Scale Programming” on page 15-1. This scanner must be installed in a horizontal position in a counter top.

**Note:** The IBM 4687 Point of Sale Scanner is not supported on the IBM Point of Sale Subsystem for Linux.

The IBM 4687 Point of Sale Scanner Models 1 and 2 are capable of reading the following bar code types:

- UPC-A
- UPC-E
- UPC-D
- EAN-8
- EAN-13
- JAN-8
- JAN-13

## IBM 4696 Point of Sale Scanner Scale

The IBM 4696 Point of Sale Scanner Scale Model 1 is a high-performance horizontal scanner with an integrated retail scale. It is a compact device with a centered window that allows installation options such as seated scanning or

installation of a cash drawer under the scanner. Operator scanner controls include a speaker volume switch and a scan/mode indicator. See “PosNbarCodes1” on page 21-62 for a list of the bar code types read by the IBM 4696 Point of Sale Scanner Scale Model 1.

**Note:** The IBM 4696 Point of Sale Scanner Scale is not supported on the IBM Point of Sale Subsystem for Linux.

See Chapter 15. Scale Programming for information about communications between an application and the IBM 4696 scale.

## IBM 4697 Point of Sale Scanner

The IBM 4697 Point of Sale Scanner Model 1 is a high-performance bar code scanner. It is a compact device with a centered window that allows installation options such as seated scanning or installation of a cash drawer under the scanner. The IBM 4697 Point of Sale Scanner is installed in a horizontal position. Operator controls include a speaker volume switch and a scan/mode indicator.

See “PosNbarCodes1” on page 21-62 and “PosNbarCodes2” on page 21-64 for a list of the bar code types read by the IBM 4697 Point of Sale Scanner Model 1.

**Note:** The IBM 4697 Point of Sale Scanner is not supported on the IBM Point of Sale Subsystem for Linux.

## IBM 4698 Point of Sale Scanner

Both models of the IBM 4698 Point of Sale Scanner include a 360-degree, high-performance scanner that is designed as an ergonomic solution for scanning. This 360-degree scanner can read the bottom and all four sides of a product simultaneously so minimal product orientation by the operator is required.

See “PosNbarCodes1” on page 21-62, “PosNbarCodes2” on page 21-64, “PosNbarCodes3” on page 21-65, and “PosNbarCodes4” on page 21-66 for a list of the bar code types read by the IBM 4698 Point of Sale Scanner models.

The IBM 4698 Point of Sale Scanner Model 1 is a scanner only. The IBM 4698 Point of Sale Scanner Model 2 is the Model 1 scanner combined with a scale. For information about communications between an application and the IBM 4698 scale, see “Chapter 15. Scale Programming” on page 15-1.

## IBM USB Scanner Interface

Scanner devices that conform to the IBM USB Scanner Interface specification are most similar in feature to the IBM 4698 Scanner models although the interface specification actually combines features from all of the scanners described here and adds support for some of the newer bar code types. Properly written IBM Point of Sale Subsystem applications that were developed for any of the SIO-attached scanners supported by this device handler, should run correctly (unmodified) with USB scanners that conform to the IBM USB Scanner Interface specification.

**Note:** The IBM USB Scanner Interface is not supported on the IBM Point of Sale Subsystem for Linux.

See “PosNbarCodes1” on page 21-62, “PosNbarCodes2” on page 21-64, “PosNbarCodes3” on page 21-65, and “PosNbarCodes4” on page 21-66 for a list of the bar code types supported in the IBM USB Scanner Interface.

---

## Functions Your Application Performs

Your application can perform the following functions with a scanner:

- Unlock and lock the scanner
- Read scanner data
- Discard scanner data
- Configure the scanner
- Write data to the scanner (USB Scanners only)

Before your application program can access the scanner, it must open the scanner and acquire exclusive use of the scanner.

## Unlocking and Locking the Scanner

The scanner can be in either the locked state or the unlocked state. In the locked state, bar codes should not be returned by the scanner. When an application acquires a scanner, it is in the locked state.

The scanner changes from the unlocked state to the locked state when one of the following occurs:

- The application issues the `POS_SYS_LOCK_DEVICE PosIOctl()` request
- The application issues the `POS_SYS_RELEASE_DEVICE PosIOctl()` request
- The application calls the `PosClose()` subroutine while it has the scanner acquired

See “`PosIOctl()`” on page 18-8 and “`PosClose()`” on page 18-3 for the syntax of these subroutine calls.

An application must issue the `POS_SYS_UNLOCK_DEVICE Chapter 19. PosIOctl()` *Requests* request to allow the scanner to recognize and return bar code data. The scanner should be left in the locked state any time that scanner data is not expected to prevent scanner input from getting too far ahead of the application.

## Reading Scanner Data

Data is available after it is read by the scanner, but it is not passed to an application until the `PosRead()` subroutine is called. The application receives a `POSM_SCAN_DATA_AVAIL` event message on the application’s input queue when data is available from the scanner. See “Getting Input Messages” on page 5-2 for more information about the input queue.

The `PosRead()` subroutine call is used by an application to get data that has been read by the scanner. The `PosRead()` subroutine returns to your application immediately with either scanner data or an error code. If no data is available and no error occurs, the read completes successfully with a length of zero returned. See “`PosRead()`” on page 18-16 for the syntax of the read subroutine call.

Data is read on a first-in-first-out (FIFO) basis. The first data to be read by a scanner is the first data returned to an application. It is necessary for your application to acquire the scanner device before calling the `PosRead()` subroutine. See “Acquiring and Releasing Your Devices” on page 5-6 for more information about acquiring the scanner device.

Scanner data is placed in the application’s buffer as specified on the `PosRead()` subroutine call. The format of the buffer returned to the scanner device handler is shown in Figure 16-1 on page 16-5. The header portion of the returned data is defined in the structure `PosScannerDataHdr` (see the header file `C:\POS\INCLUDE\POS\SCANNER.H`).

The POSERR\_SYS\_LOCKED\_NO\_DATA\_READ error code is returned when the scanner is in the locked state and no data is available. Any data available prior to the scanner being locked is returned by a read request before this error code is returned.

The scanner data buffer consists of the following four fields, which are repeated for

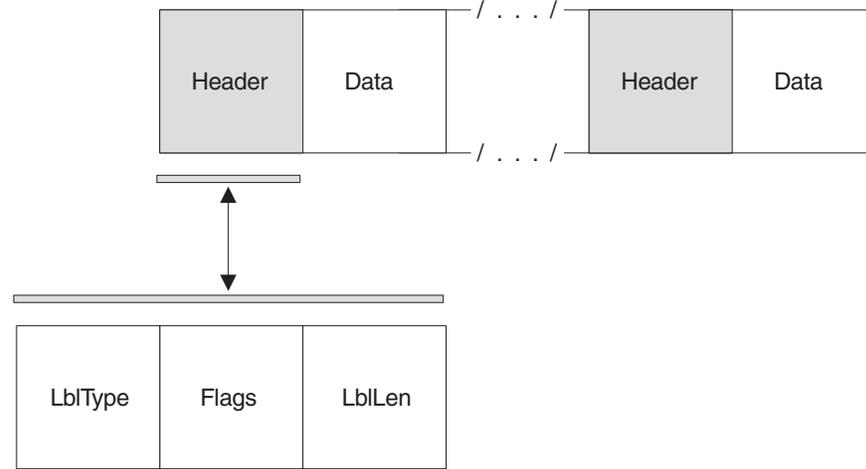


Figure 16-1. Scanner Data Buffer Format

each bar code in the buffer:

- LblType (unsigned long)
- Flags (unsigned short)
- LblLen (unsigned short)
- Data (unsigned char[ ])

The following sections contain descriptions of these fields.

**LblType** (unsigned long)

Indicates which type of bar code the following data represents. See the following table for possible values of this field.

Table 16-1. Label Types Returned by the Scanner Device Handler

Constant Value	Bar Code Type
PosLABEL_CODABAR (0x00080000)	Codabar
PosLABEL_CODE_39 (0x00040000)	Code 39
PosLABEL_CODE_93 (0x00020000)	Code 93
PosLABEL_CODE_128 (0x00010000)	Code 128
PosLABEL_EAN_8 (0x00002000)	EAN-8
PosLABEL_EAN_8_PLUS_2 (0x00002002)	EAN-8 plus 2-digit supplemental
PosLABEL_EAN_8_PLUS_5 (0x00002001)	EAN-8 plus 5-digit supplemental
PosLABEL_EAN_8_PLUS_CODE128 (0x00002010)	EAN-8 plus Code 128 supplemental
PosLABEL_EAN_13 (0x00001000)	EAN-13
PosLABEL_EAN_13_PLUS_2 (0x00001002)	EAN-13 plus 2-digit supplemental
PosLABEL_EAN_13_PLUS_5 (0x00001001)	EAN-13 plus 5-digit supplemental
PosLABEL_EAN_13_PLUS_CODE128 (0x00001010)	EAN-13 plus Code 128 supplemental
PosLABEL_INT_2_OF_5 (0x00000008)	Interleaved 2 of 5

Table 16-1. Label Types Returned by the Scanner Device Handler (continued)

Constant Value	Bar Code Type
PosLABEL_STD_2_OF_5 (0x00000004)	Standard 2 of 5
PosLABEL_UNKNOWN (0x00000000)	Unknown. All data received from the scanner is returned to the application.
PosLABEL_UCC_EAN_128 (0x00100000)	UCC/EAN-128
PosLABEL_UPC_A (0x00008000)	UPC Version A
PosLABEL_UPC_A_PLUS_2 (0x00008002)	UPC Version A plus 2-digit supplemental
PosLABEL_UPC_A_PLUS_5 (0x00008001)	UPC Version A plus 5-digit supplemental
PosLABEL_UPC_A_PLUS_CODE128 (0x00008010)	UPC Version A plus Code 128 supplemental
PosLABEL_UPC_D1 (0x00000800)	UPC Version D-1
PosLABEL_UPC_D2 (0x00000400)	UPC Version D-2
PosLABEL_UPC_D3 (0x00000200)	UPC Version D-3
PosLABEL_UPC_D4 (0x00000100)	UPC Version D-4
PosLABEL_UPC_D5 (0x00000080)	UPC Version D-5
PosLABEL_UPC_E (0x00004000)	UPC Version E
PosLABEL_UPC_E_PLUS_2 (0x00004002)	UPC Version E plus 2-digit supplemental
PosLABEL_UPC_E_PLUS_5 (0x00004001)	UPC Version E plus 5-digit supplemental
PosLABEL_UPC_E_PLUS_CODE128 (0x00004010)	UPC Version E plus Code 128 supplemental

**Note:** The JAN-8 and JAN-13 bar codes are subsets of the EAN-8 and EAN-13 bar codes.

**Flags** (unsigned short)

Indicates the status of the data. The flag values below are bitwise ORed together to form the *Flags* value.

**POS\_SCAN\_LAST\_BLOCK (0x0001)**

This flag is set when this is the last block of data for this bar code. It is always set when multiple block read mode is disabled.

**POS\_SCAN\_UNEXPECTED\_DATA (0x0002)**

This flag is set when this data was received from the scanner while the scanner was locked or was not acquired by this application. The data could be out of synchronization with what the application is expecting. This flag is also set for any data buffered for this application before the current POS\_SYS\_ACQUIRE\_DEVICE request was processed. For this flag to be clear (0), the scanner must be both acquired and unlocked by the application when the data is received.

**POS\_SCAN\_PARTIAL\_LABEL (0x0004)**

This flag is set when a hardware or other error with the scanner caused the scanner to be reset before the entire label was received by the IBM Point of Sale Subsystem. Whatever label data was received from the scanner is returned to the application. This flag is also set when there is

insufficient memory available to store the entire label. In this case, as much of the label that was stored is returned. This flag is set in each block of a multi-block label.

#### **POS\_SCAN\_LABEL\_READ\_ERROR (0x0008)**

This flag is set when the HHBCR-1 or HHBCR-2 encounters a problem with the label it is attempting to read. The label type might be one that the scanner is not currently configured to recognize or, if modulo checking is enabled, the label might contain an incorrect check digit or no check digit. No label data is returned when this flag is set.

#### **POS\_SCAN\_DIRECT\_IO\_RESPONSE (0x0010)**

This flag is set when the scanner data is a response to a previous *PosWrite()* command. When this flag is set, the **LblType** will be zero and the **LblLen** field will indicate whether there is additional data.

#### **POS\_SCAN\_DIRECT\_IO\_ERROR (0x0020)**

This flag is set when the scanner data is an error response to a previous *PosWrite()* command. When this flag is set, the **LblType** will be zero and the **LblLen** field will indicate whether there is additional data.

#### **POS\_SCAN\_UNEXPECTED\_LENGTH (0x0040)**

This flag is set when the length of the received scanner data is longer or shorter than the expected length for the label type. This flag will be set for UPC-A and UPC-E labels that do not contain check digits. Label data will be returned to the application when this flag is set.

**LblLen** (unsigned short)

Indicates the length of the **Data** field in bytes.

**Data** (unsigned char[ ])

Contains data returned by the scanner. This data can be barcode data or the scanner's response to a previous *PosWrite()* command.

When barcode data is received from the scanner, it has header and trailer information that indicates the type of barcode that was read. If the scanner device handler recognizes the barcode type, it removes the type information before sending the barcode data to the application; it also converts any binary digits to ASCII before sending the data to the application - - for example, binary one (0x01) is converted to ASCII one (0x31). If the barcode type is unknown, all data received from the scanner is returned to the application unchanged.

When the data received from the scanner is in response to a *PosWrite()* command, the data is sent to the application unchanged. The format of the returned data is dependent on the scanner command that was sent. The scanner device manufacturer is responsible for documenting these commands.

If multiple labels have been buffered for this application, only the first label is returned by *PosRead()*. If multiple block read mode is enabled, there is a header/data pair for each bar code in the label.

The size of the buffer passed on the *PosRead()* call must be large enough to hold the maximum amount of data for the bar code plus the 8-byte header. If multiple block read mode is enabled, the buffer must be big enough to hold the maximum amount of data for each bar code in the label plus the 8-byte header for each bar code. If the buffer is too small for the data being returned, the **POSERR\_SYS\_BUFFER\_TOO\_SMALL** error code is returned and data is not put into the application's buffer.

An application should be able to process bar code types that it is not expecting. This situation can occur if the application requires the scanner to read a specific set of bar code types, but the scanner can only be configured such that more than just those specific bar code types are recognized.

Label data received from the scanner device is associated with a particular scanner device connection. Label data is stored until the application reads it or closes the scanner device connection through which the label data was received. An application only has access to the label data received from the scanner through its scanner device connections and only while it has the scanner device acquired. An application does not have access to the label data received by any other application.

An application should process all of the label data that has been queued for a scanner device connection before closing that device connection. If *PosClose()* is issued for the scanner device connection while labels are still queued, the device connection is successfully closed, the labels are discarded, and the error code `POSERR_SYS_DATA_DISCARDED` is returned to the application. The application can issue the `POS_SYS_GET_VALUES PosIOctl()` request for the **PosNLabelsQueued** resource to determine how many buffered labels have not yet been processed for the scanner device connection.

### Discarding Scanner Data

An application can discard any buffered labels that it has not read by issuing the `POS_SCAN_DISCARD_DATA PosIOctl()` request. See “`POS_SCAN_DISCARD_DATA`” on page 19-40 for more information about discarding scanner data.

### Configuring the Scanner

A scanner can be configured using three methods:

1. Change the resource file entries for the scanner. See “The Resource File” on page 3-1 for information about changing the resource file.
2. Pass overriding arguments on the *PosOpen()* function call. See “*PosOpen()*” on page 18-10 for the syntax of this subroutine call.
3. Issue the `POS_SYS_SET_VALUES PosIOctl()` request. See “*PosIOctl()*” on page 18-8 for the syntax of this subroutine call.

“*PosDevice Resource Set*” on page 21-7 and “*PosScanner Resource Set*” on page 21-60 list the resources that are configurable for scanner devices. “*Scanner Default Values*” on page 16-9 lists the default configuration for each supported scanner.

### Writing Data to the Scanner

The *PosWrite()* function is only supported for USB-attached scanners. The use of *PosWrite()* is intended to allow applications to configure new scanners for features not considered in the IBM USB Scanner Interface specification. When new scanning features or bar code types are introduced, they can be integrated into existing applications without a change in this device handler. No checking or interpretation of the data is done by the device handler. It is the responsibility of the scanner device manufacturer to document the data that can be sent using the *PosWrite()* function.

---

## Processing Unexpected Scanner Data

When the scanner device has been locked by an application, it should not be able to read labels. However, some scanners will read labels even when they have been locked. For example, the IBM 4686 Retail Point of Sale Scanners (all models) have a button which controls the scanner laser and motor. When this button is pushed, the motor and laser will be reactivated and the scanner will immediately be able to read labels regardless of the previous state (locked or unlocked) of the scanner device. An application should be prepared to process this unexpected label data received from the scanner.

You can choose to have the application ignore any label data received while the scanner is acquired and locked by setting the **PosNqueueAllLabels** resource to PosDISABLE. When the resource is set to PosDISABLE, the scanner device driver discards any label data received while the scanner device connection is locked. You can choose to have the application store the label data received while the scanner device connection is acquired and locked by setting the **PosNqueueAllLabels** resource to PosENABLE. When the resource is set to PosENABLE, the scanner device driver stores all label data received from the scanner, setting the POS\_SCAN\_UNEXPECTED\_DATA flag for each label that is received while the scanner device is acquired and locked.

See “PosNqueueAllLabels” on page 21-80 for more information about configuring the device handler to ignore label data while the scanner device connection is acquired and locked. When the scanner device is locked but is not currently acquired by any application, the **PosNqueueAllLabels** resource has no effect. In this case, all label data is stored with the POS\_SCAN\_UNEXPECTED\_DATA flag set.

If the **PosNlabelsQueued** resource has a value greater than 0 (zero) after your application has acquired the scanner device but before it has unlocked the device, all of the queued labels have the POS\_SCAN\_UNEXPECTED\_DATA flag set. By checking the **PosNlabelsQueued** resource just after acquiring the scanner device but before unlocking it, your application can determine whether it has received unexpected label data, and how many unexpected labels are in the queue. You can use the POS\_SYS\_GET\_VALUES *PosIOctl()* request to get the value of **PosNlabelsQueued**.

Unexpected label data might require processing by the application. Discarding these labels might mean that items are being given away. For example, suppose the scanner device is acquired but locked, the **PosNqueueAllLabels** resource is set to PosDISABLE, and this particular scanner device is capable of reading label data when it has been locked. The operator scans an item, the locked scanner emits a beep indicating that a good read has occurred, and the operator, believing that the label has been processed, scans the next item. Because the scanner is locked and labels are not being stored, the label data never reaches the application.

---

## Scanner Default Values

This section lists the default values for all resources used by each of the supported scanners. The default value for a particular resource is used when a value for that resource is not specified in one of the following ways:

- By the application in the argument list on the scale *PosOpen()* call
- In the resource file for the application

If a value for the resource is specified in the argument list and in the resource file, the value in the argument list is used. This section can be used as a cross-reference to determine which of the resources are configurable for each scanner. Any resources not listed for a particular scanner are not used by that scanner.

See “PosScanner Resource Set” on page 21-60 for discussion of the resources supported by scanners and the valid values for each resource.

## Hand-Held Bar Code Reader (All Models)

The default resource values for the HHBCR models 1 and 2 and the 4685 Hand Held Bar Code Readers are:

<b>Resource</b>	<b>Default Value</b>
PosNbarCodes1	PosLGROUP_UPC_EAN_D3
PosNbeepState	PosENABLE
PosNblockReadMode	1
PosNblock1Type	PosLABEL_NO_CHECK
PosNblock2Type	PosLABEL_NO_CHECK
PosNblock3Type	PosLABEL_NO_CHECK
PosNcheckModulo	PosDISABLE
PosNdTouchMode	PosDISABLE
PosNqueueAllLabels	PosENABLE

## IBM 1520 Hand-Held Scanner Model A02

The default resource values for the IBM 1520-A02 Scanner are:

<b>Resource</b>	<b>Default Value</b>
PosNbarCodes1	PosLGROUP_UPC_EAN
PosNbarCodes2	PosLGROUP_UPC_D1_TO_D5
PosNbarCodes3	PosLGROUP_NONE
PosNbarCodes4	PosLGROUP_NONE
PosNbeepState	PosENABLE
PosNiTFLength1	0
PosNqueueAllLabels	PosENABLE

## IBM 4686 Retail Point of Sale Scanner (All Models)

The default resource values for all IBM 4686 Retail Point of Sale Scanner models are:

<b>Resource</b>	<b>Default Value</b>
PosNbarCodes1	PosLGROUP_UPC_EAN_D1_TO_D5
PosNbarCodes2	PosLGROUP_NONE
PosNbeepFreq	PosHIGH
PosNbeepLength	PosSHORT
PosNbeepState	PosENABLE
PosNbeepVolume	PosHIGH
PosNblinkLength	PosLONG
PosNdReadTimeout	PosSHORT
PosNiTFLength1	0
PosNiTFLength2	0 (Models 3 & 4 only)
PosNlaserSwitchState	PosDISABLE
PosNlaserTimeout	15
PosNmotorTimeout	15
PosNqueueAllLabels	PosENABLE
PosNscansPerRead	2

## IBM 4687 Point of Sale Scanner (All Models)

The default resource values for the IBM 4687 Point of Sale Scanner Models 1 and 2 are:

<b>Resource</b>	<b>Default Value</b>
PosNbeepState	PosENABLE
PosNqueueAllLabels	PosENABLE

## IBM 4696 Point of Sale Scanner Scale Model 1

The default scanner resource values for the IBM 4696 Point of Sale Scanner Scale Model 1 are:

<b>Resource</b>	<b>Default Value</b>
PosNbarCodes1	PosLGROUP_UPC_EAN_D1_TO_D5
PosNbeepFreq	PosHIGH
PosNbeepState	PosENABLE
PosNbeepVolume	PosHIGH
PosNblinkLength	PosLONG
PosNbVolSwitchState	PosENABLE
PosNdecodeAlgorithm	PosENABLE
PosNdReadTimeout	PosSHORT
PosNeAN13ScansPerRead	2
PosNeAN8ScansPerRead	2
PosNstoreScansPerRead	2
PosNlaserTimeout	15
PosNmotorTimeout	15
PosNqueueAllLabels	PosENABLE
PosNuPCAScansPerRead	2
PosNuPCDScansPerRead	2
PosNuPCEscansPerRead	2
PosNuPCExpansion	NO_EXPANSION
PosNverifyPriceChk	NO_VERIFY

## IBM 4697 Point of Sale Scanner Model 1

The default resource values for the IBM 4697 Point of Sale Scanner Model 1 are:

<b>Resource</b>	<b>Default Value</b>
PosNbarCodes1	PosLGROUP_UPC_EAN_D1_TO_D5
PosNbarCodes2	PosLGROUP_NONE
PosNbeepFreq	PosHIGH
PosNbeepState	PosENABLE
PosNbeepVolume	PosHIGH
PosNblinkLength	PosLONG
PosNbVolSwitchState	PosENABLE
PosNdecodeAlgorithm	PosENABLE
PosNdReadTimeout	PosSHORT
PosNeAN13ScansPerRead	2
PosNeAN8ScansPerRead	2
PosNiTFLength1	0
PosNiTFLength2	0
PosNjANTwoLabelDecode	PosDISABLE
PosNlaserTimeout	15
PosNmotorTimeout	15
PosNqueueAllLabels	PosENABLE
PosNscansPerRead	2
PosNstoreScansPerRead	2

PosNtwoLabelFlagPair1	0x2122
PosNtwoLabelFlagPair2	0x2128
PosNtwoLabelFlagPair3	0x2129
PosNtwoLabelFlagPair4	0x2122
PosNuPCAScansPerRead	2
PosNuPCDScansPerRead	2
PosNuPCEscansPerRead	2
PosNuPCExpansion	NO_EXPANSION
PosNverifyPriceChk	NO_VERIFY

## IBM 4698 Point of Sale Scanner (All Models)

The default resource values for the IBM 4698 Point of Sale Scanner are:

<b>Resource</b>	<b>Default Value</b>
PosNbarCodes1	PosLGROUP_UPC_EAN_D1_TO_D5
PosNbarCodes2	PosLGROUP_NONE
PosNbarCodes3	PosLGROUP_NONE
PosNbarCodes4	PosLGROUP_NONE
PosNbeepState	PosENABLE
PosNbeepVolume	PosHIGH
PosNblinkLength	PosLONG
PosNbVolSwitchState	PosENABLE
PosNcode128ScansPerRead	2
PosNcode39ScansPerRead	2
PosNdecodeAlgorithm	PosENABLE
PosNdReadTimeout	PosSHORT
PosNeAN13ScansPerRead	2
PosNeAN8ScansPerRead	2
PosNiTFLength1	0
PosNiTFLength2	0
PosNiTFScansPerRead	2
PosNjANTwoLabelDecode	PosDISABLE
PosNlaserTimeout	15
PosNmotorTimeout	15
PosNqueueAllLabels	PosENABLE
PosNstoreScansPerRead	2
PosNtwoLabelFlagPair1	0x2122
PosNtwoLabelFlagPair2	0x2128
PosNtwoLabelFlagPair3	0x2129
PosNtwoLabelFlagPair4	0x2122
PosNuPCAScansPerRead	2
PosNuPCDScansPerRead	2
PosNuPCEscansPerRead	2
PosNuPCExpansion	NO_EXPANSION
PosNverifyPriceChk	NO_VERIFY

## IBM USB Scanner Interface

The default resource values for the IBM USB Scanner Interface are:

<b>Resource</b>	<b>Default Value</b>
PosNbarCodes1	PosLGROUP_UPC_EAN_D1_TO_D5
PosNbarCodes2	PosLGROUP_NONE
PosNbarCodes3	PosLGROUP_NONE
PosNbarCodes4	PosLGROUP_NONE
PosNbarCodeProgramming	PosENABLE
PosNbeepState	PosENABLE

PosNbeepVolume	PosHIGH
PosNblinkLength	PosLONG
PosNbVolSwitchState	PosENABLE
PosNdReadTimeout	PosSHORT
PosNiTFLength1	0
PosNiTFLength2	0
PosNiTFLengthType	PosDISCRETE
PosNjANTwoLabelDecode	PosDISABLE
PosNlaserTimeout	15
PosNmotorTimeout	15
PosNqueueAllLabels	PosENABLE
PosNstoreScansPerRead	2
PosNsupplementals	NO_SUPPLEMENTALS
PosNtransmitCheckDigit	UPCE_UPCA_CHECK_DIGIT
PosNtwoLabelFlagPair1	0x2122
PosNtwoLabelFlagPair2	0x2128
PosNtwoLabelFlagPair3	0x2129
PosNtwoLabelFlagPair4	0x2122
PosNuPCExpansion	NO_EXPANSION
PosNverifyPriceChk	NO_VERIFY

---

## Related Information

Additional information about scanner programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOCtl() Requests
- Chapter 20. Event Messages
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with Scanner

See Chapter 18. Application Programming Interface:

```
PosClose()
PosIOCtl()
PosOpen()
PosRead()
PosWrite()
```

## Scanner PosIOCtl() Control Requests

See Chapter 19. PosIOCtl() Requests:

```
POS_SCAN_DISCARD_DATA
POS_SYS_ACQUIRE_DEVICE
POS_SYS_LOCK_DEVICE
POS_SYS_GET_VALUES
POS_SYS_RELEASE_DEVICE
POS_SYS_SET_VALUES
POS_SYS_UNLOCK_DEVICE
```

## Scanner Event Messages

See Chapter 20. Event Messages:

```
POSM_SCAN_DATA_AVAIL
```

### Scanner Resources

See Chapter 21. Resource Sets:

- PosNbarCodes1
- PosNbarCodes2
- PosNbarCodes3
- PosNbarCodes4
- PosNbarCodeProgramming
- PosNbeepFreq
- PosNbeepLength
- PosNbeepState
- PosNbeepVolume
- PosNblinkLength
- PosNblockReadMode
- PosNblock1Type
- PosNblock2Type
- PosNblock3Type
- PosNbVoISwitchState
- PosNcheckModulo
- PosNcode128ScansPerRead
- PosNcode39ScansPerRead
- PosNdecodeAlgorithm
- PosNdReadTimeout
- PosNdTouchMode
- PosNeAN13ScansPerRead
- PosNeAN8ScansPerRead
- PosNiTFLength1
- PosNiTFLength2
- PosNiTFLengthType
- PosNiTFScansPerRead
- PosNjANTwoLabelDecode
- PosNlabelsQueued
- PosNlaserSwitchState
- PosNlaserTimeout
- PosNmotorTimeout
- PosNqueueAllLabels
- PosNtwoLabelFlagPair1
- PosNtwoLabelFlagPair2
- PosNtwoLabelFlagPair3
- PosNtwoLabelFlagPair4
- PosNscansPerRead
- PosNstoreScansPerRead
- PosNsupplementals
- PosNtransmitCheckDigit
- PosNuPCAScansPerRead
- PosNuPCDScansPerRead
- PosNuPCEscansPerRead
- PosNuPCExpansion
- PosNverifyPriceChk

### Scanner Error Codes

See Appendix D. Error Codes:

- 5737 POSERR\_SCAN\_2\_LABEL\_FLAG\_CONFIG\_ERROR
- 5736 POSERR\_SCAN\_CONFIGURATION\_ERROR
- 5738 POSERR\_SCAN\_INVALID\_2\_LABEL\_DECODE\_STATE
- 5740 POSERR\_SCAN\_INVALID\_2\_LABEL\_FLAG\_PAIR\_1

5741 POSERR\_SCAN\_INVALID\_2\_LABEL\_FLAG\_PAIR\_2  
5742 POSERR\_SCAN\_INVALID\_2\_LABEL\_FLAG\_PAIR\_3  
5743 POSERR\_SCAN\_INVALID\_2\_LABEL\_FLAG\_PAIR\_4  
5752 POSERR\_SCAN\_INVALID\_BARCODE\_PROG\_STATE  
5702 POSERR\_SCAN\_INVALID\_BAR\_CODES\_1  
5703 POSERR\_SCAN\_INVALID\_BAR\_CODES\_2  
5747 POSERR\_SCAN\_INVALID\_BAR\_CODES\_3  
5748 POSERR\_SCAN\_INVALID\_BAR\_CODES\_4  
5704 POSERR\_SCAN\_INVALID\_BEEP\_FREQ  
5705 POSERR\_SCAN\_INVALID\_BEEP\_LENGTH  
5706 POSERR\_SCAN\_INVALID\_BEEP\_STATE  
5707 POSERR\_SCAN\_INVALID\_BEEP\_VOLUME  
5708 POSERR\_SCAN\_INVALID\_BLINK\_LENGTH  
5710 POSERR\_SCAN\_INVALID\_BLOCK\_1\_TYPE  
5711 POSERR\_SCAN\_INVALID\_BLOCK\_2\_TYPE  
5712 POSERR\_SCAN\_INVALID\_BLOCK\_3\_TYPE  
5709 POSERR\_SCAN\_INVALID\_BLOCK\_READ\_MODE  
5725 POSERR\_SCAN\_INVALID\_BVOL\_SWITCH\_STATE  
5713 POSERR\_SCAN\_INVALID\_CHECK\_MODULO  
5753 POSERR\_SCAN\_INVALID\_XMIT\_CHECK\_DIGIT  
5746 POSERR\_SCAN\_INVALID\_CODE128\_SCANS\_PER\_READ  
5744 POSERR\_SCAN\_INVALID\_CODE39\_SCANS\_PER\_READ  
5714 POSERR\_SCAN\_INVALID\_D\_READ\_TIMEOUT  
5715 POSERR\_SCAN\_INVALID\_D\_TOUCH\_MODE  
5726 POSERR\_SCAN\_INVALID\_DECODE\_ALGORITHM  
5727 POSERR\_SCAN\_INVALID\_EAN13\_SCANS\_PER\_READ  
5728 POSERR\_SCAN\_INVALID\_EAN8\_SCANS\_PER\_READ  
5739 POSERR\_SCAN\_INVALID\_FLAG\_PAIR\_COMBINATION  
5716 POSERR\_SCAN\_INVALID\_ITF\_LENGTH\_1  
5717 POSERR\_SCAN\_INVALID\_ITF\_LENGTH\_2  
5750 POSERR\_SCAN\_INVALID\_ITF\_LENGTH\_TYPE  
5745 POSERR\_SCAN\_INVALID\_INT2OF5\_SCANS\_PER\_READ  
5720 POSERR\_SCAN\_INVALID\_LASER\_SWITCH\_STATE  
5718 POSERR\_SCAN\_INVALID\_LASER\_TIMEOUT  
5719 POSERR\_SCAN\_INVALID\_MOTOR\_TIMEOUT  
5735 POSERR\_SCAN\_INVALID\_QUEUE\_ALL\_INDICATOR  
5721 POSERR\_SCAN\_INVALID\_SCANS\_PER\_READ  
5729 POSERR\_SCAN\_INVALID\_STORE\_SCANS\_PER\_READ  
5751 POSERR\_SCAN\_INVALID\_SUPPLEMENTALS  
5730 POSERR\_SCAN\_INVALID\_UPCA\_SCANS\_PER\_READ  
5731 POSERR\_SCAN\_INVALID\_UPCD\_SCANS\_PER\_READ  
5732 POSERR\_SCAN\_INVALID\_UPCE\_SCANS\_PER\_READ  
5733 POSERR\_SCAN\_INVALID\_UPC\_EXPANSION  
5734 POSERR\_SCAN\_INVALID\_VERIFY\_PRICE\_CHK  
5722 POSERR\_SCAN\_LABEL\_TOO\_SHORT



---

## Chapter 17. Touch Screen Programming

The touch screen device handler is software that manages communication between your application and the touch screen. Your application can access the touch and tone components of the device only through the device handler.

**Note:** The Touch device is not supported on the IBM Point of Sale Subsystem for Linux.

---

### Characteristics of the Touch Screen

The IBM Point of Sale Subsystem supports the following point-of-sale touch screen terminals:

- IBM 4695 Point of Sale Distributed Touch Terminal Model 002
- IBM 4695 Point of Sale Distributed Touch Terminal Model 012
- IBM 4695 Point of Sale Distributed Touch Terminal Model 022
- IBM 4695 Point of Sale Distributed Touch Terminal Model 032
- IBM 4695 Point of Sale Integrated Touch Terminal Model 201
- IBM 4695 Point of Sale Integrated Touch Terminal Model 211
- IBM 4695 Point of Sale Integrated Touch Terminal Model 321
- IBM 4695 Point of Sale Integrated Touch Terminal Model 322
- IBM 4695 Point of Sale Integrated Touch Terminal Model 331
- IBM 4695 Point of Sale Integrated Touch Terminal Model 342
- IBM 4695 Point of Sale Integrated Touch Terminal Model N43
- IBM SurePoint Monochrome Touch Screen
- IBM SurePoint Color Touch Screen
- IBM 4820 SurePoint Solution Color Touch Screen Model 46T, 46R

**Note:** The 4820 SurePoint Solution keypad comes online even if a physical keypad is not attached. This behavior affects the enumeration of attached POS keyboards depending on their USB port location. For example, attaching a POS Keyboard downstream from the 4820 display enumerates the keyboard as a secondary POS keyboard. Attaching a POS Keyboard upstream from the 4820 display enumerates the keyboard as a primary POS keyboard.

The touch screen device is made up of three components:

- Video Display
- Touch Input
- Tone Output

#### Video Display

The video display component of the touch screen is supported by the operating system video device driver. The IBM Point of Sale Subsystem touch screen device handler provides application programs with the ability to control contrast, backlight, and brightness by using the **PosNtouchContrast**, **PosNtouchScreenSaverTime**, and **PosNtouchBrightness** resources.

#### Touch Input

The touch-sensitive input component is presented to your application as a grid of horizontal and vertical units. Position information is reported as x,y coordinates numbered 0 (zero) through **PosNtouchMaxX** for the horizontal units and 0 (zero) through **PosNtouchMaxY** for the vertical units. The coordinates 0,0 represent the upper-left corner of the touch screen.

All information about a touch event is contained in the event message. The touch device handler provides event messages for the following conditions:

- A touch becomes active (touch down)
- The active touch coordinates have changed
- A touch becomes inactive (lift off)

### Touch Screen Microcode Updates

The internal microprocessor code for the touch screen device might be updated by IBM. IBM delivers the microprocessor code update in a file named AIPxxx.TCH, where xxx is the microprocessor engineering change (EC) level. For OS/2, this file must be in a directory specified in the DPATH statement in the CONFIG.SYS file in order to be found and applied by the IBM Point of Sale Subsystem for OS/2. For the Microsoft Windows operating system, this file must be in the IBM Point of Sale Subsystem for Windows BIN directory in order to be found and applied by the IBM Point of Sale Subsystem for Windows.

## Tone Output

The tone component of the touch screen device is similar in function and application to the keyboard tone. The tone's volume, frequency, and duration can be controlled by the application program.

## Touch Mouse Emulation

The IBM Point of Sale Subsystem provides a feature which allows the touch screen to emulate a mouse. This feature sends mouse events to the operating system whenever the touch screen is touched, moved, or released. The operating system will then send to your application mouse events that correspond to the touch events.

**Note:** The IBM Point of Sale Subsystem touch mouse emulation is designed to work with the default value of the PosNtouchMode resource provided by the IBM Point of Sale Subsystem. If you change the value of the PosNtouchMode resource from its default, it will have no effect unless Mouse Emulation is turned off.

The IBM Point of Sale Subsystem has two different implementations of the touch mouse emulation feature. One is for OS/2 and the other is for the Microsoft Windows operating system.

### Touch Mouse Emulation on OS/2

For the IBM Point of Sale Subsystem for OS/2, touch mouse emulation is accomplished through the AIPPOINT.EXE program and AIPPOINT.SYS driver. Touch mouse emulation is an optional installation feature.

### Touch Mouse Emulation on the Microsoft Windows Operating System

For the IBM Point of Sale Subsystem for Windows, the touch mouse emulation is integrated into the IBM Point of Sale Subsystem for Windows. When the IBM Point of Sale Subsystem for Windows starts up, the touch mouse emulation will also start. If you do not want Touch Mouse Emulation, you can turn it off by creating a file called, TOUCH.OFF in the C:\POS directory.

**Double-click Sensitivity Adjustment:** In IBM Point of Sale Subsystem for Windows Version 1.4.2, and later, IBM Point of Sale Subsystem no longer manages double-click sensitivity. The adjustable double-click parameters are height, width, and speed which can all be modified through the Windows operating systems. The double-click speed is adjusted through the Control Panel Mouse applet; double-click

height and width are adjusted through either the WIN.INI file or the registry. The height and width parameters are string values called `DoubleClickHeight` and `DoubleClickWidth` which specify the number of pixels within which a second touch must occur to be considered a double-click. The default value for both `DoubleClickHeight` and `DoubleClickWidth` is 4; values between 20 and 40 are suggested for the IBM Touch devices. The following table describes where the double-click parameters are located for the various Windows platforms.

Table 17-1. *DoubleClickHeight and DoubleClickWidth Adjustment for Windows*

Windows Version	DoubleClickHeight and DoubleClickWidth Location
Windows 3.1x	WIN.INI in [Windows]
Windows 95/98	Registry in \Control Panel\Desktop
Windows NT	Registry in \My Computer\HKEY_CURRENT_USER\Control Panel\Mouse

## Restriction of the Touch Screen Device Handler

The touch screen device handler does not support the *PosWrite()* subroutine. In addition, it does not support the *PosRead()* subroutine for the device descriptor that you receive from the *PosOpen()* subroutine. The reason is that all touch input is received from the presentation facility input queue (for example, from the Presentation Manager event queue) or by issuing a *PosRead()* subroutine for the IBM Point of Sale Subsystem input queue.

---

## Functions Your Application Performs

Your application can perform the following functions with the touch screen:

- Read touch event data
- Use the tone
- Control audible feedback
- Control the LCD backlight
- Control the LCD brightness
- Control the LCD contrast
- Determine which Touch Screen is Available

## Reading Touch Event Data

Your application reads all touch screen input from the presentation facility input queue or by issuing *PosRead()* for the IBM Point of Sale Subsystem input queue. All of the information about a touch event is contained in the queue message. No other *PosRead()* call is required to get additional information. See “Getting Input Messages” on page 5-2 for more information.

The touch screen device descriptor obtained from the *PosOpen()* subroutine should not be used on the *PosRead()* subroutine. If it is, the 311 `POSERR_SYS_FUNCTION_NOT_SUPPORTED` error is set in *errno*.

## Using the Tone

The internal touch screen speaker is sounded by issuing the `POS_TOUCH_SOUND_TONE` *PosIOctl()* request. The frequency, duration, and volume of the tone are determined by the values of the **PosNtouchToneDuration**, **PosNtouchToneFreq**, and **PosNtouchToneVolume** resources. Use any of the following methods to specify values for these resources:

- Specify the value in the resource file.
- Set the value using the `POS_SYS_SET_VALUES` *PosIOctl()* request.
- Use the *args* and *nargs* parameters of the *PosOpen()* subroutine call.

If your application sounds the tone with the **PosNtouchToneDuration** resource set to **PosON**, it must issue the `POS_TOUCH_SILENCE_TONE PosIOctl()` request to silence the tone.

## Controlling Audible Feedback

Audible touch feedback can be produced through the internal speaker of the touch screen using:

- **PosNtouchEntryClick** resource (Click when a touch becomes active, also referred to as touch-down).
- **PosNtouchExitClick** resource (Click when a touch becomes inactive, also referred to as lift-off).

In addition, the volume of the click can be set with the **PosNtouchClickVolume** resource. Use any of the following methods to specify values for these resources:

- Specify the value in the resource file.
- Set the value using the `POS_SYS_SET_VALUES PosIOctl()` request.
- Use the *args* and *nargs* parameters of the `PosOpen()` subroutine call.

## Determining which Touch Screen is Available

Your application determines which touch display is attached to the point-of-sale terminal by looking at the *subtype* field in the `POSM_SYS_DEVICE_ONLINE` event message.

Table 17-2. Determining which Touch Screen is available

subtype	Touch Display
PosTOUCH_SUBTYPE_4695_104_M	10.4-inch 4695 monochrome display
PosTOUCH_SUBTYPE_4695_95_C	9.5-inch 4695 color display
PosTOUCH_SUBTYPE_4695_104_C	10.4-inch 4695 color display
PosTOUCH_SUBTYPE_4695_121_C	12.1-inch 4695 color display
PosTOUCH_SUBTYPE_SUREPOINT_95_M	9.5-inch SurePoint monochrome display
PosTOUCH_SUBTYPE_SUREPOINT_95_C	9.5-inch SurePoint color display
PosTOUCH_SUBTYPE_SUREPOINT_104_C	10.4-inch SurePoint color display
PosTOUCH_SUBTYPE_4820	4820 Touch Display

## Controlling the LCD Brightness

On models that support programatic control of brightness, your application can set the brightness to 8 different levels (0-7) by using the **PosNtouchBrightness** resource. Other models will ignore the setting of this resource.

See “PosNtouchBrightness” on page 21-90 for more information

## Controlling the LCD Contrast

On models that support programatic control of contrast, your application can set the contrast to 64 different levels (0-63) by using the **PosNtouchContrast** resource. Other models will ignore the setting of this resource.

See “PosNtouchContrast” on page 21-91 for more information

## Controlling the Screen Saver Time

To protect the touch display screen, the IBM 4695 Point of Sale Terminals. will automatically dim themselves after a predefined amount of time has elapsed without any activity. Your application can control the amount of time of inactivity before the screen is dimmed by setting the **PosNtouchScreenSaverTime** resource. The minimum value is 1 (one) second and the maximum is 65,535 seconds. However, it is not recommended that the value be set below 60 seconds.

See “PosNtouchScreenSaverTime” on page 21-93 for more information

## Controlling the Backlight On Event Messages

If you set the **PosNtouchBackLightOnEvent** resource to PosENABLE, your application will receive a POSM\_TOUCH\_DATA event message when the screen is touched to refresh the display. If you set it to PosDISABLE, no event message is sent to your application when the screen is being refreshed.

See “PosNtouchBackLightOnEvent” on page 21-90 for more information

---

## Related Information

Additional information about touch programming is in these chapters:

- Chapter 5. General Point of Sale Device Programming
- Chapter 18. Application Programming Interface
- Chapter 19. PosIOCtl() Requests
- Chapter 20. Event Messages
- Chapter 21. Resource Sets
- Appendix D. Error Codes

## Subroutines Used with the Touch Screen

See Chapter 18. Application Programming Interface:

PosClose()  
PosIOCtl()  
PosOpen()  
PosRead()

## Touch PosIOCtl Control Requests

See Chapter 19. PosIOCtl() Requests:

POS\_TOUCH\_SILENCE\_TONE  
POS\_TOUCH\_SOUND\_TONE  
POS\_SYS\_ACQUIRE\_DEVICE  
POS\_SYS\_GET\_VALUES  
POS\_SYS\_RELEASE\_DEVICE  
POS\_SYS\_SET\_VALUES

## Touch Event Messages

See Chapter 20. Event Messages:

POSM\_TOUCH\_DATA

## Touch Resources

See Chapter 21. Resource Sets:

PosNtouchBackLightOnEvent  
PosNtouchBrightness  
PosNtouchClickVolume

## Touch Screen Programming

created on October 2, 2001

PosNtouchContrast  
PosNtouchEntryClick  
PosNtouchExitClick  
PosNtouchMaxX  
PosNtouchMaxY  
PosNtouchMode  
PosNtouchScreenSaverTime  
PosNtouchToneDuration  
PosNtouchToneFreq  
PosNtouchToneVolume

## Touch Error Codes

See Appendix D. Error Codes:

6701 POSERR\_TOUCH\_INVALID\_BACKLIGHT\_ON  
6711 POSERR\_TOUCH\_INVALID\_BRIGHTNESS  
6702 POSERR\_TOUCH\_INVALID\_CLICK\_VOLUME  
6703 POSERR\_TOUCH\_INVALID\_CONTRAST  
6704 POSERR\_TOUCH\_INVALID\_ENTRY\_CLICK  
6705 POSERR\_TOUCH\_INVALID\_EXIT\_CLICK  
6706 POSERR\_TOUCH\_INVALID\_MODE  
6707 POSERR\_TOUCH\_INVALID\_SCREEN\_SAVER\_TIME  
6708 POSERR\_TOUCH\_INVALID\_TONE\_DURATION  
6709 POSERR\_TOUCH\_INVALID\_TONE\_FREQUENCY  
6710 POSERR\_TOUCH\_INVALID\_TONE\_VOLUME

---

## Chapter 18. Application Programming Interface

This chapter describes the subroutines provided by the IBM Point of Sale Subsystem that are used to provide program control of point-of-sale devices. The following subroutines are available to the application:

- PosClose()
- PosInitialize()
- PosIOctl()
- PosOpen()
- PosRead()
- PosWrite()

Your application's use of the IBM Point of Sale Subsystem application programming interface (API) will vary from device to device. The following describes a general API call sequence that varies depending on the device being controlled:

1. Your application calls the *PosInitialize()* subroutine to register with the IBM Point of Sale Subsystem.
2. Your application creates a device connection by opening the device with the *PosOpen()* subroutine. Each device connection established by the application is assigned a unique identifier called the *device descriptor*. This device descriptor is used on all subsequent calls to the IBM Point of Sale Subsystem for that device.
3. Your application acquires exclusive use of a device by using a POS\_SYS\_ACQUIRE\_DEVICE *PosIOctl()* request. Notice that all devices require that an application process acquire the device before using *PosRead()* or *PosWrite()*, and most *PosIOctl()* requests.
4. Your application uses the different *PosIOctl()* requests to set up the device resource values and to control the device. Your application is not required to set the device resources explicitly when one of the following conditions is true:
  - If the default value for the resource is acceptable and if a different value is not set in the resource file.
  - If the value specified in the resource file is acceptable.
5. Your application uses the point-of-sale device.
6. When your application no longer requires exclusive use of a device, the application should issue a POS\_SYS\_RELEASE\_DEVICE *PosIOctl()* request. This allows other applications to use the device.

A typical closing sequence for an application is as follows:

1. Use the POS\_SYS\_RELEASE\_DEVICE *PosIOctl()* request to release each device that was acquired.
2. Use the *PosClose()* subroutine to close any devices that the application opened.

Part 2. Programming Guide describes how your application program can use the IBM Point of Sale Subsystem devices.

**Important:** No more than one IBM Point of Sale Subsystem API subroutine can be active at the same time. When an IBM Point of Sale Subsystem application programming interface subroutine is called by one thread in a process, IBM Point of Sale Subsystem API calls by any other thread in the same process are blocked (prevented from processing) until the first call is complete. When the first call completes, one of the waiting calls is processed.

The only exception to this is when the *PosRead()* subroutine is called for the IBM Point of Sale Subsystem input queue (device descriptor zero). The **PosNreadTimeout** resource allows a read operation on the IBM Point of Sale Subsystem input queue to be suspended until data is available to be read. When one thread is waiting for data on the input queue, other threads can successfully call IBM Point of Sale Subsystem API subroutines for other device descriptors and not be blocked. Even with this exception, only one of the other IBM Point of Sale Subsystem API calls can be active simultaneously.

---

## PosClose()

### Purpose

Closes the device connection associated with a device descriptor.

### Syntax

```
#include <pos/pos.h>

int PosClose( int devdes ) ;
```

### Parameters

*devdes*

Specifies a valid device descriptor. This descriptor identifies a connection to a device. This descriptor must have been previously created using the *PosOpen()* subroutine.

### Description

The *PosClose()* subroutine closes the connection associated with the *devdes* parameter. An application should call this subroutine for each device connection it opened before it terminates.

If the *PosClose()* is issued for a device descriptor that has acquired exclusive use of the device, the device is released before the device connection is closed.

All open device connections are closed when a process exits.

This subroutine can be used only after the process has been initialized and a target device has been opened.

**Note:** If an error occurs while closing a connection, the error is returned to the application. However, the connection is considered closed and the *devdes* parameter cannot be used in any subsequent calls.

### Return Values

Upon successful completion, a value of 0 (zero) is returned. Otherwise, a value of -1 is returned and *errno* is set to indicate the error.

### Error Codes

If the *PosClose()* subroutine fails, *errno* is set to one of the following:

<b>302</b>	302 POSERR_SYS_NOT_INITIALIZED
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>311</b>	311 POSERR_SYS_FUNCTION_NOT_SUPPORTED
<b>320</b>	320 POSERR_SYS_DATA_DISCARDED
<b>321</b>	321 POSERR_SYS_INTERNAL_ERROR
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

### Related Information

“PosOpen()” on page 18-10.

**Examples**

The following example opens and then closes the printer.

```
#include <pos/pos.h>

int  prndes;                               /* printer device descriptor */
int  rc = 0;

rc = PosInitialize( "myappl", "checkout", 0, 0, 0, 0 );

prndes = PosOpen( "myprinter", PosPrinter, 0,0 );

.
.
.

rc = PosClose( prndes );
```

---

## PosInitialize()

### Purpose

Initializes the IBM Point of Sale Subsystem for use by the application.

### Syntax

```
#include <pos/pos.h>
```

```
int PosInitialize(char *name, char *file,  
                 int *argc, char *argv[],  
                 PosArgPtr args, int nargs) ;
```

### Parameters

- name* Specifies a name for this application. This name is used to identify resources in the resource file. Typically, *argv[0]* is given. This parameter points to a null terminated string of not more than 32 characters.
- file* Specifies the resource file for this application. This parameter can be used to group multiple applications together for sharing the same resource file. See “Configuring Your Applications” on page 3-1 for an explanation of how the *file* parameter is used in determining the resource file. This parameter points to a null terminated string. The IBM Point of Sale Subsystem looks for a file with this name in the directory specified in the environment variable `POSAPPLRESDIR`. If this environment variable is not set, the current directory is used.
- argc* Specifies a pointer to the number of command line parameters. This parameter is reserved for future use. Set this value to 0 (zero).
- argv* Specifies the command line parameters. This parameter is reserved for future use. Set this value to 0 (zero).
- args* Specifies a parameter list of system resources.
- nargs* Specifies the number of parameters in the parameter list. If this value is 0 (zero), *args* is ignored.

### Description

The *PosInitialize()* subroutine initializes the IBM Point of Sale Subsystem for use by the application. Once any thread within a process issues this call, all threads for that process are initialized. Therefore, *PosInitialize()* must be called only once for each process.

*PosInitialize()* must be successfully called before other IBM Point of Sale Subsystem subroutines are used.

Null strings or a null pointer for the *name* or *file* parameters generates an error.

The following resources can be specified on the argument list of the *PosInitialize()* subroutine:

- **PosNqueueHandle**
- **PosNreadTimeout**

The **PosNqueueHandle** resource specifies the queue which receives all system-wide event messages, such as `POSM_SYS_DEVICE_ONLINE` and

POSM\_SYS\_DEVICE\_OFFLINE. These are event messages that are not necessarily associated with a particular opened device. All applications that have initialized the IBM Point of Sale Subsystem receive these event messages. See “PosNqueueHandle” on page 21-6 for more information on this resource.

**Note:** The **PosNqueueHandle** resource is ignored on systems that use the IBM Point of Sale Subsystem for Linux. Linux systems must use the IBM Point of Sale Subsystem input queue.

The **PosNreadTimeout** resource specifies how long a *PosRead()* subroutine for the IBM Point of Sale Subsystem input queue should wait for an event message. See “PosNreadTimeout” on page 21-6 for more information on this resource.

## Return Values

Upon successful completion, the *PosInitialize()* subroutine returns a value of 0 (zero). Otherwise, it returns a value of -1 and *errno* is set to indicate the error.

**Note:** If the resource file is not found or cannot be read, no error is returned to the application. The default values and the values specified by your application will be used.

## Error Codes

If the *PosInitialize()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>304</b>	304 POSERR_SYS_ALREADY_INITIALIZED
<b>305</b>	305 POSERR_SYS_MEMORY_ALLOCATION
<b>330</b>	330 POSERR_SYS_INVALID_NAME
<b>334</b>	334 POSERR_SYS_INVALID_ADDRESS
<b>336</b>	336 POSERR_SYS_INVALID_FILE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

## Related Information

“PosOpen()” on page 18-10.

## Examples

The following example registers an application with the IBM Point of Sale Subsystem. The application receives the system event messages (such as on-line or off-line) in the IBM Point of Sale Subsystem input queue (device descriptor zero).

```
#include <pos/pos.h>

int rc = 0;

rc = PosInitialize( "myappl", "checkout", 0, 0, 0, 0 );
```

The following example registers an application with the IBM Point of Sale Subsystem. The application receives the system event messages (such as on-line or off-line) on the Presentation Manager event queue.

```
#define INCL_WIN

#include <os2.h>
#include <pos/pos.h>

HAB hab; /* Anchor block handle */
HMq hmq; /* Message queue handle */
```

```
PosArg  resource;          /* resource name and value */
int    i, rc;

hab = WinInitialize( 0 );

hmq = WinCreateMsgQueue( hab, 0 );

resource.name = PosNqueueHandle; /* resource name is queueHandle */
resource.value = hmq;           /* resource value is in hmq */

rc = PosInitialize( "myappl", "checkout", 0, 0, &resource, 1 );
```

---

## PosIOctl()

### Purpose

Performs input/output control functions associated with device descriptors.

### Syntax

```
#include <pos/pos.h>
```

```
int PosIOctl( int devdes, int request, PosArgPtr args, int nargs ) ;
```

### Parameters

<i>devdes</i>	Specifies the device descriptor the control operation is to be performed for. This descriptor identifies a connection to a device. This descriptor must have been previously created using the <i>PosOpen()</i> subroutine.
<i>request</i>	Specifies the control function to be performed. The value of this parameter depends on which device is specified by the <i>devdes</i> parameter.
<i>args</i>	Specifies the argument list required by the control operation specified by the <i>request</i> parameter, or an argument list, to override the resource default values.
<i>nargs</i>	Specifies the number of parameters in the parameter list. If this value is 0 (zero), <i>args</i> is ignored.

### Description

The *PosIOctl()* subroutine performs a variety of control operations on the device associated with the specified device descriptor.

The control operation provided by this subroutine is specific to the device being addressed, as are the contents of the parameters passed. See the individual device chapters in Part 2. Programming Guide for more information.

The *args* and *nargs* parameters are used to specify undefined resource values or to override resource values defined in the resource table. Requests that use these parameters list the resources that they use in the "Resources Used" section of their description. Only the resources associated with the *PosIOctl()* request for the device are validated.

This subroutine can be used only after the process has been initialized and a target device has been opened.

### Return Values

Upon successful completion, a value of 0 (zero) is returned. If the *PosIOctl()* subroutine fails, a value of -1 is returned, and *errno* is set to indicate the error.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>302</b>	302 POSERR_SYS_NOT_INITIALIZED
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED

```

316 316 POSERR_SYS_INVALID_REQUEST
325 325 POSERR_SYS_INVALID_NARGS
332 332 POSERR_SYS_INTERRUPTED
334 334 POSERR_SYS_INVALID_ADDRESS
337 337 POSERR_SYS_SERVICE_NOT_AVAILABLE

```

Additional error codes are listed with each individual control function.

## Related Information

“PosOpen()” on page 18-10.

Chapter 19. PosIOctl() Requests.

## Examples

This example opens the keyboard, acquires the keyboard, and sounds the keyboard tone for 1 second. The device is released at the end of the example.

```

#include <pos/pos.h>

int kbddes;
int rc = 0;
PosArg resource; /* tone duration resource */

rc = PosInitialize( "myappl", "checkout", 0, 0, 0, 0 );

kbddes = PosOpen( "mykbd", PosKeyboard, 0, 0 );

if ( -1 != kbddes )
{
    rc = PosIOctl( kbddes, POS_SYS_ACQUIRE_DEVICE, 0, 0 );
}

if ( 0 == rc )
{
    PosSetArg( resource, PosNtoneDuration, 10 );
    rc = PosIOctl( kbddes, POS_KBD_SOUND_TONE, &resource, 1 );
}

/*
perform some other processing
*/

rc = PosIOctl( kbddes, POS_SYS_RELEASE_DEVICE, 0, 0 );

```

---

## PosOpen()

### Purpose

Establishes a device connection.

### Syntax

```
#include <pos/pos.h>
```

```
int PosOpen( char *name, char * class, PosArgPtr args, int nargs ) ;
```

### Parameters

- name* Specifies the instance name for the created device connection. This name is chosen by the application and is used for retrieving resources. This parameter points to a null terminated string of not more than 32 characters.
- class* Specifies a device class name. The specified device class determines the type of device connection being opened and also determines what resources are associated with the device. This parameter must be one of the names defined by the IBM Point of Sale Subsystem. See the device resource set of the device being opened for the device class name.
- args* Specifies an argument list to override the resource defaults.
- nargs* Specifies the number of arguments in the argument list. If this value is 0 (zero), *args* is ignored.

### Description

The *PosOpen()* subroutine establishes a connection between the named device and your application. A device descriptor is returned to identify this connection. The device descriptor is used by subsequent IBM Point of Sale Subsystem subroutines, such as *PosRead()* and *PosWrite()*, to access the device. Therefore, no application programming interface subroutines except for the *PosInitialize()* subroutine can be issued before the device has been opened. Your application should open the devices you wish it to work with after receiving the POSM\_SYS\_DEVICE\_ONLINE event message. Any open requests issued prior to the device being reported on-line will fail with error code 317 POSERR\_SYS\_DEVICE\_OFFLINE.

To open a device, the following resources with the appropriate values must be specified to the IBM Point of Sale Subsystem:

- **PosNslotNumber**
- **PosNportNumber**
- **PosNdeviceNumber**

Your application can specify the values for these resources as arguments on the *PosOpen()* subroutine call or in a resource file used by your application.

After you open a device, you can use the POS\_SYS\_SET\_VALUES *PosIOctl()* subroutine request to perform setup operations on the device. However, the results of these setup operations will not take affect until the device has been acquired with the POS\_SYS\_ACQUIRE\_DEVICE *PosIOctl()* subroutine request.

Once any thread within a process opens a device, all threads for that process are allowed to use the device descriptor on other IBM Point of Sale Subsystem

application programming interface subroutine calls. There can be up to 31 open devices per process at a time. The device descriptor cannot be shared between processes.

The *args* and *nargs* parameters are used to override resource values defined in the resource file or the device's default resource values. Only the resources associated with this subroutine for the device are validated. The **PosNqueueHandle** resource can be specified on the argument list of the *PosOpen()* function. The queue specified by this resource receives any event messages that apply specifically to the device being opened.

**Note:** The **PosNqueueHandle** resource is ignored on systems that use the IBM Point of Sale Subsystem for Linux. Linux systems must use the IBM Point of Sale Subsystem input queue.

This subroutine can be used only after the process has been initialized.

## Return Values

Upon successful completion, the device descriptor (a non-negative integer) is returned. Otherwise, a value of -1 is returned and *errno* is set to indicate the error.

## Error Codes

If the *PosOpen()* subroutine fails, the named device connection is not established, and *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>302</b>	302 POSERR_SYS_NOT_INITIALIZED
<b>307</b>	307 POSERR_SYS_INVALID_DEVICE
<b>305</b>	305 POSERR_SYS_MEMORY_ALLOCATION
<b>309</b>	309 POSERR_SYS_TOO_MANY_DEVICES
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>319</b>	319 POSERR_SYS_INVALID_CLASS_DEVICE_COMBO
<b>320</b>	320 POSERR_SYS_DATA_DISCARDED
<b>321</b>	321 POSERR_SYS_INTERNAL_ERROR
<b>325</b>	325 POSERR_SYS_INVALID_NARGS
<b>326</b>	326 POSERR_SYS_INVALID_SLOT
<b>327</b>	327 POSERR_SYS_UNSUPPORTED_ADAPTER
<b>328</b>	328 POSERR_SYS_INVALID_PORT
<b>330</b>	330 POSERR_SYS_INVALID_NAME
<b>331</b>	331 POSERR_SYS_INVALID_CLASS
<b>334</b>	334 POSERR_SYS_INVALID_ADDRESS
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>4102</b>	4102 POSERR_NVRAM_INVALID_CURSOR
<b>4401</b>	4401 POSERR_DSP_INVALID_CURSOR
<b>4402</b>	4402 POSERR_DSP_INVALID_MODE
<b>4403</b>	4403 POSERR_DSP_INVALID_SIZE
<b>4406</b>	4406 POSERR_DSP_INVALID_CODE_PAGE
<b>4701</b>	4701 POSERR_KBD_INVALID_FREQUENCY
<b>4702</b>	4702 POSERR_KBD_INVALID_DURATION
<b>4703</b>	4703 POSERR_KBD_INVALID_VOLUME
<b>4705</b>	4705 POSERR_KBD_INVALID_DOUBLE_KEY
<b>4706</b>	4706 POSERR_KBD_INVALID_FAT_FINGER_TIMEOUT
<b>4708</b>	4708 POSERR_KBD_INVALID_KEYBOARD_CLICK
<b>4709</b>	4709 POSERR_KBD_INVALID_NUMPAD_STYLE
<b>4710</b>	4710 POSERR_KBD_INVALID_NUMPAD_ZERO
<b>4711</b>	4711 POSERR_KBD_INVALID_TYPEMATIC_DELAY

4712	4712 POSERR_KBD_INVALID_TYPEMATIC_FREQ
4713	4713 POSERR_KBD_INVALID_NUMPAD_LOCATION
4901	4901 POSERR_PRN_INVALID_DI_WIDTH
4902	4902 POSERR_PRN_INVALID_INTERLEAVED_VALUE
4903	4903 POSERR_PRN_INVALID_HEAD_PARKED_POSITION
4904	4904 POSERR_PRN_INVALID_STATION
4905	4905 POSERR_PRN_INVALID_MODE
4906	4906 POSERR_PRN_INVALID_CR_LF_DISTANCE
4907	4907 POSERR_PRN_INVALID_SJ_LF_DISTANCE
4908	4908 POSERR_PRN_INVALID_DI_LF_DISTANCE
4909	4909 POSERR_PRN_INVALID_FEED_DIRECTION
4910	4910 POSERR_PRN_INVALID_FISCAL_NOTIFY
4911	4911 POSERR_PRN_INVALID_DI_ORIENTATION
4912	4912 POSERR_PRN_INVALID_LEFT_MARGIN_CR
4913	4913 POSERR_PRN_INVALID_PRINT_ALIGNMENT
5401	5401 POSERR_CDR_INVALID_PULSE_WIDTH
5702	5702 POSERR_SCAN_INVALID_BAR_CODES_1
5703	5703 POSERR_SCAN_INVALID_BAR_CODES_2
5704	5704 POSERR_SCAN_INVALID_BEEP_FREQ
5705	5705 POSERR_SCAN_INVALID_BEEP_LENGTH
5706	5706 POSERR_SCAN_INVALID_BEEP_STATE
5707	5707 POSERR_SCAN_INVALID_BEEP_VOLUME
5708	5708 POSERR_SCAN_INVALID_BLINK_LENGTH
5709	5709 POSERR_SCAN_INVALID_BLOCK_READ_MODE
5710	5710 POSERR_SCAN_INVALID_BLOCK_1_TYPE
5711	5711 POSERR_SCAN_INVALID_BLOCK_2_TYPE
5712	5712 POSERR_SCAN_INVALID_BLOCK_3_TYPE
5713	5713 POSERR_SCAN_INVALID_CHECK_MODULO
5714	5714 POSERR_SCAN_INVALID_D_READ_TIMEOUT
5715	5715 POSERR_SCAN_INVALID_D_TOUCH_MODE
5716	5716 POSERR_SCAN_INVALID_ITF_LENGTH_1
5717	5717 POSERR_SCAN_INVALID_ITF_LENGTH_2
5718	5718 POSERR_SCAN_INVALID_LASER_TIMEOUT
5719	5719 POSERR_SCAN_INVALID_MOTOR_TIMEOUT
5720	5720 POSERR_SCAN_INVALID_LASER_SWITCH_STATE
5721	5721 POSERR_SCAN_INVALID_SCANS_PER_READ
5725	5725 POSERR_SCAN_INVALID_BVOL_SWITCH_STATE
5726	5726 POSERR_SCAN_INVALID_DECODE_ALGORITHM
5727	5727 POSERR_SCAN_INVALID_EAN13_SCANS_PER_READ
5728	5728 POSERR_SCAN_INVALID_EAN8_SCANS_PER_READ
5729	5729 POSERR_SCAN_INVALID_STORE_SCANS_PER_READ
5730	5730 POSERR_SCAN_INVALID_UPCA_SCANS_PER_READ
5731	5731 POSERR_SCAN_INVALID_UPCD_SCANS_PER_READ
5732	5732 POSERR_SCAN_INVALID_UPCE_SCANS_PER_READ
5733	5733 POSERR_SCAN_INVALID_UPC_EXPANSION
5734	5734 POSERR_SCAN_INVALID_VERIFY_PRICE_CHK
5735	5735 POSERR_SCAN_INVALID_QUEUE_ALL_INDICATOR
5736	5736 POSERR_SCAN_CONFIGURATION_ERROR
5737	5737 POSERR_SCAN_2_LABEL_FLAG_CONFIG_ERROR
5738	5738 POSERR_SCAN_INVALID_2_LABEL_DECODE_STATE
5739	5739 POSERR_SCAN_INVALID_FLAG_PAIR_COMBINATION
5740	5740 POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_1
5741	5741 POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_2
5742	5742 POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_3
5743	5743 POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_4
5744	5744 POSERR_SCAN_INVALID_CODE39_SCANS_PER_READ

```

5745 5745 POSERR_SCAN_INVALID_INT2OF5_SCANS_PER_READ
5746 5746 POSERR_SCAN_INVALID_CODE128_SCANS_PER_READ
5747 5747 POSERR_SCAN_INVALID_BAR_CODES_3
5748 5748 POSERR_SCAN_INVALID_BAR_CODES_4
5750 5750 POSERR_SCAN_INVALID_ITF_LENGTH_TYPE
5751 5751 POSERR_SCAN_INVALID_SUPPLEMENTALS
5752 5752 POSERR_SCAN_INVALID_BARCODE_PROG_STATE
5753 5753 POSERR_SCAN_INVALID_XMIT_CHECK_DIGIT
5901 5901 POSERR_RS232_INVALID_BAUD_RATE
5902 5902 POSERR_RS232_INVALID_STOP_BITS
5903 5903 POSERR_RS232_INVALID_PARITY
5904 5904 POSERR_RS232_INVALID_DATA_BITS
5905 5905 POSERR_RS232_INVALID_TIMEOUT_CHAR
6201 6201 POSERR_SCALE_INVALID_OPERATIONS_MODE
6202 6202 POSERR_SCALE_INVALID_REMOTE_DISPLAY_STATE
6203 6203 POSERR_SCALE_INVALID_VIBRATION_FILTER
6204 6204 POSERR_SCALE_INVALID_WEIGHT_MODE
6205 6205 POSERR_SCALE_INVALID_ZERO_INDICATOR_STATE
6206 6206 POSERR_SCALE_INVALID_ZERO_RETURN_STATE
6211 6211 POSERR_SCALE_INVALID_NUM_WEIGHT_DIGITS
6701 6701 POSERR_TOUCH_INVALID_BACKLIGHT_ON
6702 6702 POSERR_TOUCH_INVALID_CLICK_VOLUME
6703 6703 POSERR_TOUCH_INVALID_CONTRAST
6704 6704 POSERR_TOUCH_INVALID_ENTRY_CLICK
6705 6705 POSERR_TOUCH_INVALID_EXIT_CLICK
6706 6706 POSERR_TOUCH_INVALID_MODE
6707 6707 POSERR_TOUCH_INVALID_SCREEN_SAVER_TIME
6708 6708 POSERR_TOUCH_INVALID_TONE_DURATION
6709 6709 POSERR_TOUCH_INVALID_TONE_FREQUENCY
6710 6710 POSERR_TOUCH_INVALID_TONE_VOLUME
6711 6711 POSERR_TOUCH_INVALID_BRIGHTNESS

```

## Related Information

“PosClose()” on page 18-3.

## Examples

In this example, the *OpenDevice* function opens devices after the on-line event message has been received.

```

#include <pos/pos.h>

/*****
/* Global data */
/*****
int kbddes; /* keyboard device descriptor */
int msrdes; /* msr device descriptor */
int dspdes; /* display device descriptor */
int prndes; /* printer device descriptor */
int scandes; /* scanner device descriptor */
int tilldes; /* till device descriptor */
int alarmdes; /* alarm device descriptor */

/*****
/* Function Name: OpenDevice */
/* */
/* Description: This subroutine opens the device for which an on-line */
/* message has been received. */
/* */
/* Input: pMsg points to a POSM_SYS_DEVICE_ONLINE message */
/*****

```

```

/*****
void OpenDevice(POSQMSG *pMsg)
{
    PosArg  openargs[3];          /* PosArg array for PosOpen  */
    int  DevType  = 0;           /* Device type from ONLINE msg*/
    int  rc       = 0;           /* Return codes              */

    PosSetArg(openargs[0], PosNslotNumber, ((int)(pMsg->mp1 >> 24) & 0x00FF));
    PosSetArg(openargs[1], PosNportNumber, ((int)(pMsg->mp1 >> 16) & 0x00FF));
    PosSetArg(openargs[2], PosNdeviceNumber, ((int)(pMsg->mp1 >> 8) & 0x00FF));
    DevType = (int) pMsg->mp1 & 0x00FF;

    if ( POSM_SYS_DEVICE_ONLINE == pMsg->msg )
    {
        switch(DevType)
        {
            case PosTYPE_DISPLAY:
            {
                dspdes = PosOpen( "mydisplay", PosDisplay, openargs,
                                PosNumber(openargs) );
                break;
            } /* end case (PosTYPE_DISPLAY) */

            case PosTYPE_PRINTER:
            {
                prndes = PosOpen( "myprinter", PosPrinter, openargs,
                                PosNumber(openargs) );
                break;
            } /* end case (PosTYPE_PRINTER) */

            case PosTYPE_SCANNER:
            {
                scandes = PosOpen( "myscanner", PosScanner, openargs,
                                PosNumber(openargs) );
                break;
            } /* end case (PosTYPE_SCANNER) */

            case PosTYPE_MSR:
            {
                msrdes = PosOpen( "mymstr", PosMsr, openargs,
                                PosNumber(openargs) );
                break;
            } /* end case (PosTYPE_MSR) */

            case PosTYPE_KEYBOARD:
            {
                kbddes = PosOpen( "mykeyboard", PosKeyboard, openargs,
                                PosNumber(openargs) );
                break;
            } /* end case (PosTYPE_KEYBOARD) */

            case PosTYPE_CASH_DRAWER:
            {
                tilldes = PosOpen( "mytill", PosDrawer, openargs,
                                PosNumber(openargs) );
                break;
            } /* end case (PosTYPE_CASH_DRAWER) */

            case PosTYPE_ALARM:
            {
                alarmdes = PosOpen( "myalarm", PosAlarm, openargs,
                                PosNumber(openargs) );
                break;
            } /* end case (PosTYPE_ALARM) */

            default:
            {

```

```
        break;
    }

    } /* end switch (DevType) */
} /* end if ( POSM_SYS_DEVICE_ONLINE == pMsg->msg ) */
} /* end OpenDevice */
```

## PosRead()

### Purpose

Reads from a device.

### Syntax

```
#include <pos/pos.h>
```

```
ssize_t PosRead( int devdes, void *buf, size_t nbyte ) ;
```

### Parameters

*devdes*

Specifies a valid device descriptor. This descriptor identifies a connection to a device. This descriptor must have been previously created using the *PosOpen()* subroutine.

*buf* A pointer to a buffer where the data read from the device is placed.

*nbyte* The maximum number of bytes available to receive data at *buf*.

### Description

The *PosRead()* subroutine attempts to read *nbyte* bytes from the device associated with the device descriptor (*devdes*) into the buffer pointed to by *buf*. In most cases, if the device being read does not have any data to return, 0 (zero) is returned to indicate that no data was read. The exception is device descriptor zero, the IBM Point of Sale Subsystem input queue. The **PosNreadTimeout** resource allows the *PosRead()* subroutine call to be suspended until there is data on the input queue.

If *nbyte* is 0 (zero), the *PosRead()* subroutine returns 0 (zero) and has no other results.

Upon successful completion, the *PosRead()* subroutine returns the number of bytes placed in the buffer. This number is never greater than *nbyte*. The *PosRead()* subroutine can read the IBM Point of Sale Subsystem input queue by specifying a device descriptor of 0 (zero). The IBM Point of Sale Subsystem input queue is automatically opened and acquired for a process that does not specify a queue handle on the *PosInitialize()*.

This subroutine can only be used:

1. After initialization.
2. After a target device is opened and acquired, with the exception of reading the IBM Point of Sale Subsystem input queue.

### Return Values

Upon successful completion, *PosRead()* returns an integer indicating the number of bytes placed into the buffer. This number is never greater than *nbytes*. Otherwise, *PosRead()* returns a value of -1 and *errno* is set to indicate the error, and the contents of the buffer pointed to by *buf* are indeterminate.

For some types of devices, the number of bytes placed into the buffer is not the same as the number of bytes read from a device. There can be control information returned in the buffer in addition to the actual data read from a device. See the individual device chapters for the format of the buffer returned for a read operation.

## Error Codes

If the *PosRead()* subroutine fails, *errno* is set to one of the following:

<b>302</b>	302	POSERR_SYS_NOT_INITIALIZED
<b>303</b>	303	POSERR_SYS_INVALID_DESCRIPTOR
<b>305</b>	305	POSERR_SYS_MEMORY_ALLOCATION
<b>311</b>	311	POSERR_SYS_FUNCTION_NOT_SUPPORTED
<b>312</b>	312	POSERR_SYS_BUFFER_TOO_SMALL
<b>315</b>	315	POSERR_SYS_NOT_ACQUIRED
<b>317</b>	317	POSERR_SYS_DEVICE_OFFLINE
<b>329</b>	329	POSERR_SYS_TIMEOUT
<b>332</b>	332	POSERR_SYS_INTERRUPTED
<b>334</b>	334	POSERR_SYS_INVALID_ADDRESS
<b>335</b>	335	POSERR_SYS_LOCKED_NO_DATA_READ
<b>337</b>	337	POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>4102</b>	4102	POSERR_NVRAM_INVALID_CURSOR
<b>4103</b>	4103	POSERR_NVRAM_EOF
<b>4105</b>	4105	POSERR_NVRAM_INVALID_LENGTH_RECORD
<b>4106</b>	4106	POSERR_NVRAM_INVALID_DATA_CRC

## Related Information

“PosOpen()” on page 18-10.

## Examples

This example reads from the IBM Point of Sale Subsystem input queue (device descriptor zero), looking for specific event messages. If scanner or MSR data is available, it reads from the scanner or from the MSR.

```
#include <pos/pos.h>

int    msrdes;                /* msr device descriptor    */
int    scandes;              /* scanner device descriptor */
char   Buffer[256];          /* data                      */
char   QMsg[EVENT_MESSAGE_SIZE]; /* message                  */
POSQMSG *pMsg;              /* message pointer          */

/*
 * code to initialize, open, acquire and set up the devices
 */

rc = PosRead( 0, QMsg, sizeof( QMsg )); /* get next event          */
if ( rc > 0 )
{
    pMsg = (POSQMSG *) QMsg;
    switch ( pMsg->msg )
    {
        case POSM_MSR_DATA_AVAIL:
        {
            rc = PosRead( msrdes, Buffer, sizeof( Buffer ));
            break;
        } /* end case(POSM_MSR_DATA_AVAIL) */

        case POSM_SCAN_DATA_AVAIL:
        {
            rc = PosRead( scandes, Buffer, sizeof( Buffer ));
            break;
        } /* end case(POSM_SCAN_DATA_AVAIL) */

        case POSM_KBD_WM_CHAR:
        {
            /*
             * process SIO keyboard scan code
            */
        }
    }
}
```

## PosRead()

created on October 2, 2001

```
        */
        break;
    }

    case POSM_SYS_DEVICE_ONLINE:
    {
        /*
         * determine if this is my device
         */
        OpenDevice(pMsg);          /* see PosOpen example */
        break;

    } /* end case(POSM_SYS_DEVICE_ONLINE) */

    default:
    {
        break;
    }

} /* end switch(pMsg->msg) */

} /* end (PosRead(0...) rc > 0) */
```

## PosWrite()

### Purpose

Writes to a device.

### Syntax

```
#include <pos/pos.h>
```

```
ssize_t PosWrite( int devdes, const void *buf, size_t nbyte ) ;
```

### Parameters

*devdes*

Specifies a valid device descriptor. This descriptor identifies a connection to a device. This descriptor must have been previously created using the *PosOpen()* subroutine.

*buf* A pointer to the data to be written.

*nbyte* The number of bytes to be written.

### Description

The *PosWrite()* subroutine attempts to process *nbyte* bytes from the buffer pointed to by *buf* for the device associated with the device descriptor (*devdes*). This subroutine queues a write operation to the device to be completed later. If an error occurs when the write is actually performed, your application is notified by way of an event message.

If *nbyte* is 0 (zero), the *PosWrite()* subroutine returns 0 (zero) and has no other results.

This subroutine can be used only after the process has been initialized and a target device has been opened and acquired.

### Return Values

Upon successful completion, *PosWrite()* returns an integer indicating the number of bytes processed from the buffer. Otherwise, it returns a value of -1 and *errno* is set to indicate the error.

### Error Codes

If the *PosWrite()* subroutine fails, *errno* is set to one of the following:

<b>302</b>	302 POSERR_SYS_NOT_INITIALIZED
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>311</b>	311 POSERR_SYS_FUNCTION_NOT_SUPPORTED
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>318</b>	318 POSERR_SYS_INVALID_LENGTH
<b>334</b>	334 POSERR_SYS_INVALID_ADDRESS
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>4914</b>	4914 POSERR_PRN_INCORRECT_DATA_FORMAT
<b>4904</b>	4904 POSERR_PRN_INVALID_STATION
<b>4915</b>	4915 POSERR_PRN_LOGO_EXISTS
<b>5907</b>	5907 POSERR_RS232_PREV_NOT_COMPLETE

## Related Information

“PosOpen()” on page 18-10.

## Examples

This example writes “Hello World” at position 0 on the display.

```
#include <pos/pos.h>
#include <pos/display.h>

int  dspdes;                /* display device descriptor */
PosArg myarg;              /* resource name & value    */

rc = PosInitialize( "myappl", "checkout", 0, 0, 0, 0 );

dspdes = PosOpen( "mydisplay", PosDisplay, 0,0 );

if ( -1 != dspdes )
{
    rc = PosIOctl( dspdes, POS_SYS_ACQUIRE_DEVICE, 0, 0 );
}

if ( 0 == rc )
{
    PosSetArg( myarg[0], PosNdisplayCursor, 0 );
    rc = PosIOctl( dspdes, POS_SYS_SET_VALUES, myarg, 1 );
    rc = PosWrite( dspdes, "Hello World", 11 );
}
```

created on October 2, 2001

---

## **Part 3. Programming Reference**

created on October 2, 2001

## Chapter 19. PosIOctl() Requests

POS_ALARM_SILENCE_ALARM . . . . .	19-3
POS_ALARM_SOUND_ALARM . . . . .	19-4
POS_DSP_CLEAR_SCREEN . . . . .	19-5
POS_DSP_DEFINE_CHARACTERS . . . . .	19-6
POS_KBD_DISABLE_HOT_KEYS . . . . .	19-7
POS_KBD_DISABLE_NUM_LOCK . . . . .	19-8
POS_KBD_DISABLE_SCROLL_LOCK . . . . .	19-9
POS_KBD_ENABLE_HOT_KEYS . . . . .	19-10
POS_KBD_ENABLE_NUM_LOCK . . . . .	19-11
POS_KBD_ENABLE_SCROLL_LOCK . . . . .	19-12
POS_KBD_SET_NUM_LOCK_OFF . . . . .	19-13
POS_KBD_SET_NUM_LOCK_ON . . . . .	19-14
POS_KBD_SET_SCROLL_LOCK_OFF . . . . .	19-15
POS_KBD_SET_SCROLL_LOCK_ON . . . . .	19-16
POS_KBD_SET_TYPEMATIC_OFF . . . . .	19-17
POS_KBD_SET_TYPEMATIC_ON . . . . .	19-18
POS_KBD_SILENCE_TONE . . . . .	19-19
POS_KBD_SOUND_TONE . . . . .	19-20
POS_POWER_OFF . . . . .	19-21
POS_POWER_ON . . . . .	19-22
POS_POWER_SET_ALARM . . . . .	19-23
POS_PRN_DEFINE_CHARACTERS . . . . .	19-24
POS_PRN_DISABLE_DI_PRINTING . . . . .	19-25
POS_PRN_DISABLE_FISCAL_PRINTING . . . . .	19-26
POS_PRN_DISCARD_DATA . . . . .	19-27
POS_PRN_ENABLE_DI_PRINTING . . . . .	19-28
POS_PRN_ENABLE_FISCAL_PRINTING . . . . .	19-29
POS_PRN_RESET_PRINTER . . . . .	19-30
POS_PRN_HOLD_PRINTING . . . . .	19-31
POS_PRN_RELEASE_PRINTING . . . . .	19-32
POS_PRN_RESUME_PRINTING . . . . .	19-33
POS_PRN_RETRY_PRINTING . . . . .	19-34
POS_PRN_SILENCE_TONE . . . . .	19-35
POS_PRN_SOUND_TONE . . . . .	19-36
POS_RS232_SEND_BREAK . . . . .	19-37
POS_SCALE_CLEAR_SCREEN . . . . .	19-38
POS_SCALE_ZERO_SCALE . . . . .	19-39
POS_SCAN_DISCARD_DATA . . . . .	19-40
POS_SYS_ACQUIRE_DEVICE . . . . .	19-41
POS_SYS_GET_VALUES . . . . .	19-42
POS_SYS_LOCK_DEVICE . . . . .	19-43
POS_SYS_RELEASE_DEVICE . . . . .	19-44
POS_SYS_SET_VALUES . . . . .	19-45
POS_SYS_UNLOCK_DEVICE . . . . .	19-47
POS_TILL_OPEN_TILL . . . . .	19-48
POS_TOUCH_SILENCE_TONE . . . . .	19-49
POS_TOUCH_SOUND_TONE . . . . .	19-50

This chapter contains the valid requests for the *PosIOctl()* subroutine. Most of the requests require exclusive control of the device, so your application process must acquire the device before issuing these *PosIOctl()* requests. The exceptions to this rule are:

### **POS\_SYS\_ACQUIRE\_DEVICE**

Can only be used before the device is acquired.

### **POS\_SYS\_GET\_VALUES**

Can be used either before or after the device is acquired.

### **POS\_SYS\_SET\_VALUES**

Can be used either before or after the device is acquired.

The POS\_SYS\_SET\_VALUES request does have some device-dependent restrictions. See the individual device chapters in Part 2. Programming Guide and the individual device resources in Chapter 21. Resource Sets for these restrictions.

**Note:** The POS\_SYS\_SET\_VALUES request allows you to change resource values via the *args* and *nargs* parameters and the POS\_KBD\_SOUND\_TONE and the POS\_PRN\_SOUND\_TONE request allows you to temporarily override resource values via the *args* and *nargs* parameters.

---

## POS\_ALARM\_SILENCE\_ALARM

### Description

This request turns the alarm off.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is not supported on any model of the IBM 4695.
2. This request is not supported on IBM Point of Sale Subsystem for Linux.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_ALARM\_SOUND\_ALARM

### Description

This request turns the alarm on.

This request can be issued after opening and acquiring the device.

**Notes:**

1. This request is not supported on any model of the IBM 4695.
2. This request is not supported on IBM Point of Sale Subsystem for Linux.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_DSP\_CLEAR\_SCREEN

### Description

This request clears the display. The indicator lights are not changed.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_DSP\_DEFINE\_CHARACTERS

### Description

This request defines characters with the values that are listed in the *PosArg* and that are passed as the *args* parameter on the *PosIOctl()* subroutine.

**Important:** When this request is issued, all previous user-defined characters (UDC) will be discarded. Only the characters defined with this request will be available for your application to use after this request successfully completes.

**Character/Graphics Display:** To define a character for the Character/Graphics Display, the application using the display must be using a DBCS code page, the two-byte character code must be pointed to by the *name* member of the *PosArg* structure and the character definition data must be pointed to by the *value* member. The allowed double-byte character code range is 0x8140 through 0xFC4B for Japan and 0xA1A1 through 0xFDFE for Korea. The character definition data must be 32 bytes in length, and a maximum of 96 characters can be defined.

For the USB Character/Graphics Display, the allowed double-byte character code range is 0x8100 through 0xFFFF. The character definition data must be 32 bytes in length, and a maximum of 64 characters can be defined. Single-byte character code can be defined when the application is using a SBCS code page. The allowed single-byte character code range is 0x00 through 0xFF. The character definition data must be 16 bytes in length, and a maximum of 16 characters can be defined.

**Vacuum Fluorescent and Operator Displays:** To define a character for a VFD II or Operator Display, the single-byte character code must be pointed to by the *name* member of the *PosArg* structure and the character definition data must be pointed to by the *value* member. The maximum number of characters you can define is 8.

**Note:** You must provide 74 bytes of character definition data.

This request can be issued after opening and acquiring the device. This request is supported only on the following displays:

- Character/Graphics Display.
- 40-Character Vacuum Fluorescent Display II
- Two-sided Vacuum Fluorescent Display II
- Operator Display

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>334</b>	334 POSERR_SYS_INVALID_ADDRESS
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_DISABLE\_HOT\_KEYS

### Description

This request disables the use of the operating system hot key sequences. The user is prevented from using the system hot keys to switch to other applications.

This request causes the last key of the following key sequences to be discarded by the IBM Point of Sale Subsystem. The hot keys sequences are:

- Ctrl-Esc
- Alt-Esc
- Alt-Tab
- Alt-Shift-Tab
- Alt-Home

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is supported only for keyboards attached to the system keyboard port.
2. This request is ignored by all keyboards attached to an SIO keyboard port.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_DISABLE\_NUM\_LOCK

### Description

This request disables the use of the **Num Lock** key.

This request can be used to keep the user from changing the Num Lock state when the keyboard is acting as a point-of-sale keyboard. After using this request, your application should issue the POS\_KBD\_SET\_NUM\_LOCK\_ON or POS\_KBD\_SET\_NUM\_LOCK\_OFF *PosIOctl()* request to set the Num Lock state.

This request can be issued after opening and acquiring the device.

This request is supported by the alphanumeric point-of-sale keyboards. All other keyboards ignore this request.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_DISABLE\_SCROLL\_LOCK

### Description

This request disables the use of the **Scroll Lock** key.

This request can be used to keep the user from changing the Scroll Lock state when the keyboard is acting as a point-of-sale keyboard. After using this request, your application should issue the POS\_KBD\_SET\_SCROLL\_LOCK\_ON request or the POS\_KBD\_SET\_SCROLL\_LOCK\_OFF *PosIOctl()* request to set the Scroll Lock state.

This request can be issued after opening and acquiring the device.

This request is supported by the alphanumeric point-of-sale keyboards. All other keyboards ignore this request.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_ENABLE\_HOT\_KEYS

### Description

This request enables the use of the operating system hot key sequences. The user is able to use the system hot keys to switch to other applications.

This request can be issued after opening and acquiring the device.

**Notes:**

1. This request is only supported for keyboards attached to the system keyboard port.
2. This request is ignored by all keyboards attached to an SIO keyboard port.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_ENABLE\_NUM\_LOCK

### Description

This request enables the use of the **Num Lock** key.

This request can be issued after opening and acquiring the device.

This request is supported by the alphanumeric point-of-sale keyboards. All other keyboards ignore this request.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_KBD\_ENABLE\_SCROLL\_LOCK

### Description

This request enables the use of the **Scroll Lock** key.

This request can be issued after opening and acquiring the device.

This request is supported by the alphanumeric point-of-sale keyboards. All other keyboards ignore this request.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_SET\_NUM\_LOCK\_OFF

### Description

This request sets the state of the Num Lock to off.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is supported by the alphanumeric point-of-sale keyboards. All other keyboards ignore this request.
2. On OS/2, for the alphanumeric point-of-sale keyboards attached to the system keyboard port, this request results in a Category 4 Function 53H *DosDevIOctl()* call. For Presentation Manager applications, the 53H *DosDevIOctl()* changes only the state and not the keyboard indicator light. Presentation Manager programs should consider using the *WinSetKeyboardStateTable()* subroutine call to set the keyboard state.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_SET\_NUM\_LOCK\_ON

### Description

This request sets the state of the Num Lock to on.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is supported by the alphanumeric point-of-sale keyboards. All other keyboards ignore this request.
2. On OS/2, for the alphanumeric point-of-sale keyboards attached to the system keyboard port, this request results in a Category 4 Function 53H *DosDevIOctl()* call. For Presentation Manager applications, the 53H *DosDevIOctl()* changes only the state and not the keyboard indicator light. Presentation Manager programs should consider using the *WinSetKeyboardStateTable()* subroutine call to set the keyboard state.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_SET\_SCROLL\_LOCK\_OFF

### Description

This request sets the state of the Scroll Lock to off.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is supported by the alphanumeric point-of-sale keyboards. All other keyboards ignore this request.
2. On OS/2, for the alphanumeric point-of-sale keyboards attached to the system keyboard port, this request results in a Category 4 Function 53H *DosDevIOctl()* call. For Presentation Manager applications, the 53H *DosDevIOctl()* changes only the state and not the keyboard indicator light. Presentation Manager programs should consider using the *WinSetKeyboardStateTable()* subroutine call to set the keyboard state.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_SET\_SCROLL\_LOCK\_ON

### Description

This request sets the state of the Scroll Lock to on.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is supported by the alphanumeric point-of-sale keyboards. All other keyboards ignore this request.
2. On OS/2, for the alphanumeric point-of-sale keyboards attached to the system keyboard port, this request results in a Category 4 Function 53H *DosDevIOctl()* call. For Presentation Manager applications, the 53H *DosDevIOctl()* changes only the state and not the keyboard indicator light. Presentation Manager programs should consider using the *WinSetKeyboardStateTable()* subroutine call to set the keyboard state.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_SET\_TYEMATIC\_OFF

### Description

This request turns off the typematic function of the keyboard.

This request is ignored by the 50-Key Modifiable Layout Keyboard and the 50-Key Modifiable Layout Keyboard/Operator Display. All other keyboards support this request.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_KBD\_SET\_TYPEMATIC\_ON

### Description

This request turns on the typematic function of the keyboard.

This request is ignored by the 50-Key Modifiable Layout Keyboard and the 50-Key Modifiable Layout Keyboard/Operator Display. All other keyboards support this request.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_SILENCE\_TONE

### Description

This request turns off the internal speaker of the keyboard.

For the PLU Keyboard/Display-III and Keyboard-V, the POS\_KBD\_SILENCE\_TONE request can be used to turn off the tone only if the **PosNtoneDuration** resource is set to PosON, which means “on until turned off.”

This request can be issued after opening and acquiring the device.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_KBD\_SOUND\_TONE

### Description

This request turns on the internal speaker of the keyboard.

This request can be issued after opening and acquiring the device.

### Resources Used

The frequency, duration, and volume of the tone are determined by the value of the resources listed in the following list.

- **PosNtoneFreq**
- **PosNtoneDuration**
- **PosNtoneVolume**

These resource values can be specified by your application by using the *args* and *nargs* parameters on this *PosIOctl()* request, on the `POS_SYS_SET_VALUES` *PosIOctl()* request, or on the *PosOpen()* subroutine. If your application does not specify a resource value, the value set in the resource file is used. See Chapter 21. Resource Sets for details of the individual resources.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301	POSERR_SYS_OS_ERROR
<b>303</b>	303	POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315	POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316	POSERR_SYS_INVALID_REQUEST
<b>317</b>	317	POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337	POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>4701</b>	4701	POSERR_KBD_INVALID_FREQUENCY
<b>4702</b>	4702	POSERR_KBD_INVALID_DURATION
<b>4703</b>	4703	POSERR_KBD_INVALID_VOLUME

---

## POS\_POWER\_OFF

### Description

This request can be used to turn off the power supply to any of the following point-of-sale terminals:

- IBM 4693-2x2
- IBM 4693-3x1
- IBM 4693-4x1
- IBM 4693-5x1
- IBM 4693-7x1

In the preceding list, “x” can be a number or alphabetic character.

This request can be issued after opening and acquiring the device.

#### Notes:

1. The programmable power device is supported by the IBM Point of Sale Subsystem for Windows on the Microsoft Windows on 4693 2x2 machines only.
2. This request is not supported on IBM Point of Sale Subsystem for Linux.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_POWER\_ON

### Description

This request can be used to turn on the power supply to the IBM 4693-2x2 point-of-sale terminal, where “x” can be a number or alphabetic character.

This request can be issued after opening and acquiring the device.

**Note:** This request is not supported on IBM Point of Sale Subsystem for Linux.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_POWER\_SET\_ALARM

### Description

This request can be used to set the day of month and time of day that power is to be restored to the following point-of-sale terminals:

- IBM 4693-3x1
- IBM 4693-4x1
- IBM 4693-5x1
- IBM 4693-7x1

In the preceding list, “x” can be a number or alphabetic character.

This request can be issued only after opening and acquiring the device.

**Note:** This request is only supported on OS/2.

### Resources Used

**PosNpowerAlarm** is the only resource used by the POS\_POWER\_SET\_ALARM request.

See Chapter 21. Resource Sets for details of the individual resources.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>6401</b>	6401 POSERR_POWER_INVALID_DAY
<b>6402</b>	6402 POSERR_POWER_INVALID_HOUR
<b>6403</b>	6403 POSERR_POWER_INVALID_MINUTES

---

## POS\_PRN\_DEFINE\_CHARACTERS

### Description

This request defines characters with the values that are listed in the *PosArg* and that are passed as the *args* parameter on the *PosIOctl()* subroutine.

**Important:** For the IBM 4689-00x printers, when this request is issued, all previous user-defined characters (UDC) will be discarded. Only the characters defined with this request will be available for your application to use after this request successfully completes.

To define a character, the application using the printer must be using a DBCS code page, the two-byte character code must be pointed to by the *name* member of the *PosArg* structure and the character data must be pointed to by the *value* member.

For the 4689-001, the allowed double-byte character code range is 0x8140 through 0xFC4B and a maximum of 128 characters can be defined. In 25 CPL mode, the character definition data must be 42 bytes in length. In 30 CPL mode, the character definition data must be 34 bytes in length.

For the 4689-002, the allowed double-byte character code range is 0xA1A1 through 0xFDFE, the character definition data must be 42 bytes in length, and a maximum of 128 characters can be defined.

For the 4689-3x1, the allowed double-byte character code range is 0x8000 through 0x9FFF and 0xE000 through 0xFFFF, the character definition data must be 72 bytes in length and preceded by the two-byte character code, and a maximum of 64 characters can be defined. Only normal font character (32 CPL mode) can be defined.

For the new 4689-TD5, the allowed double-byte character code range is 0x8100 through 0xFFFE.

This request can be issued after opening and acquiring the device.

**Note:** This request is supported only on the IBM 4689-001, the IBM 4689-002, and the IBM 4689-3x1 and 4689-TD5 printers.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>334</b>	334 POSERR_SYS_INVALID_ADDRESS
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>4916</b>	4916 POSERR_PRN_UDC_CHARACTER_EXISTS

---

## POS\_PRN\_DISABLE\_DI\_PRINTING

### Description

This request is required in order to print data at the CR or SJ station.

If the DI station is disabled, the printer device handler prints only at the CR or SJ stations. Any data sent to the DI station is queued. When the DI station is disabled, any queued CR or SJ data is printed.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is not supported on the IBM 4689 Model 3x1 and TD5 printers.
2. On the USB version of the IBM 4610 SureMark Point of Sale Fiscal printers, this request is ignored. Fiscal printing is always enabled.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_PRN\_DISABLE\_FISCAL\_PRINTING

### Description

This request is required in order to print non-fiscal data. If fiscal printing is disabled, all fiscal commands sent to the printer device handler are queued and any queued non-fiscal data is printed.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is available only on the fiscal printer.
2. On the USB versions of the IBM 4610 SureMark Point of Sale Fiscal printers, this request is ignored. Fiscal printing is always enabled.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_PRN\_DISCARD\_DATA

### Description

This request is required in order to discard any queued data after an unrecoverable printer error or after a fiscal command error. The queued data can also be discarded at any time.

This request can be issued after opening and acquiring the device.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_PRN\_ENABLE\_DI\_PRINTING

### Description

This request is required in order to print data on the inserted document.

If the DI station is enabled, the printer device handler prints only at the DI station. Any data sent to the CR or SJ stations is queued. When the DI station is enabled, any queued DI data is printed.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is not supported on the IBM 4689 Model 3x1 and TD5 printers.
2. On the USB versions of the IBM 4610 SureMark Point of Sale Fiscal printers, this request is ignored. Fiscal printing is always enabled.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_PRN\_ENABLE\_FISCAL\_PRINTING

### Description

This request is required in order to print fiscal commands.

If fiscal printing is enabled, the printer device handler prints only fiscal commands. Any non-fiscal data is queued. When fiscal printing is enabled, any queued fiscal commands are printed.

This request can be issued after opening and acquiring the device.

#### Notes:

1. This request is available only on the fiscal printer.
2. For the USB versions of the IBM 4610 SureMark Point of Sale Fiscal printers, fiscal printing is always enabled.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_PRN\_RESET\_PRINTER

### Description

This request issues a power-on reset command to the printer.

This request can be issued after opening and acquiring the device.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_PRN\_HOLD\_PRINTING

### Description

This request tells the printer to hold the print buffer, so that all commands can be sent to printer as one command string.

This request can be issued after opening and acquiring the device.

**Note:** This request is not supported on the USB versions of the IBM 4610 SureMark Point of Sale Fiscal printers.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_PRN\_RELEASE\_PRINTING

### Description

This request tells the printer to execute the commands that were previously buffered with POS\_PRN\_HOLD\_PRINTING.

This request can be issued after opening and acquiring the device.

**Note:** This request is not supported on the USB versions of the IBM 4610 SureMark Point of Sale Fiscal printers.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_PRN\_RESUME\_PRINTING

### Description

This request is required in order to resume printing, after a printer error occurs, without retrying the error lines.

The printer device handler stops printing if an error occurs, and sends the event message `POSM_PRN_PRINTER_ERROR`. See page “`POSM_PRN_PRINTER_ERROR`” on page 20-11 for a definition of this message.

Typically, the application instructs the user to correct the error. The user signals the application with a key stroke when the printer is ready again. The application must issue a resume printing request to start printing again. The resume printing request causes the printer device handler to skip the line or lines that failed. Data might be lost as a result.

This request can be issued after opening and acquiring the device.

### Resources Used

`PosNresumeString` is the only resource used by the `POS_PRN_RESUME_PRINTING` request.

See Chapter 21. Resource Sets for details of the individual resources.

### Error Codes

If the `PosIOctl()` subroutine fails, `errno` is set to one of the following:

<b>301</b>	301	POSERR_SYS_OS_ERROR
<b>303</b>	303	POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315	POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316	POSERR_SYS_INVALID_REQUEST
<b>317</b>	317	POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337	POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_PRN\_RETRY\_PRINTING

### Description

This request is required in order to retry printing after a printer error.

The printer device handler stops printing if an error occurs, and sends the event message `POSM_PRN_PRINTER_ERROR`. See page “`POSM_PRN_PRINTER_ERROR`” on page 20-11 for a definition of this event message.

Typically, the application instructs the user to correct the error. The user signals the application with a key stroke when the printer is ready again. The application must issue a retry printing request to start printing again.

The `POS_PRN_RETRY_PRINTING` request causes the printer device handler to reprint the line or lines that failed. The overlay string will only be printed if an error occurred during a command that prints data, and the printer device handler has sent the command to the printer. No data is lost. There is a chance of duplicate lines in the SJ station, but they are overlaid with the overlay string supplied by the application. In the customer receipt station, the error line is followed immediately by the correct line. The reprinted line is overlaid with the overlay string.

**Note:** On the IBM 4610 printers, only the first character from the overlay string is used on a retry. This character will be printed only if the retry is for a line that has a home error. In all other instances the line will be printed without the overlay character in the first position.

This request can be issued after opening and acquiring the device.

### Resources Used

**PosNretryString** is the only resource used by the `POS_PRN_RETRY_PRINTING` request.

See Chapter 21. Resource Sets for details of the individual resources.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301	POSERR_SYS_OS_ERROR
<b>303</b>	303	POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315	POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316	POSERR_SYS_INVALID_REQUEST
<b>317</b>	317	POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337	POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_PRN\_SILENCE\_TONE

### Description

This request turns off the internal speaker of the printer.

This request can be issued after opening and acquiring the device.

**Note:** This request is available only on the Single Station SureMark printers (TF6, TM6, TF7, TM7).

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_PRN\_SOUND\_TONE

### Description

This request turns on the internal speaker of the printer.

This request can be issued after opening and acquiring the device.

**Note:** This request is available only on the Single Station SureMark printers (TF6, TM6, TF7, TM7).

### Resources Used

The frequency, duration, and volume of the tone are determined by the value of the following resources:

- **PosNprintToneFrequency**
- **PosNprintToneDuration**
- **PosNprintToneVolume**
- **PosNprintToneOctave**
- **PosNprintToneNote**

These resource values can be specified by your application by using the *args* and *nargs* parameters on this *PosIOctl()* request, on the *POS\_SYS\_SET\_VALUES PosIOctl()* request, on the *PosOpen()* call, or in the resource file for the applications. . See Chapter 21. Resource Sets for details of the individual resources.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>4922</b>	4922 POSERR_PRN_INVALID_TONE_VOLUME
<b>4923</b>	4923 POSERR_PRN_INVALID_TONE_DURATION
<b>4924</b>	4924 POSERR_PRN_INVALID_TONE_NOTE
<b>4925</b>	4925 POSERR_PRN_INVALID_TONE_OCTAVE
<b>4926</b>	4926 POSERR_PRN_INVALID_TONE_FREQUENCY

---

## POS\_RS232\_SEND\_BREAK

### Description

This request sends a break signal to the receiving RS-232C port.

The RS-232C hardware on IBM point-of-sale hardware only returns an incoming break signal to the application if it is followed by at least 1 byte of data. Two messages are then received on the input queue. One indicates that data is available, and the other indicates that a break has been detected. The break signal sent by this *PosIOctl()* request is sent immediately, regardless of whether it is followed by any data.

This request can be issued after opening and acquiring the device.

**Note:** This request is not supported on IBM Point of Sale Subsystem for Linux.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_SCALE\_CLEAR\_SCREEN

### Description

This request causes the scale display to be cleared of any displayed numbers. When the displayed weight is zero, this request has no effect. When the displayed weight is not zero, each displayed number is replaced by a blank character but the decimal point (.) and the units designator (lb or kg) remain. This request is only valid when the scale resource **PosNoperMode** is set to PosUK.

This request can be issued after opening and acquiring the device.

This request is supported only by the following scales:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4698 Point of Sale Scanner Model 2
- IBM USB Scale Interface

**Note:** The scale device is not supported by the IBM Point of Sale Subsystem for Windows on the Microsoft Windows 3.1 operating system.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>329</b>	329 POSERR_SYS_TIMEOUT
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>6208</b>	6208 POSERR_SCALE_INVALID_CLEAR_SCREEN_REQUEST

---

## POS\_SCALE\_ZERO\_SCALE

### Description

This request causes the scale to zero itself. The scale will attempt to correlate the current weight reading with a “no-load” scale condition. If an item weighing less than approximately 0.6 pounds (or less than approximately 0.295 kilograms) is on the weighing surface when this request is issued, the request will complete successfully, causing the scale to register a negative weight when the weighing surface is empty.

This request can be issued after opening and acquiring the device.

This request is supported only by the following scales:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4698 Point of Sale Scanner Model 2
- IBM USB Scale Interface

**Note:** The scale device is not supported by the IBM Point of Sale Subsystem for Windows on the Microsoft Windows 3.1 operating system.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>6207</b>	6207 POSERR_SCALE_ZERO_SCALE_FAILED

---

## POS\_SCAN\_DISCARD\_DATA

### Description

This request discards any queued bar codes that have not been read by the application. Only those bar codes that belong to the application that issues this request are discarded. This request should be used with caution, because the scanner operator might assume that the scanned items (represented by the bar codes in the queue) have been processed by the application.

Bar codes that are received from the scanner device belong to the application that has the scanner device acquired. Some scanners successfully read bar codes even when the scanner device is not acquired by any application. Any bar codes received from the scanner while the device is not acquired are identified as unexpected data. The POS\_SCAN\_UNEXPECTED\_DATA flag in the *Flags* field of the scanner data buffer is set. These bar codes belong to the next application to acquire the scanner device.

As a result, bar codes might already be queued for an application at the time the scanner device is acquired. Prior to unlocking the scanner device, the application can either read each queued bar code, checking for POS\_SCAN\_UNEXPECTED\_DATA in the *Flags* field, or it can issue this request to discard all of its queued bar codes.

**Note:** POS\_SCAN\_UNEXPECTED\_DATA is also used to identify bar codes received while the scanner device is acquired but locked. Any bar codes queued after the application acquires the scanner device but before the application unlocks the scanner device will also be marked as unexpected data.

See “Reading Scanner Data” on page 16-4 and “Processing Unexpected Scanner Data” on page 16-9 for more information.

This request can be issued after opening and acquiring the device.

If this request is issued when there are no bar codes queued, the request has no effect and no error is returned.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_SYS\_ACQUIRE\_DEVICE

### Description

This request gives the specified device descriptor exclusive use of the device associated with the device descriptor. The *devdes* parameter of the *PosIOctl()* subroutine specifies the device descriptor.

The device descriptor can be used by any thread in the process that opened the device to issue any IBM Point of Sale Subsystem subroutine call. The success of the subroutine call depends on how well the separate threads sequence the IBM Point of Sale Subsystem subroutine calls.

For example, assume a process has two threads, *t1* and *t2*:

1. Thread *t1* issues the `POS_SYS_ACQUIRE_DEVICE PosIOctl()` for a cash drawer, using the appropriate device descriptor.
2. Thread *t2* successfully issues `POS_TILL_OPEN_TILL PosIOctl()`, using the same device descriptor.
3. Thread *t1* issues the `POS_SYS_RELEASE_DEVICE PosIOctl()` for the same device descriptor.
4. Thread *t2* attempts to issue `POS_TILL_OPEN_TILL PosIOctl()` with the same device descriptor. This request fails because *t1* has released the device descriptor for the cash drawer.

If any thread in the process has already acquired a device by a specific device descriptor, any subsequent `POS_SYS_ACQUIRE_DEVICE` request using the same device descriptor fails with an error code of `POSERR_SYS_ALREADY_ACQUIRED`.

If the device is already acquired by a different device descriptor than the requested device descriptor, the request fails with an error code of `POSERR_SYS_ACQUIRED_BY_OTHER`.

When the device is released by the application that currently has it acquired, the `POSM_SYS_DEVICE_RELEASED` event message is sent to the message queue that was specified on the *PosOpen()* subroutine call for that device.

The only *PosIOctl()* requests that are valid prior to a device being acquired are `POS_SYS_SET_VALUES` and `POS_SYS_GET_VALUES`.

This request can be issued after opening the device.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>313</b>	313 POSERR_SYS_ACQUIRED_BY_OTHER
<b>314</b>	314 POSERR_SYS_ALREADY_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_SYS\_GET\_VALUES

### Description

This request retrieves the values for the resources that are listed in the PosArgPtr structure and that are passed as the *args* parameter on the *PosIOctl()* subroutine.

This request can be issued after opening the device.

### Resources Used

This request accepts all resources that have an access code of "G" (gettable), described in Chapter 21. Resource Sets.

See Chapter 21. Resource Sets for details of the individual resources.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_SYS\_LOCK\_DEVICE

### Description

This request inhibits input from the device specified by the device descriptor.

This request can be issued after opening and acquiring the device. If this request is issued when the device is already locked, the request has no effect and no error is returned.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301	POSERR_SYS_OS_ERROR
<b>303</b>	303	POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315	POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316	POSERR_SYS_INVALID_REQUEST
<b>317</b>	317	POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337	POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_SYS\_RELEASE\_DEVICE

### Description

This request causes the specified device descriptor to relinquish exclusive control of the device.

Control of the device is released after the current unit of work for the device is complete. The *devdes* parameter of the *PosIOctl()* subroutine specifies the device descriptor.

This request can be issued after opening and acquiring the device.

If the device descriptor has not acquired its associated device, the *PosIOctl()* subroutine fails with an error code of `POSERR_SYS_NOT_ACQUIRED`.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301	POSERR_SYS_OS_ERROR
<b>303</b>	303	POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315	POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316	POSERR_SYS_INVALID_REQUEST
<b>337</b>	337	POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_SYS\_SET\_VALUES

### Description

This request sets the resources with the values that are listed in the PosArgPtr structure and that are passed as the *args* parameter on the *PosIOctl()* subroutine.

**Important:** Resource names that are not recognized by the device handler are ignored. This allows a single argument list for different types of devices, because each device handler ignores what it does not recognize. If any of the recognized resource name values are not valid, no resource is changed.

This request can be issued after opening the device.

### Resources Used

This request accepts all resources that have an access code of "S" (setable), described in Chapter 21. Resource Sets.

See Chapter 21. Resource Sets for details of the individual resources.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>4401</b>	4401 POSERR_DSP_INVALID_CURSOR
<b>4402</b>	4402 POSERR_DSP_INVALID_MODE
<b>4403</b>	4403 POSERR_DSP_INVALID_SIZE
<b>4404</b>	4404 POSERR_DSP_UNSUPPORTED_BITMAP
<b>4405</b>	4405 POSERR_DSP_BAD_BITMAP
<b>4102</b>	4102 POSERR_NVRAM_INVALID_CURSOR
<b>4701</b>	4701 POSERR_KBD_INVALID_FREQUENCY
<b>4702</b>	4702 POSERR_KBD_INVALID_DURATION
<b>4703</b>	4703 POSERR_KBD_INVALID_VOLUME
<b>4708</b>	4708 POSERR_KBD_INVALID_KEYBOARD_CLICK
<b>4901</b>	4901 POSERR_PRN_INVALID_DI_WIDTH
<b>4902</b>	4902 POSERR_PRN_INVALID_INTERLEAVED_VALUE
<b>4903</b>	4903 POSERR_PRN_INVALID_HEAD_PARKED_POSITION
<b>4904</b>	4904 POSERR_PRN_INVALID_STATION
<b>4905</b>	4905 POSERR_PRN_INVALID_MODE
<b>4906</b>	4906 POSERR_PRN_INVALID_CR_LF_DISTANCE
<b>4907</b>	4907 POSERR_PRN_INVALID_SJ_LF_DISTANCE
<b>4908</b>	4908 POSERR_PRN_INVALID_DI_LF_DISTANCE
<b>4909</b>	4909 POSERR_PRN_INVALID_FEED_DIRECTION
<b>4910</b>	4910 POSERR_PRN_INVALID_FISCAL_NOTIFY
<b>4911</b>	4911 POSERR_PRN_INVALID_DI_ORIENTATION
<b>4912</b>	4912 POSERR_PRN_INVALID_LEFT_MARGIN_CR
<b>4913</b>	4913 POSERR_PRN_INVALID_PRINT_ALIGNMENT
<b>4927</b>	4927 POSERR_PRN_INVALID_CODE_PAGE
<b>5401</b>	5401 POSERR_CDR_INVALID_PULSE_WIDTH
<b>5706</b>	5706 POSERR_SCAN_INVALID_BEEP_STATE
<b>5735</b>	5735 POSERR_SCAN_INVALID_QUEUE_ALL_INDICATOR

## POS\_SYS\_SET\_VALUES

created on October 2, 2001

<b>6701</b>	6701 POSERR_TOUCH_INVALID_BACKLIGHT_ON
<b>6702</b>	6702 POSERR_TOUCH_INVALID_CLICK_VOLUME
<b>6703</b>	6703 POSERR_TOUCH_INVALID_CONTRAST
<b>6704</b>	6704 POSERR_TOUCH_INVALID_ENTRY_CLICK
<b>6705</b>	6705 POSERR_TOUCH_INVALID_EXIT_CLICK
<b>6706</b>	6706 POSERR_TOUCH_INVALID_MODE
<b>6707</b>	6707 POSERR_TOUCH_INVALID_SCREEN_SAVER_TIME
<b>6708</b>	6708 POSERR_TOUCH_INVALID_TONE_DURATION
<b>6709</b>	6709 POSERR_TOUCH_INVALID_TONE_FREQUENCY
<b>6710</b>	6710 POSERR_TOUCH_INVALID_TONE_VOLUME
<b>6711</b>	6711 POSERR_TOUCH_INVALID_BRIGHTNESS

## POS\_SYS\_UNLOCK\_DEVICE

### Description

This request allows input from the device specified by the device descriptor.

This request can be issued after opening and acquiring the device. If this request is issued when the device is already unlocked, the request has no effect and no error is returned.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301	POSERR_SYS_OS_ERROR
<b>303</b>	303	POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315	POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316	POSERR_SYS_INVALID_REQUEST
<b>317</b>	317	POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337	POSERR_SYS_SERVICE_NOT_AVAILABLE

## POS\_TILL\_OPEN\_TILL

### Description

This request opens the cash drawer.

This request can be issued after opening and acquiring the device. The cash drawer must also be online (connected).

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_TOUCH\_SILENCE\_TONE

### Description

This request turns off the internal speaker of the touch screen.

The POS\_TOUCH\_SILENCE\_TONE request can be used to turn off the tone only if the **PosNtouchToneDuration** resource is set to PosON, which means “on until turned off.” If the **PosNtouchToneDuration** is set to anything else, this request is ignored.

This request can be issued after opening and acquiring the device.

**Note:** This request is not supported on IBM Point of Sale Subsystem for Linux.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301 POSERR_SYS_OS_ERROR
<b>303</b>	303 POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315 POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316 POSERR_SYS_INVALID_REQUEST
<b>317</b>	317 POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337 POSERR_SYS_SERVICE_NOT_AVAILABLE

---

## POS\_TOUCH\_SOUND\_TONE

### Description

This request turns on the internal speaker of the touch screen.

This request can be issued after opening and acquiring the device.

**Note:** This request is not supported on IBM Point of Sale Subsystem for Linux.

### Resources Used

The frequency, duration, and volume of the tone are determined by the values of the following resources:

- **PosNtouchToneDuration**
- **PosNtouchToneFreq**
- **PosNtouchToneVolume**

See Chapter 21. Resource Sets for details of the individual resources.

### Error Codes

If the *PosIOctl()* subroutine fails, *errno* is set to one of the following:

<b>301</b>	301	POSERR_SYS_OS_ERROR
<b>303</b>	303	POSERR_SYS_INVALID_DESCRIPTOR
<b>315</b>	315	POSERR_SYS_NOT_ACQUIRED
<b>316</b>	316	POSERR_SYS_INVALID_REQUEST
<b>317</b>	317	POSERR_SYS_DEVICE_OFFLINE
<b>337</b>	337	POSERR_SYS_SERVICE_NOT_AVAILABLE
<b>6708</b>	6708	POSERR_TOUCH_INVALID_TONE_DURATION
<b>6709</b>	6709	POSERR_TOUCH_INVALID_TONE_FREQUENCY
<b>6710</b>	6710	POSERR_TOUCH_INVALID_TONE_VOLUME
<b>6711</b>	6711	POSERR_TOUCH_INVALID_BRIGHTNESS

## Chapter 20. Event Messages

POSM_KBD_STATUS_CHANGE . . . . .	20-3
POSM_KBD_WM_CHAR . . . . .	20-4
POSM_MSR_DATA_AVAIL . . . . .	20-6
POSM_PRN_CHASE_COMPLETE . . . . .	20-7
POSM_PRN_DATA_AVAIL . . . . .	20-8
POSM_PRN_FISCAL_ERROR . . . . .	20-9
POSM_PRN_FISCAL_STATUS . . . . .	20-10
POSM_PRN_PRINTER_ERROR . . . . .	20-11
POSM_PRN_STATUS_CHANGE . . . . .	20-13
POSM_RS232_BREAK_DETECTED . . . . .	20-15
POSM_RS232_DATA_AVAIL . . . . .	20-16
POSM_RS232_XMIT_ABORT . . . . .	20-17
POSM_RS232_XMIT_COMPLETE . . . . .	20-18
POSM_SCAN_DATA_AVAIL . . . . .	20-19
POSM_SYS_DEVICE_OFFLINE . . . . .	20-20
POSM_SYS_DEVICE_ONLINE . . . . .	20-22
POSM_SYS_DEVICE_RELEASED . . . . .	20-24
POSM_TILL_CLOSED . . . . .	20-25
POSM_TILL_OPENED . . . . .	20-26
POSM_TOUCH_DATA . . . . .	20-27
WM_CHAR . . . . .	20-28

This section lists the event messages received by an application whenever certain types of asynchronous events occur. These event messages are sent to the application either by a specific device handler or by the system. The only time an application receives an event message from a device is when it has that device acquired. The format of the event message is defined by the structure `POSQMSG` in the header file `pos.h`.

These event messages arrive on the input queue (for example, the Presentation Manager input queue for OS/2, the window queue for the Microsoft Windows operating systems, or the IBM Point of Sale Subsystem input queue) of the application that has the device acquired. See “Getting Input Messages” on page 5-2 for more information on the IBM Point of Sale Subsystem input queue.

For each event message, the format of the fields in the **mp1** and the **mp2** parameters vary based on the operating system you are using and the type of compiler you compiled your applications with. The IBM Point of Sale Subsystem supports two versions of the **POSQMSG** field:

- |                       |   |
|-----------------------|---|
| <b>32-bit version</b> | All applications use this version when reading from the IBM Point of Sale Subsystem input queue and all applications for the IBM Point of Sale Subsystem for OS/2 use this version. |
| <b>16-bit version</b> | All 16-bit applications for the Microsoft Windows 3.1 operating system and all 32-bit applications for the Microsoft Windows 3.1 operating system using Microsoft Win32s.           |

When reading the format of the **mp1** and **mp2** fields keep the following in mind:

- The fields within **mp1** and **mp2** are listed from top to bottom within a *long* (or *short*) variable.

- To extract individual bytes from either **mp1** or **mp2** use the following:
  - Top byte**  $(\text{mp1} \gg 24) \& 0x000000FF$
  - Middle-top byte**  $(\text{mp1} \gg 16) \& 0x000000FF$
  - Middle-bottom byte**  $(\text{mp1} \gg 8) \& 0x000000FF$
  - Bottom byte**  $\text{mp1} \& 0x000000FF$
- The storage layout in memory will vary amongst different operating systems. Therefore, you cannot cast the *long* variable pointer to a *char* pointer and expect to get the same results.

Below the format of the **POSQMSG** fields, you will find a description of the fields listed in the **mp1** and **mp2** parameters.

---

## POSM\_KBD\_STATUS\_CHANGE

### Description

This event message is generated when the keyboard status changes.

### POSQMSG Fields

#### 32-bit version:

**mp1**  
 status (unsigned short)  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 extra info (1ong)

#### 16-bit version:

**mp1**  
 status (unsigned char)  
 devdes (unsigned char)

**mp2**  
 extra info (1ong)

The details of the fields in **mp1** and **mp2** are:

<b>status</b>	This field indicates the status change that caused this event. The valid value and its meaning are: <b>PosKEY_LOCK (0x01)</b> The manager key lock has changed positions.
<b>devdes</b>	This field is set to the device descriptor of the device that generated the event message.
<b>extra info</b>	This field contains additional information concerning the status change. The value of this field depends on the value of the <b>status</b> field in <b>mp1</b> .  If the PosKEY_LOCK status flag is set in <b>mp1</b> , this field contains the current value of the <b>PosNkeyLock</b> resource.

### Remarks

The application should take whatever action is appropriate for the new status after it receives a POSM\_KBD\_STATUS\_CHANGE event message.

## POSM\_KBD\_WM\_CHAR

### Description

This event message is generated when the non-system keyboard device handler receives a character.

### POSQMSG Fields

**32-bit version:**

<b>mp1</b>	charcode (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	flags (unsigned short) reserved (unsigned char) scancode (unsigned char)

**16-bit version:**

<b>mp1</b>	charcode (unsigned char) devdes (unsigned char)
<b>mp2</b>	flags (unsigned short) reserved (unsigned char) scancode (unsigned char)

The details of the fields in **mp1** and **mp2** are:

- charcode** This field contains the ASCII character code.
- devdes** This field is set to the device descriptor of the device that generated the event message.
- flags** This field contains keyboard control codes. The keyboard control codes and their meanings are:
  - PosKC\_CHAR (0x0001)**  
The character is valid.
  - PosKC\_SCANCODE (0x0004)**  
The scan code is valid.
  - PosKC\_SHIFT (0x0008)**  
The SHIFT state is active when the key was pressed.
  - PosKC\_CTRL (0x0010)**  
The CTRL state is active when the key was pressed.
  - PosKC\_ALT (0x0020)**  
The ALT state is active when the key was pressed.
  - PosKC\_KEYUP (0x0040)**  
The event is a key-up transition. This bit is set when the device driver receives a break scan code from the keyboard. If this bit is not set, the event is a key-down transition.

**PosKC\_FATFINGER (0x8000)**

The FATFINGER state is active when the key was pressed.

**scancode** This unsigned char field contains the hardware scan code for the key pressed. This is the value that identifies the keyboard event.

## Remarks

After the application receives a POSM\_KBD\_WM\_CHAR event message, it should examine the **mp1** and **mp2** parameters to determine what the key is.

**Notes:**

1. The format of the POSM\_KBD\_WM\_CHAR event message is similar to the format of the WM\_CHAR event messages in Windows and OS/2 Presentation Manager.
2. Some point-of-sale keyboards only report make scan codes.
3. The PosKC\_FATFINGER bit is an indication that one key has been pressed too quickly after another key, or that two keys have been pressed at the same time. It can be used by the application to protect against the user accidentally pressing two keys at the same time. Note that on the 50-Key Modifiable Layout Keyboard and the 50-Key Modifiable Layout Keyboard/Operator Display, the bit is set on the second of the two POSM\_KBD\_WM\_CHAR event messages. On all other keyboards, the bit is set on the first POSM\_KBD\_WM\_CHAR event message. This is determined by the hardware microcode in the keyboard.

## POSM\_MSR\_DATA\_AVAIL

### Description

This event message is generated when MSR data is available to be read.

### POSQMSG Fields

**32-bit version:**

**mp1**  
 reserved (unsigned short)  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 reserved (unsigned short)  
 length (unsigned short)

**16-bit version:**

**mp1**  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 reserved (unsigned short)  
 length (unsigned short)

The details of the fields in **mp1** and **mp2** are:

**devdes** This field is set to the device descriptor of the device that generated the event message.

**length** This field contains the size, in bytes, of the buffer required to read the data associated with this event.

### Remarks

The application should use the *PosRead()* subroutine to retrieve the MSR data after it receives a POSM\_MSR\_DATA\_AVAIL event message.

---

## POSM\_PRN\_CHASE\_COMPLETE

### Description

This event message is generated when any of the three print stations detects a chase escape character sequence in the data that is sent to the printer.

### POSQMSG Fields

#### 32-bit version:

<b>mp1</b>	station (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	status (long)

#### 16-bit version:

<b>mp1</b>	station (unsigned char) devdes (unsigned char)
<b>mp2</b>	status (long)

The details of the fields in **mp1** and **mp2** are:

<b>station</b>	This field indicates the print station where the chase sequence was found. The values and their meanings are: <b>PosCHASE_FROM_CR (0x01)</b> The chase was from the CR station. <b>PosCHASE_FROM_SJ (0x02)</b> The chase was from the SJ station. <b>PosCHASE_FROM_DI (0x04)</b> The chase was from the DI station.
<b>reserved</b>	This unsigned char field is reserved for non-device specific data and has a value of 0 (zero).
<b>devdes</b>	This unsigned char field is set to the device descriptor of the device that generated the event message.
<b>status</b>	Indicates the current status of the printer. See "PosNprintStatus" on page 21-46 for more information.

### Remarks

The application is notified that all previous operations for this print station have been completed when it receives a POSM\_PRN\_CHASE\_COMPLETE event message, for example, when data has been printed on the paper.

## POSM\_PRN\_DATA\_AVAIL

### Description

This event message is generated when printer data is available to be read.

### POSQMSG Fields

**32-bit version:**

<b>mp1</b>	from (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (unsigned short) length (unsigned char)

**16-bit version:**

<b>mp1</b>	from (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (unsigned short) length (unsigned short)

The details of the fields in **mp1** and **mp2** are:

- from** This field indicates where the printer information is from. The valid value and its meaning are:
  - PosPRINT\_DATA\_AVAIL\_FISCAL (0x01)**  
Fiscal data is available from the fiscal printer. (Only available on the fiscal printer)
  - PosPRINT\_DATA\_AVAIL\_MICR (0x02)**  
Check information is available from the MICR reader. (Only available on the IBM Model 3R, IBM Model 4R, and the IBM 4610 Model TI2 and TI4 printers)

**devdes** This field is set to the device descriptor of the device that generated the event message.

**length** This field contains the size in bytes of the buffer required to read the data associated with this event.

### Remarks

After the application receives a POSM\_PRN\_DATA\_AVAIL event message, it should use the *PosRead()* subroutine to retrieve the printer data.

## POSM\_PRN\_FISCAL\_ERROR

### Description

This event message is generated when a fiscal error occurs. The reason for the problem is indicated by **mp1** and the printer status is in **mp2**.

**Note:** This event message is available only on the fiscal printer.

### POSQMSG Fields

**32-bit version:**

**mp1**  
 cause (unsigned short)  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 status (long)

**16-bit version:**

**mp1**  
 cause (unsigned char)  
 devdes (unsigned char)

**mp2**  
 status (long)

The details of the fields in **mp1** and **mp2** are:

**cause** This field indicates the cause of the printer error.

Value	Meaning
<b>Fiscal error code</b>	This field indicates the fiscal error code.

**devdes** This field is set to the device descriptor of the device that generated the event message.

**status** Indicates the current status of the printer. See "PosNprintStatus" on page 21-46.

### Remarks

After the application receives a POSM\_PRN\_FISCAL\_ERROR event message, it should use the display and keyboard to interact with the operator to correct the problem, then issue a POS\_PRN\_RETRY\_PRINTING *PosIOctl()* request to retry the fiscal command. If this is not successful at correcting the error, issue a POS\_PRN\_DISCARD\_DATA *PosIOctl()* request to discard all outstanding printer data and then a POS\_PRN\_RESUME\_PRINTING *PosIOctl()* request to resume printing.

## POSM\_PRN\_FISCAL\_STATUS

### Description

This event message is generated when a fiscal command completes successfully and the application has requested to be notified of the completion. This event message is only sent to the application if the **PosNfiscalNotify** resource is set to PosFISCAL\_NOTIFY\_ON.

**Note:** This event message is available only on the fiscal printer.

### POSQMSG Fields

**32-bit version:**

<b>mp1</b>	reserved (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	status (long)

**16-bit version:**

<b>mp1</b>	reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	status (long)

The details of the fields in **mp1** and **mp2** are:

- devdes** This field is set to the device descriptor of the device that generated the event message.
- status** Indicates the fiscal command that completed successfully.

### Remarks

The application is notified that an outstanding fiscal command is complete when it receives a POSM\_PRN\_FISCAL\_STATUS event message.

---

## POSM\_PRN\_PRINTER\_ERROR

### Description

This event message is generated when any of the three print stations detects a problem. The reason for the problem is indicated by **mp1** and the printer status is indicated by **mp2**.

### POSQMSG Fields

#### 32-bit version:

<b>mp1</b>	cause (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	status (long)

#### 16-bit version:

<b>mp1</b>	reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (unsigned short) cause (unsigned short)

The details of the fields in **mp1** and **mp2** are:

<b>cause</b>	This field indicates the cause of the printer error. The values and their meanings are: <b>PosERROR_COVER_OPEN (0x0001)</b> The cover is open. <b>PosERROR_TRANSPORT_ERROR (0x0002)</b> The print head transport is binding. <b>PosERROR_CR_PRINT_HEAD_OPEN (0x0003)</b> The CR station print head is failing. <b>PosERROR_SJ_PAPER_ERROR (0x0004)</b> The SJ station is out of paper or has a paper jam. <b>PosERROR_SJ_PRINT_HEAD_OPEN (0x0005)</b> The SJ station print head is failing. <b>PosERROR_DOCUMENT_INSERTED (0x0008)</b> An attempt was made to print on the SJ or CR station while a document was inserted in the DI station. <b>PosERROR_DOCUMENT_NOT_INSERTED (0x0010)</b> An attempt was made to print on the DI station while a document was not inserted in the DI station. <b>PosERROR_DOCUMENT_NOT_READY (0x0020)</b> An attempt was made to register a document but there is no document inserted in the DI station. <b>PosERROR_DOCUMENT_NOT_EJECTED (0x0040)</b> An attempt was made to eject a document but there is no document inserted in the DI station. <b>PosERROR_MICR_ERROR (0x0080)</b> An error occurred reading the MICR information.
--------------	--

**PosERROR\_EEPROM\_LOAD\_ERROR (0x0200)**

An attempt was made to download a message or logo to the printer but failed to load. The message number or logo number might be defined already. (IBM 4610 printers only)

**PosERROR\_EEPROM\_FULL (0x0400)**

An attempt was made to download a message or logo to the printer when there was insufficient printer memory to store the message or or logo. (IBM 4610 printers only)

**PosERROR\_COMMUNICATION\_ERROR (0x1000)**

A communication error occurred between the printer device handler and the printer.

**PosERROR\_DI\_FEED\_ERROR (0x2000)**

An error occurred during a check flip operation or while reading MICR information. (IBM 4610 printers only)

**PosERROR\_STARTUP\_ERROR (0x4000)**

An error occurred bringing the printer online. On the 4610 printers, new firmware will be needed to fix this error. (IBM 4610 printers and IBM 4689 Model 3x1 and TD5 printers only)

**PosERROR\_UNRECOVERABLE\_ERROR (0x4001)**

An unrecoverable error occurred during printer operation. (IBM 4689 Model 3x1 and TD5 printers only)

**PosERROR\_COMMAND\_REJECTED (0x8000)**

The printer rejected a command. This could occur when printing a barcode with invalid parameters or when printing either a pre-defined message or a pre-defined logo that does not exist. (IBM 4610 printers only)

- devdes** This field is set to the device descriptor of the device that generated the event message.
- status** Indicates the current status of the printer. See "PosNprintStatus" on page 21-46 for more information.

**Remarks**

After the application receives a POSM\_PRN\_PRINTER\_ERROR event message, it should use the display and keyboard to interact with the operator to correct the problem. It should then issue either a POS\_PRN\_RESUME\_PRINTING *PosIOctl()* request or a POS\_PRN\_RETRY\_PRINTING *PosIOctl()* request.

For the IBM 4610 Printers, if the error condition is PosERROR\_EEPROM\_LOAD\_ERROR the application must issue a POS\_PRN\_DISCARD\_DATA *PosIOctl()* request before issuing either a POS\_PRN\_RETRY\_PRINTING or a POS\_PRN\_RESUME\_PRINTING *PosIOctl()* request.

---

## POSM\_PRN\_STATUS\_CHANGE

### Description

This event message is generated when the status of any paper sensors change or when the position of the printer cover changes.

### POSQMSG Fields

#### 32-bit version:

<b>mp1</b>	status1 (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	status2 (long)

#### 16-bit version:

<b>mp1</b>	reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (unsigned short) status1 (unsigned short)

The details of the fields in **mp1** and **mp2** are:

#### **status1**

This field indicates the status change that caused this event message. The values and their meanings are:

##### **PosCHANGE\_DOCUMENT\_AT\_TOP (0x0001)**

Document present at top sensor has changed. (This message is not returned on the IBM Model 2 printer.)

##### **PosCHANGE\_DOCUMENT\_AT\_FRONT (0x0002)**

Document present at front sensor has changed. (For the Model 2 printer, this message will be returned when a document is inserted.)

##### **PosCHANGE\_DOCUMENT\_REGISTER (0x0004)**

Document registered has changed.

##### **PosCHANGE\_COVER (0x0008)**

The printer cover position has changed. On printers with more than one cover, this status applies to any of the covers.

##### **PosCHANGE\_CR\_PAPER\_LOW (0x0010)**

The paper in the CR station is low. (This message is available only on the IBM 4689-00x printers.)

##### **PosCHANGE\_CR\_PAPER\_OUT (0x0011)**

The paper in the CR station is out. (This message is available only on the IBM 4689 Model 3x1 and TD5 printers.)

##### **PosCHANGE\_CR\_OVERHEAT (0x0012)**

The print head on the CR station is overheating. (This message is available only on the IBM 4689 Model 3x1 and TD5 printers.)

**PosCHANGE\_SJ\_PAPER\_LOW (0x0020)**

The paper in the SJ station is low. (This message is available only on the IBM 4689-00x printers.)

**PosCHANGE\_SJ\_PAPER\_OUT (0x0021)**

The paper in the SJ station is out. (This message is available only on the IBM 4689 Model 3x1 and TD5 printers.)

**PosCHANGE\_SJ\_OVERHEAT (0x0022)**

The print head on the SJ station is overheating. (This message is available only on the IBM 4689 Model 3x1 and TD5 printers.)

**PosCHANGE\_SJ\_PAPER (0x0080)**

The SJ paper in the printer has run out of paper or paper has been added. (This message is available only on the IBM Model 3, 3R, 4, 4R and 4A printers.)

- devdes** This field is set to the device descriptor of the device that generated the event message.
- status2** Indicates the current status of the printer. See "PosNprintStatus" on page 21-46 for more information.

**Remarks**

After the application receives a POSM\_PRN\_STATUS\_CHANGE event message, it should check the present status of the printer and initiate the appropriate action.

---

## POSM\_RS232\_BREAK\_DETECTED

### Description

This event message is generated when the RS-232C device handler has received a break signal from the RS-232C port.

**Note:** The RS-232C hardware on a Feature E card will only return an incoming break signal to the application if it is followed by at least one byte of data. Therefore, the device sending a break signal to the point-of-sale RS-232C port must follow it by at least one byte of data. Two event messages would then be received on the input queue. One indicates that data is available, and the other indicates that a break signal has been detected.

### POSQMSG Fields

#### 32-bit version:

<b>mp1</b>	reserved (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (long)

#### 16-bit version:

<b>mp1</b>	reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (long)

The details of the fields in **mp1** and **mp2** are:

**devdes** This field is set to the device descriptor of the device that generated the event message.

## POSM\_RS232\_DATA\_AVAIL

### Description

This event message is generated when the RS-232C device handler has received data from the RS-232C port.

### POSQMSG Fields

**32-bit version:**

**mp1**

reserved (unsigned short)  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**

reserved (unsigned short)  
 length (unsigned short)

**16-bit version:**

**mp1**

reserved (unsigned char)  
 devdes (unsigned char)

**mp2**

reserved (unsigned short)  
 length (unsigned short)

The details of the fields in **mp1** and **mp2** are:

- devdes**      This field is set to the device descriptor of the device that generated the event message.
- length**      This field contains the size, in bytes, of the buffer required to read the data associated with this event.

### Remarks

After the application receives a POSM\_RS232\_DATA\_AVAIL event message, it should use the *PosRead()* subroutine to retrieve the data.

---

## POSM\_RS232\_XMIT\_ABORT

### Description

This event message is generated when the previous *PosWrite()* request has ended abnormally.

This can happen when a device is closed with an outstanding *PosWrite()* request that has not completed. It can also occur if the device does not respond within 10 seconds. Data might or might not have been partially transmitted. Your application should take appropriate steps depending on the nature of the attached device.

### POSQMSG Fields

#### 32-bit version:

**mp1**  
 reserved (unsigned short)  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 reserved (long)

#### 16-bit version:

**mp1**  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 reserved (long)

The details of the fields in **mp1** and **mp2** are:

**devdes** This field is set to the device descriptor of the device that generated the event message.

### Remarks

After the application receives a POSM\_RS232\_XMIT\_ABORT event message, it can re-issue the *PosWrite()* request once the cause for cancelling the write has been corrected.

## POSM\_RS232\_XMIT\_COMPLETE

### Description

This event message is generated when the previous *PosWrite()* request has completed.

### POSQMSG Fields

**32-bit version:**

<b>mp1</b>	reserved (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (long)

**16-bit version:**

<b>mp1</b>	reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (long)

The details of the fields in **mp1** and **mp2** are:

**devdes**            This field is set to the device descriptor of the device that generated the event message.

### Remarks

After the application receives a POSM\_RS232\_XMIT\_COMPLETE event message, it can issue the next *PosWrite()* request.

---

## POSM\_SCAN\_DATA\_AVAIL

### Description

This event message is generated when the scanner device handler has received bar code data from the scanner.

### POSQMSG Fields

#### 32-bit version:

##### mp1

reserved (unsigned short)  
reserved (unsigned char)  
devdes (unsigned char)

##### mp2

reserved (unsigned short)  
length (unsigned short)

#### 16-bit version:

##### mp1

reserved (unsigned char)  
devdes (unsigned char)

##### mp2

reserved (unsigned short)  
length (unsigned short)

The details of the fields in **mp1** and **mp2** are:

- devdes** This field is set to the device descriptor of the device that generated the event message.
- length** This field contains the size, in bytes, of the buffer required to read the data associated with this event.

### Remarks

After the application receives a `POSM_SCAN_DATA_AVAIL` event message, it should use the `PosRead()` subroutine to retrieve the bar code data.

If bar code data is received from the scanner while the scanner is not acquired by an application, the data belongs to the next application that acquires the scanner. That application also receives a `POSM_SCAN_DATA_AVAIL` event message for each set of bar code data that is queued by the scanner device handler.

## POSM\_SYS\_DEVICE\_OFFLINE

### Description

This event message is generated when a device that was previously online goes offline.

### POSQMSG Fields

**32-bit version:**

<b>mp1</b>	slot (unsigned char) port (unsigned char) device (unsigned char) type (unsigned char)
<b>mp2</b>	reserved (unsigned short) reserved (unsigned char) subtype (unsigned char)

**16-bit version:**

<b>mp1</b>	device (unsigned char) type (unsigned char)
<b>mp2</b>	slot (unsigned char) port (unsigned char) reserved (unsigned char) subtype (unsigned char)

The details of the fields in **mp1** and **mp2** are:

- slot** This field is equivalent to the value of the **PosNslotNumber** resource that would be passed to open the device using the *PosOpen()* subroutine call.
- port** This field is equivalent to the value of the **PosNportNumber** resource that would be passed to open the device using the *PosOpen()* subroutine call.
- device** This field is equivalent to the value of the **PosNdeviceNumber** resource that would be passed to open the device using the *PosOpen()* subroutine call.
- type** This field indicates the type of device that generated the event message. The valid values for this field are defined by the set of PosTYPE\_xxxx constants in the header file device.h.
- subtype** This field is used by certain device numbers when the device number itself is not sufficient to differentiate between a set of devices. For example, several different point-of-sale keyboards all have the same device number (0x1C). In this case, the **subtype** field is used to distinguish between them.  
  
This field is significant only when it is paired with a specific device number, because different device numbers use the same range of subtype values to distinguish their devices. When it is not

necessary to specify a subtype, this field is set to zero (PosSUBTYPE\_NONE). The subtype values, where appropriate, are listed along with the device description on the **PosNdeviceNumber** resource.

## Remarks

Devices can go offline for a variety of reasons, for example:

- The device encountered an error and is resetting itself.
- The device failed or encountered a severe error.

The POSM\_SYS\_DEVICE\_OFFLINE event message is never delivered for the devices attached by way of the system keyboard port because the IBM Point of Sale Subsystem cannot detect that these devices have gone offline.

## POSM\_SYS\_DEVICE\_ONLINE

### Description

This event message is generated when a device comes online and is ready to be used. An application can use this event message to determine which devices are attached to the system and which devices to open.

### POSQMSG Fields

**32-bit version:**

**mp1**

slot (unsigned char)  
 port (unsigned char)  
 device (unsigned char)  
 type (unsigned char)

**mp2**

reserved (unsigned short)  
 reserved (unsigned char)  
 subtype (unsigned char)

**16-bit version:**

**mp1**

device (unsigned char)  
 type (unsigned char)

**mp2**

slot (unsigned char)  
 port (unsigned char)  
 reserved (unsigned char)  
 subtype (unsigned char)

The details of the fields in **mp1** and **mp2** are:

- slot** This field is equivalent to the value of the **PosNslotNumber** resource that would be passed to open the device using the *PosOpen()* subroutine call. (See the example in “PosOpen()” on page 18-10 for sample coding.)
- port** This field is equivalent to the value of the **PosNportNumber** resource that would be passed to open the device using the *PosOpen()* subroutine call. (See the example in “PosOpen()” on page 18-10 for sample coding.)
- device** This field is equivalent to the value of the **PosNdeviceNumber** resource that would be passed to open the device using the *PosOpen()* subroutine call. (See the example in “PosOpen()” on page 18-10 for sample coding.)
- type** This field indicates the type of device that generated the event message. The valid values for this field are defined by the set of PosTYPE\_xxxx constants in the header file device.h.
- subtype** This field is used by certain device numbers when the device number itself is not sufficient to differentiate between a set of devices. For example, several different point-of-sale keyboards all have the same device number (0x1C). In this case, the **subtype** field is used to distinguish between them.

This field is significant only when it is paired with a specific device number, because different device numbers use the same range of subtype values to distinguish devices. When it is not necessary to specify a subtype, this field is set to zero (`PosSUBTYPE_NONE`). The subtype values, where appropriate, are listed along with the device description on the **PosNdeviceNumber** resource.

## Remarks

On OS/2, the first application to successfully call the *PosInitialize()* subroutine also causes one or more programs that service the devices to be started. These programs determine what devices are attached, and bring the attached devices online. This can take several minutes, so the first application receives a `POSM_SYS_DEVICE_ONLINE` event messages as each individual device is found and comes online.

The second and subsequent applications to call the *PosInitialize()* subroutine successfully receive a `POSM_SYS_DEVICE_ONLINE` event message immediately for all devices that are currently online.

On Microsoft Windows, most devices will have been detected before any application issues *PosInitialize()*. However, some devices (e.g. Touch) take longer to initialize and may not be immediately online.

Your application should not attempt to use any device for which it has not received as `POSM_SYS_DEVICE_ONLINE` message. Your application should also be prepared to handle a `DEVICE_ONLINE` message at any time.

## POSM\_SYS\_DEVICE\_RELEASED

### Description

This event message is generated when a device previously requested by the application has been released. This event message is only sent to applications that have attempted to acquire the device and have received the 313 POSERR\_SYS\_ACQUIRED\_BY\_OTHER event message. The event message is placed on the message queue that was specified on the *PosOpen()* subroutine call for the device that the application attempted to acquire. If the application attempted to acquire more than one device connection that references the same device, the event message is sent to the message queue that was specified for each device connection.

Upon receipt of this message, an application must still acquire the device before using it. Because other applications might be waiting to use the device, the device might already be acquired by the time the current application attempts to acquire it. In this case, the 313 POSERR\_SYS\_ACQUIRED\_BY\_OTHER error code is returned to the application, and the application is notified when the current owner releases it.

### POSQMSG Fields

**32-bit version:**

<b>mp1</b>	reserved (unsigned short) reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (long)

**16-bit version:**

<b>mp1</b>	reserved (unsigned char) devdes (unsigned char)
<b>mp2</b>	reserved (long)

The details of the fields in **mp1** and **mp2** are:

**devdes** This field is set to the device descriptor of the device that the application attempted to acquire.

---

## POSM\_TILL\_CLOSED

### Description

This event message is generated when the cash drawer till physically closes.

### POSQMSG Fields

#### 32-bit version:

##### mp1

reserved (unsigned short)  
reserved (unsigned char)  
devdes (unsigned char)

##### mp2

reserved (long)

#### 16-bit version:

##### mp1

reserved (unsigned char)  
devdes (unsigned char)

##### mp2

reserved (long)

The details of the fields in **mp1** and **mp2** are described below:

**devdes** This field is set to the device descriptor of the device that generated the event message.

### Remarks

This event is given to the application that has the cash drawer acquired at the time the event is generated.

## POSM\_TILL\_OPENED

### Description

This event message is generated when the cash drawer till physically opens.

### POSQMSG Fields

**32-bit version:**

**mp1**  
 reserved (unsigned short)  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 reserved (long)

**16-bit version:**

**mp1**  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 reserved (long)

The details of the fields in **mp1** and **mp2** are:

**devdes**      This field is set to the device descriptor of the device that generated the event message.

### Remarks

This event is given to the application that has the cash drawer acquired at the time the event is generated.

---

## POSM\_TOUCH\_DATA

### Description

This event message is generated when a touch event occurs.

### POSQMSG Fields

#### 32-bit version:

**mp1**  
 status (unsigned short)  
 reserved (unsigned char)  
 devdes (unsigned char)

**mp2**  
 x (unsigned short)  
 y (unsigned short)

#### 16-bit version:

**mp1**  
 status (unsigned char)  
 devdes (unsigned char)

**mp2**  
 x (unsigned short)  
 y (unsigned short)

The details of the fields in **mp1** and **mp2** are:

- status** This field indicates the status of the touch device. The values and their meanings are:  
**PosTOUCH\_BUTTON1\_DOWN (0x01)**  
 The touch screen has been pressed.  
**PosTOUCH\_ERROR (0x0010)**  
 An error occurred reading touch information
- devdes** This field is set to the device descriptor of the device that generated the event message.
- x** This field indicates the **x** coordinate where the screen was touched.
- y** This field indicates the **y** coordinate where the screen was touched.

---

## WM\_CHAR

### Description

This message is generated by OS/2 Presentation Manager for Presentation Manager applications or Microsoft Windows for Microsoft Windows applications. For OS/2, see the *OS/2 2.0 Technical Library: Presentation Manager Programming Reference* for more information about the WM\_CHAR message.

### POSQMSG Fields

#### OS/2:

- mp1** The same as defined for OS/2 Presentation Manager.
- mp2** The same as defined for OS/2 Presentation Manager.

#### Microsoft Windows:

- mp1** The same as defined for the *wParam* for the Microsoft Windows operating system.
- mp2** The same as defined for the *lParam* for the Microsoft Windows operating system.

### Remarks

After the application receives a WM\_CHAR event message, it should examine the **mp1** and **mp2** message parameters to determine what the key is. This event is generated for both the standard keys and the point-of-sale-unique keys for the alphanumeric point-of-sale keyboards when they are system keyboards.

**Note:** OS/2 Presentation Manager will set the KC\_VIRTUALKEY flag on in the WM\_CHAR message for some point-of-sale-unique keys. On the ANPOS keyboard, these keys are 107 and 111. On the Retail Alphanumeric Point of Sale Keyboard with Card Reader, these keys are 107 and 124. See the sample code in C:\POS\SAMPLE\ANPOSKEY\ANPOSKEY.C for an example of how to distinguish between the point-of-sale-unique keys and a virtual key.

## Chapter 21. Resource Sets

Resource Access Codes . . . . .	21-5
PosSystem Resource Set . . . . .	21-5
PosNqueueHandle . . . . .	21-6
PosNreadTimeout . . . . .	21-6
PosNvitalProductData . . . . .	21-7
PosDevice Resource Set . . . . .	21-7
PosNdeviceNumber . . . . .	21-8
PosNportNumber . . . . .	21-14
PosNqueueHandle . . . . .	21-14
PosNslotNumber . . . . .	21-15
PosAlarm Resource Set . . . . .	21-16
PosNalarmStatus . . . . .	21-16
PosDisplay Resource Set . . . . .	21-16
PosNcharSize . . . . .	21-17
PosNdisplayCodePage . . . . .	21-17
PosNdisplayCursor . . . . .	21-19
PosNdisplayMode . . . . .	21-19
PosNdisplayLightsOn . . . . .	21-20
PosNpixelX . . . . .	21-21
PosNpixelY . . . . .	21-21
PosDrawer Resource Set . . . . .	21-21
PosNpulseWidth . . . . .	21-21
PosNtillStatus . . . . .	21-22
PosKeyboard Resource Set . . . . .	21-22
PosNdoubleKey01 - PosNdoubleKey60 . . . . .	21-23
PosNfatFingerTimeOut . . . . .	21-24
PosNkeyboardClick . . . . .	21-25
PosNkeyLock . . . . .	21-25
PosNkeyboardLightsOn . . . . .	21-26
PosNnumpadLocation . . . . .	21-26
PosNnumpadStyle . . . . .	21-27
PosNnumpadZero . . . . .	21-27
PosNtoneDuration . . . . .	21-28
PosNtoneFreq . . . . .	21-28
PosNtoneVolume . . . . .	21-28
PosNtypematicDelay . . . . .	21-29
PosNtypematicFreq . . . . .	21-29
PosMsr Resource Set . . . . .	21-30
PosNvram Resource Set . . . . .	21-30
PosNnvramCursor . . . . .	21-30
PosNnvramMode . . . . .	21-31
PosNnvramSize . . . . .	21-31
PosPower Resource Set . . . . .	21-31
PosNpowerAlarm . . . . .	21-31
PosPrinter Resource Set . . . . .	21-32
PosNcodePage . . . . .	21-33
PosNCRWidth . . . . .	21-35
PosNdiOrientation . . . . .	21-35
PosNdiWidth . . . . .	21-35
PosNfeedDirection . . . . .	21-36
PosNfiscalCountry . . . . .	21-37
PosNfiscalNotify . . . . .	21-37
PosNfiscalPLDStatus . . . . .	21-37

PosNfiscalVersion . . . . .	21-38
PosNheadParkedPosition . . . . .	21-38
PosNinterleaved . . . . .	21-39
PosNleftMarginCR . . . . .	21-39
PosNlineFeedCR . . . . .	21-40
PosNlineFeedDI . . . . .	21-40
PosNlineFeedSJ . . . . .	21-41
PosNprintAlignment . . . . .	21-42
PosNprintColorMode . . . . .	21-42
PosNprintCRCharSetx . . . . .	21-42
PosNprintDICharSetx . . . . .	21-43
PosNprintFeatures . . . . .	21-43
PosNprintMode . . . . .	21-44
PosNprintQualityMode . . . . .	21-44
PosNprintStation . . . . .	21-45
PosNprintStatus . . . . .	21-46
PosNprintStatus2 . . . . .	21-47
PosNprintTabStops . . . . .	21-47
PosNprintToneDuration . . . . .	21-48
PosNprintToneFrequency . . . . .	21-48
PosNprintToneNote . . . . .	21-48
PosNprintToneOctave . . . . .	21-49
PosNprintToneVolume . . . . .	21-49
PosNprintUpsideDown . . . . .	21-50
PosNrawPrintStatus . . . . .	21-50
PosNresumeString . . . . .	21-50
PosNretryString . . . . .	21-52
Model 2, Model 3, Model 3F, Model 3R, Model 4, Model 4A, and Model 4R Printers . . . . .	21-52
4689-001 and 4689-002 Printers . . . . .	21-52
4689 Model 3x1 and TD5 printers . . . . .	21-52
4610 SureMark Point of Sale Printers (all models) . . . . .	21-53
PosRs232c Resource Set . . . . .	21-53
PosNbaudRate . . . . .	21-53
PosNdataBits . . . . .	21-54
PosNlineMode . . . . .	21-54
PosNparity . . . . .	21-55
PosNrs232Status . . . . .	21-55
PosNstopBits . . . . .	21-55
PosNtimeoutChar . . . . .	21-56
PosScale Resource Set . . . . .	21-56
PosNdisplayRequired . . . . .	21-57
PosNnumWeightDigits . . . . .	21-57
PosNoperMode . . . . .	21-58
PosNvibrationFilter . . . . .	21-58
PosNweightMode . . . . .	21-59
PosNzeroIndState . . . . .	21-59
PosNzeroRetState . . . . .	21-60
PosScanner Resource Set . . . . .	21-60
Scanner Model Identifiers . . . . .	21-62
PosNbarCodes1 . . . . .	21-62
PosNbarCodes2 . . . . .	21-64
PosNbarCodes3 . . . . .	21-65
PosNbarCodes4 . . . . .	21-66
PosNbarCodeProgramming . . . . .	21-67
PosNbeepFreq . . . . .	21-67

PosNbeepLength . . . . .	21-67
PosNbeepState . . . . .	21-68
PosNbeepVolume. . . . .	21-68
PosNblinkLength . . . . .	21-69
PosNblockReadMode . . . . .	21-69
PosNblock1Type . . . . .	21-70
PosNblock2Type . . . . .	21-71
PosNblock3Type . . . . .	21-71
PosNbVolSwitchState . . . . .	21-72
PosNcheckModulo . . . . .	21-72
PosNcode128ScansPerRead . . . . .	21-73
PosNcode39ScansPerRead . . . . .	21-73
PosNdecodeAlgorithm . . . . .	21-73
PosNdReadTimeout . . . . .	21-74
PosNdTouchMode . . . . .	21-74
PosNeAN13ScansPerRead . . . . .	21-75
PosNeAN8ScansPerRead . . . . .	21-75
PosNiTFLength1 . . . . .	21-76
PosNiTFLength2 . . . . .	21-77
PosNiTFLengthType. . . . .	21-77
PosNiTFScansPerRead . . . . .	21-78
PosNjANTwoLabelDecode . . . . .	21-78
PosNlabelsQueued . . . . .	21-78
PosNlaserSwitchState . . . . .	21-79
PosNlaserTimeout . . . . .	21-79
PosNmotorTimeout . . . . .	21-80
PosNqueueAllLabels . . . . .	21-80
PosNscansPerRead . . . . .	21-81
PosNstoreScansPerRead . . . . .	21-81
PosNsupplementals . . . . .	21-82
PosNtransmitCheckDigit . . . . .	21-82
PosNtwoLabelFlagPair1 . . . . .	21-83
PosNtwoLabelFlagPair2 . . . . .	21-84
PosNtwoLabelFlagPair3 . . . . .	21-85
PosNtwoLabelFlagPair4 . . . . .	21-86
PosNuPCAScansPerRead . . . . .	21-87
PosNuPCDScansPerRead . . . . .	21-87
PosNuPCEscansPerRead . . . . .	21-88
PosNuPCExpansion . . . . .	21-88
PosNverifyPriceChk . . . . .	21-89
PosTouch Resource Set . . . . .	21-89
PosNtouchBackLightOnEvent . . . . .	21-90
PosNtouchBrightness . . . . .	21-90
PosNtouchClickVolume. . . . .	21-91
PosNtouchContrast . . . . .	21-91
PosNtouchEntryClick . . . . .	21-91
PosNtouchExitClick . . . . .	21-92
PosNtouchMaxX . . . . .	21-92
PosNtouchMaxY . . . . .	21-92
PosNtouchMode . . . . .	21-92
PosNtouchScreenSaverTime . . . . .	21-93
PosNtouchToneDuration . . . . .	21-93
PosNtouchToneFreq. . . . .	21-93

## Resource Sets

created on October 2, 2001

PosNtouchToneVolume. . . . . 21-94

This chapter contains reference information for the resource sets used with the IBM Point of Sale Subsystem. See Chapter 5. General Point of Sale Device Programming for information about using these resources sets.

## Resource Access Codes

The tables that contain lists of each resource set use access codes to describe the ways in which the resources can be set. The codes are:

- C** The resource can be set when the device is opened. The resource value can come from a resource file or through the *args* parameter on the *PosOpen()* subroutine. For resources in the PosSystem Resource Set resource set, the value can come from the resource file or through the *args* parameter on the *PosInitialize()* subroutine.
- S** The resource can be set using the *PosIOctl()* subroutine with the `POS_SYS_SET_VALUES` control function or overridden through the *args* and *nargs* parameters on some other *PosIOctl()* request.
- G** The resource can be retrieved using the *PosIOctl()* subroutine with the `POS_SYS_GET_VALUES` control function.

---

## PosSystem Resource Set

**Class Name** PosSystem  
**Include** <pos/device.h>

The PosSystem resource set controls all system-wide aspects of the IBM Point of Sale Subsystem. The resources in this set determine the non-device specific behavior of the system. Because resources in this resource set are not associated with a device, use a device descriptor of 0 (zero) when making `POS_SYS_GET_VALUES` or `POS_SYS_SET_VALUES` *PosIOctl()* requests.

Unless otherwise noted, PosSystem resources can be specified in the resource file just like device specific resources. Names in the resource file can be prefixed by the application name and the device instance name in order to tailor resources to a specific device instance. To allow this for PosSystem resources, use a device instance name of AIPSYS in the resource file. For example:

```
!
! Default value unless overridden (return immediately)
!
*readTimeout: 0

!
! Value for a specific application (1 second)
!
CheckoutApp1.AIPSYS.readTimeout: 1000
```

The following table gives an overview of the resources in this set.

Table 21-1. PosSystem Resources

Name	Type	Default	Access
PosNqueueHandle	long	0	CG
PosNreadTimeout	long	0	CGS
PosNvitalProductData	unsigned char *	0	G

See “Resource Access Codes” for descriptions of the access codes.

## PosNqueueHandle

<b>Type</b>	long
<b>Default</b>	0
<b>Access</b>	CG

The **PosNqueueHandle** resource specifies the input queue that is to receive event messages generated by the IBM Point of Sale Subsystem. This resource is specified on the *PosInitialize()* subroutine call.

**Note:** The **PosNqueueHandle** resource is ignored on systems that use the IBM Point of Sale Subsystem for Linux. Linux systems must use the IBM Point of Sale Subsystem input queue.

The value for the **PosNqueueHandle** resource indicates the queue on which to place non-device specific event messages (such as system event messages). This resource is a system-wide resource. Querying the value of this resource is done differently than for device specific resources, because there is no device descriptor to use when calling *Chapter 19. PosIOctl() Requests*. Use a device descriptor of 0 (zero) when making a *POS\_SYS\_GET\_VALUES Chapter 19. PosIOctl() Requests* request to query the value of this resource used to post system event messages.

A **PosNqueueHandle** value of 0 (zero) is used to indicate the IBM Point of Sale Subsystem input queue is to be used instead of a presentation facility queue. A value other than 0 (zero) indicates that a presentation facility queue is to be used for posting event messages. For Presentation Manager, set this resource to the value returned by the *WinCreateMsgQueue()* subroutine call. For the Microsoft Windows operating system, set this resource to the value returned by the *CreateWindow()* subroutine call.

If the application closes the presentation facility queue after passing it to the *PosInitialize()* subroutine call, asynchronous event messages cannot be queued and errors are logged.

This resource cannot be specified in the resource file because any values other than the default value must be determined by the application at run time.

### Attention

On OS/2, the IBM Point of Sale Subsystem for OS/2 attempts to verify that the queue handle is valid, and returns a 308 *POSERR\_SYS\_INVALID\_QUEUE* return code if it is not valid. In OS/2, an incorrect value for this resource can sometimes cause Presentation Manager to end abnormally when performing the validation. Applications must make every effort to ensure that the queue handle that is passed in is valid.

## PosNreadTimeout

<b>Type</b>	long
<b>Default</b>	0
<b>Access</b>	CGS

By default, when an application reads the IBM Point of Sale Subsystem input queue with *PosRead()*, the subroutine returns immediately if there is no data to be read. Using this method of getting input is inefficient because an application must continually poll the input queue to determine if data is available.

The **PosNreadTimeout** resource allows an application to call *PosRead()* and wait for data to become available rather than continually polling for data. The value of this resource is equivalent to the number of milliseconds that the IBM Point of Sale Subsystem will wait for data before returning to the application. If data is available when the *PosRead()* subroutine is called or if data becomes available before the time-out has expired, then that data will be returned to the application. If the time-out expires and no data is available, then the 329 POSERR\_SYS\_TIMEOUT error will be returned to the application.

Two values have special meaning when used with the **PosNreadTimeout** resource:

- 0** If the resource value is 0 (zero) and there is no data available, the *PosRead()* subroutine returns immediately and no error is returned.
- 1** A resource value of -1 will cause the *PosRead()* subroutine to wait indefinitely for data to come available. If no data ever becomes available, the subroutine will not return.

This resource can be specified in the resource file.

**Notes:**

1. This resource is only used when the application reads the IBM Point of Sale Subsystem input queue. It is not used when calling *PosRead()* for other device descriptors or for the presentation facility queue.
2. This resource is ignored for 16-bit applications for the Microsoft Windows operating system and for 32-bit applications using Microsoft Win32s on Microsoft Windows 3.1.

## PosNvitalProductData

<b>Type</b>	unsigned char *
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNvitalProductData** resource allows an application to request the product data (machine type, serial number, BIOS level) for the system. This resource is only valid for 4694 models with model type 244 or later and BIOS level 9.12 or later. Twenty-four bytes of data are returned for this resource. The structure, PosVPD, describes the format of the returned data (see the header file, pos.h).

**Notes:**

1. Use a device descriptor of 0 (zero) when making a POS\_SYS\_GET\_VALUES request to query the value of this resource.
2. This resource cannot be set and has no effect if specified in the resource file.

---

## PosDevice Resource Set

<b>Class Name</b>	PosDevice
<b>Include</b>	<pos/device.h>

The PosDevice resource set controls all aspects of the devices that are supported by the IBM Point of Sale Subsystem and that are connected to the system. The resources in this set determine the physical connection between the device and the point-of-sale terminal. As a result, these values can only be set when the device is opened. Applications can query the values, but are not allowed to modify them.

Because all other resource sets inherit members from this resource set, all other sets contain the resources in this set as well as their own additions.

The following table gives an overview of the resources in this set.

Table 21-2. PosDevice Resources

Name	Type	Default	Access
PosNdeviceNumber	char	none	CG
PosNportNumber	char	PosPORT_1	CG
PosNqueueHandle	long	0	CG
PosNslotNumber	char	PosSLOT_5	CG

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## PosNdeviceNumber

<b>Type</b>	char
<b>Default</b>	none
<b>Access</b>	CG

The **PosNslotNumber**, **PosNportNumber**, and the **PosNdeviceNumber** resources together uniquely identify each point-of-sale device found by the IBM Point of Sale Subsystem. The **PosNdeviceNumber** resource can have the following values:

### **PosDEVICE\_ALARM (0x84)**

An alarm. This device can only be attached to socket 3B.

### **PosDEVICE\_ALPHANUMERIC\_DISPLAY\_A (0x20)**

Vacuum Fluorescent Display with two rows of 20 alphanumeric characters

### **PosDEVICE\_ALPHANUMERIC\_DISPLAY\_B (0x21)**

Vacuum Fluorescent Display with two rows of 20 alphanumeric characters

### **PosDEVICE\_ANOP\_DISPLAY\_A (0x24)**

One of the following displays:

- 40-Character Liquid Crystal Display II (PosDSP\_SUBTYPE\_2X20\_LCD)
- 40-Character Liquid Crystal Display II (PosDSP\_SUBTYPE\_2X20\_LCD\_I)
- 40-Character Vacuum Fluorescent Display II (PosDSP\_SUBTYPE\_2X20\_VFD)
- 2x20 Character VFD Customer Display (PosDSP\_SUBTYPE\_2X20\_VFD)
- Two-Sided Vacuum Fluorescent Display II (PosDSP\_SUBTYPE\_2\_SIDE\_VFD)

### **PosDEVICE\_ANOP\_DISPLAY\_B (0x25)**

One of the following displays:

- 40-Character Liquid Crystal Display II (PosDSP\_SUBTYPE\_2X20\_LCD)
- 40-Character Liquid Crystal Display II (PosDSP\_SUBTYPE\_2X20\_LCD\_I)

- 40-Character Vacuum Fluorescent Display II (PosDSP\_SUBTYPE\_2X20\_VFD)
- 2x20 Character VFD Customer Display (PosDSP\_SUBTYPE\_2X20\_VFD)
- Two-Sided Vacuum Fluorescent Display II (PosDSP\_SUBTYPE\_2\_SIDE\_VFD)

**PosDEVICE\_APA\_DISPLAY\_A (0x2A)**

Character and Graphics Display:

- Single-byte (PosSUBTYPE\_NONE)
- Japan (PosDSP\_SUBTYPE\_APA\_JAPAN)
- Korea (PosDSP\_SUBTYPE\_APA\_KOREA)
- Traditional Chinese (PosDSP\_SUBTYPE\_APA\_TRAD\_CHINESE)
- Simplified Chinese (PosDSP\_SUBTYPE\_APA\_SIMP\_CHINESE)

**PosDEVICE\_APA\_DISPLAY\_B (0x2B)**

Character and Graphics Display:

- Single-byte (PosSUBTYPE\_NONE)
- Japan (PosDSP\_SUBTYPE\_APA\_JAPAN)
- Korea (PosDSP\_SUBTYPE\_APA\_KOREA)
- Traditional Chinese (PosDSP\_SUBTYPE\_APA\_TRAD\_CHINESE)
- Simplified Chinese (PosDSP\_SUBTYPE\_APA\_SIMP\_CHINESE)

**PosDEVICE\_APA\_DISPLAY\_C (0x2C)**

Character/Graphics Display attached to the PLU Extension Box:

- Single-byte (PosSUBTYPE\_NONE)
- Japan (PosDSP\_SUBTYPE\_APA\_JAPAN)
- Korea (PosDSP\_SUBTYPE\_APA\_KOREA)
- Traditional Chinese (PosDSP\_SUBTYPE\_APA\_TRAD\_CHINESE)
- Simplified Chinese (PosDSP\_SUBTYPE\_APA\_SIMP\_CHINESE)

**PosDEVICE\_APA\_DISPLAY\_D (0x2D)**

Character/Graphics Display attached to the PLU Extension Box:

- Single-byte (PosSUBTYPE\_NONE)
- Japan (PosDSP\_SUBTYPE\_APA\_JAPAN)
- Korea (PosDSP\_SUBTYPE\_APA\_KOREA)
- Traditional Chinese (PosDSP\_SUBTYPE\_APA\_TRAD\_CHINESE)
- Simplified Chinese (PosDSP\_SUBTYPE\_APA\_SIMP\_CHINESE)

**PosDEVICE\_ANPOS\_KEYBOARD\_A (0x1A)**

Alphanumeric Point of Sale Keyboard

**PosDEVICE\_ANPOS\_KEYBOARD\_B (0x1B)**

Alphanumeric Point of Sale Keyboard

**PosDEVICE\_CASH\_DRAWER\_A (0x54)**

A cash drawer attached to socket 3A.

**PosDEVICE\_CASH\_DRAWER\_B (0x83)**

A cash drawer attached to socket 3B.

**PosDEVICE\_CHECKOUT\_KEYBOARD\_A (0x10)**

One of the following checkout keyboards:

- 50-Key Modifiable Layout Keyboard
- 50-Key Modifiable Layout Keyboard/Operator Display

**PosDEVICE\_CHECKOUT\_KEYBOARD\_B (0x11)**

One of the following checkout keyboards:

- 50-Key Modifiable Layout Keyboard
- 50-Key Modifiable Layout Keyboard/Operator Display

**PosDEVICE\_MSR\_1\_TRACK\_A (0x40)**

One-track MSR

**PosDEVICE\_MSR\_1\_TRACK\_B (0x41)**

One-track MSR

**PosDEVICE\_MSR\_2\_TRACK\_A (0x46)**

Two-track MSR

**PosDEVICE\_MSR\_2\_TRACK\_B (0x47)**

Two-track MSR

**PosDEVICE\_MSR\_3\_TRACK\_A (0x48)**

Three-track MSR

**PosDEVICE\_MSR\_3\_TRACK\_B (0x49)**

Three-track MSR

**PosDEVICE\_MSR\_J\_TRACK\_A (0x48)**

Japanese Two-Head MSR

**PosDEVICE\_MSR\_J\_TRACK\_B (0x49)**

Japanese Two-Head MSR

**PosDEVICE\_NVRAM\_A (0x50)**

NVRAM in an attached 4683-x02

**PosDEVICE\_NVRAM\_B (0x51)**

NVRAM in an attached 4693-2x2

**PosDEVICE\_NVRAM\_LOCAL (0x85)**

NVRAM in:

- 4684-300
- 4693 (all models except 2x2)
- 4694 (all models)
- 4695 (all Integrated models)
- 4695 Point of Sale Adapter
- 4695 Point of Sale Adapter/A
- 7497-001 Point of Sale Attachment Adapter

**PosDEVICE\_OPERATOR\_DISPLAY\_A (0x22)**

LCD display with two rows of 20 alphanumeric characters

**PosDEVICE\_OPERATOR\_DISPLAY\_B (0x23)**

LCD display with two rows of 20 alphanumeric characters

**PosDEVICE\_PLU\_KEYBOARD (0x19)**

PLU Keyboard/Display-III on the PLU Extension Box

**PosDEVICE\_POS\_KEYBOARD\_A (0x1C)**

One of the following point-of-sale keyboards:

- Retail Alphanumeric Point of Sale Keyboard with Card Reader (PosKBD\_SUBTYPE\_ANPOS\_2)
- Retail Point of Sale Keyboard (PosKBD\_SUBTYPE\_CHECKOUT\_2)
- Retail Point of Sale Keyboard with Card Reader (PosKBD\_SUBTYPE\_CHECKOUT\_2)
- Retail Point of Sale Keyboard with Card Reader and Display (PosKBD\_SUBTYPE\_CHECKOUT\_2)
- Modifiable Layout Keyboard with Card Reader (PosKBD\_SUBTYPE\_MODIFIABLE\_LAYOUT)
- PC Point of Sale Keyboard (PosKBD\_SUBTYPE\_PC\_POS)
- Point of Sale Keyboard V (PosKBD\_SUBTYPE\_KEYBOARD\_V)
- Point of Sale Keyboard VI (PosKBD\_SUBTYPE\_KEYBOARD\_VI)

- 4685 Point of Sale Keyboard Model K01 (PosKBD\_SUBTYPE\_4685\_K01)
- IBM 4820 SurePoint Keypad (PosKBD\_SUBTYPE\_4820\_NO\_KEYS)
- IBM 4820 SurePoint Keypad (PosKBD\_SUBTYPE\_4820\_KEYPAD)

**PosDEVICE\_POS\_KEYBOARD\_B (0x1D)**

One of the following point-of-sale keyboards:

- Retail Alphanumeric Point of Sale Keyboard with Card Reader (PosKBD\_SUBTYPE\_ANPOS\_2)
- Retail Point of Sale Keyboard (PosKBD\_SUBTYPE\_CHECKOUT\_2)
- Retail Point of Sale Keyboard with Card Reader (PosKBD\_SUBTYPE\_CHECKOUT\_2)
- Retail Point of Sale Keyboard with Card Reader and Display (PosKBD\_SUBTYPE\_CHECKOUT\_2)
- Modifiable Layout Keyboard with Card Reader (PosKBD\_SUBTYPE\_MODIFIABLE\_LAYOUT)
- PC Point of Sale Keyboard (PosKBD\_SUBTYPE\_PC\_POS)
- Point of Sale Keyboard V (PosKBD\_SUBTYPE\_KEYBOARD\_V)
- Point of Sale Keyboard VI (PosKBD\_SUBTYPE\_KEYBOARD\_VI)
- 4685 Point of Sale Keyboard Model K01 (PosKBD\_SUBTYPE\_4685\_K01)
- IBM 4820 SurePoint Keypad (PosKBD\_SUBTYPE\_4820\_NO\_KEYS)
- IBM 4820 SurePoint Keypad (PosKBD\_SUBTYPE\_4820\_KEYPAD)

**PosDEVICE\_POWER\_B (0x86)**

Programmable power device that controls the power supply to an attached 4693-2x2

**PosDEVICE\_POWER\_LOCAL (0x87)**

Programmable power device that controls the power supply to a 4693 (all models except 2x2)

**PosDEVICE\_PRINTER\_2 (0x30)**

Model 2 printer

**PosDEVICE\_PRINTER\_4689\_THERMAL (0x32)**

One of the following point-of-sale printers:

- The IBM 4689 Point of Sale Printer Models 3x1 and TD5 (PosPRN\_SUBTYPE\_4689\_301)

**PosDEVICE\_PRINTER\_3 (0x34)**

One of the following point-of-sale printers:

- Model 3 printer
- Model 3R printer

**PosDEVICE\_PRINTER\_4 (0x34)**

One of the following point-of-sale printers:

- Model 4 printer (PosSUBTYPE\_NONE)
- Model 4A printer (PosPRN\_SUBTYPE\_MODEL\_4A)
- Model 4R printer (PosSUBTYPE\_NONE)

**PosDEVICE\_PRINTER\_4610 (0x35)**

One of the IBM 4610 SureMark Point of Sale Printers

**PosDEVICE\_PRINTER\_4689 (0x37)**

One of the following IBM 4689 point-of-sale printers:

- IBM 4689-001 Printer (PosPRN\_SUBTYPE\_4689\_1)
- IBM 4689-002 Printer (PosPRN\_SUBTYPE\_4689\_2)

**PosDEVICE\_PRINTER\_FISCAL\_3 (0x38)**

- Model 3F Fiscal Printer

- IBM 4610 SureMark Point of Sale Fiscal printers

**PosDEVICE\_RS232\_23A (0x68)**

One of the following:

- RS-232C device attached to socket 23 of the RS-232C Feature E card in slot 2A of a point-of-sale terminal
- RS-232C device which has been modified by the manufacturer to attach directly to an SIO device channel
- USB device that conforms to the IBM USB Pseudo-RS232 Interface Specification

**PosDEVICE\_RS232\_23B (0x69)**

One of the following:

- RS-232C device attached to socket 23 of the RS-232C Feature E card in slot 2B of a point-of-sale terminal
- RS-232C device that has been modified by the manufacturer to attach directly to an SIO device channel
- USB device that conforms to the IBM USB Pseudo-RS232 Interface Specification

**PosDEVICE\_RS232\_25A (0x64)**

One of the following:

- RS-232C device attached to socket 25 of the RS-232C Feature E card in slot 2A of a point-of-sale terminal
- RS-232C device that has been modified by the manufacturer to attach directly to an SIO device channel
- USB device that conforms to the IBM USB Pseudo-RS232 Interface Specification

**PosDEVICE\_RS232\_25B (0x65)**

One of the following:

- RS-232C device attached to socket 25 of the RS-232C Feature E card in slot 2B of a point-of-sale terminal
- RS-232C device that has been modified by the manufacturer to attach directly to an SIO device channel
- USB device that conforms to the IBM USB Pseudo-RS232 Interface Specification

**PosDEVICE\_RS232\_A (0x66)**

RS-232C device attached to the serial connector labeled A on a IBM 4693-2x2

**PosDEVICE\_RS232\_B (0x67)**

RS-232C device attached to the serial connector labeled B on a IBM 4693-2x2

**PosDEVICE\_SCALE\_A (0x6A)**

A scale whose interface is compatible with one of the following IBM point-of-sale scales:

- IBM 4687 Point of Sale Scanner Model 2 (PosSCALE\_SUBTYPE\_4687)
- IBM 4696 Point of Sale Scanner Scale Model 1 (PosSCALE\_SUBTYPE\_4696)
- IBM 4698 Point of Sale Scanner Model 2 (PosSCAN\_SUBTYPE\_4698)
- IBM USB Scale Interface Specification

**PosDEVICE\_SCALE (0x6E)**

One of the following point-of-sale scales:

- IBM 4687 Point of Sale Scanner Model 2 (PosSCALE\_SUBTYPE\_4687)
- IBM 4696 Point of Sale Scanner Scale Model 1 (PosSCALE\_SUBTYPE\_4696)

- IBM 4698 Point of Sale Scanner Model 2 (PosSCAN\_SUBTYPE\_4698)
- IBM USB Scale Interface Specification

**PosDEVICE\_SCANNER\_A (0x4A)**

One of the following point-of-sale scanners:

- IBM 4686 Retail Point of Sale Scanner Model 1 or 2 (PosSCAN\_SUBTYPE\_4686\_1\_2)
- IBM 4686 Retail Point of Sale Scanner Model 3 or 4 (PosSCAN\_SUBTYPE\_4686\_3\_4)
- IBM 4687 Point of Sale Scanner Model 1 or 2 (PosSCAN\_SUBTYPE\_4687)
- IBM 4696 Point of Sale Scanner Scale Model 1 (PosSCAN\_SUBTYPE\_4696)
- IBM 4697 Point of Sale Scanner Model 1 (PosSCAN\_SUBTYPE\_4697)
- IBM 4698 Point of Sale Scanner Model 1 or 2 (PosSCAN\_SUBTYPE\_4698)
- IBM USB Scanner Interface Specification

**PosDEVICE\_SCANNER\_B (0x4B)**

One of the following point-of-sale scanners:

- Hand-Held Bar Code Reader Model 1 (PosSCAN\_SUBTYPE\_HHBCR\_1)
- Hand-Held Bar Code Reader Model 2 (PosSCAN\_SUBTYPE\_HHBCR\_2)
- IBM 1520 Hand-Held Scanner Model A02 (PosSCAN\_SUBTYPE\_1520\_A02)
- IBM 4685 Hand-Held Bar Code Reader Model 001 (PosSCAN\_SUBTYPE\_HHBCR\_2)
- IBM 4685 Hand-Held Bar Code Reader Model K01 (PosSCAN\_SUBTYPE\_HHBCR\_2)
- IBM USB Scanner Interface Specification

**PosDEVICE\_SHOPPER\_DISPLAY\_A (0x26)**

Display with one row of 8 numeric characters and 6 function indicators

**PosDEVICE\_SHOPPER\_DISPLAY\_B (0x27)**

Display with one row of 8 numeric characters and 6 function indicators

**PosDEVICE\_SYSTEM\_ATTACHED\_KBD (0x81)**

Alphanumeric point-of-sale keyboard when used as the system keyboard

**PosDEVICE\_SYSTEM\_ATTACHED\_MSR (0x82)**

MSR housed in an alphanumeric point-of-sale keyboard and is the system keyboard

**PosDEVICE\_TOUCH\_SCREEN (0x5C)**

One of the following point-of-sale touch screens:

- IBM 4695 Point of Sale Distributed Touch Terminal Model 002 (PosTOUCH\_SUBTYPE\_4695\_104\_M)
- IBM 4695 Point of Sale Distributed Touch Terminal Model 012 (PosTOUCH\_SUBTYPE\_4695\_95\_C or PosTOUCH\_SUBTYPE\_4695\_104\_C)
- IBM 4695 Point of Sale Distributed Touch Terminal Model 022 (PosTOUCH\_SUBTYPE\_4695\_121\_C)
- IBM 4695 Point of Sale Distributed Touch Terminal Model 032 (PosTOUCH\_SUBTYPE\_4695\_104\_C)
- IBM 4695 Point of Sale Integrated Touch Terminal Model 201 (PosTOUCH\_SUBTYPE\_4695\_104\_M)
- IBM 4695 Point of Sale Integrated Touch Terminal Model 211 (PosTOUCH\_SUBTYPE\_4695\_95\_C or PosTOUCH\_SUBTYPE\_4695\_104\_C)

- IBM 4695 Point of Sale Integrated Touch Terminal Model 321 (PosTOUCH\_SUBTYPE\_4695\_121\_C)
- IBM 4695 Point of Sale Integrated Touch Terminal Model 331 (PosTOUCH\_SUBTYPE\_4695\_104\_C)
- IBM SurePoint Monochrome Touch Screen (PosTOUCH\_SUBTYPE\_SUREPOINT\_95\_M)
- IBM SurePoint Color Touch Screen (PosTOUCH\_SUBTYPE\_SUREPOINT\_95\_C)
- IBM SurePoint Color Touch Screen (PosTOUCH\_SUBTYPE\_SUREPOINT\_104\_C)
- IBM 4820 SurePoint Touch Screen (PosTOUCH\_SUBTYPE\_4820)

#### PosDEVICE\_TOUCH\_SCREEN (0x5D)

One of the following point-of-sale touch screens:

- IBM SurePoint Monochrome Touch Screen (PosTOUCH\_SUBTYPE\_SUREPOINT\_95\_M)
- IBM SurePoint Color Touch Screen (PosTOUCH\_SUBTYPE\_SUREPOINT\_95\_C)
- IBM SurePoint Color Touch Screen (PosTOUCH\_SUBTYPE\_SUREPOINT\_104\_C)
- IBM 4820 SurePoint Touch Screen (PosTOUCH\_SUBTYPE\_4820)

## PosNportNumber

<b>Type</b>	char
<b>Default</b>	PosPORT_1
<b>Access</b>	CG

The **PosNslotNumber**, **PosNportNumber**, and **PosNdeviceNumber** resources together uniquely identify each point-of-sale device found by IBM Point of Sale Subsystem. (See “PosOpen()” on page 18-10 for an example. usage.) The **PosNportNumber** resource can have the following values:

**PosPORT\_0 (0)**

**PosPORT\_1 (0x11)**

**PosPORT\_2 (0x22)**

The **PosNportNumber** is contained in the POSM\_SYS\_DEVICE\_ONLINE message for each device (see “POSM\_SYS\_DEVICE\_ONLINE” on page 20-22).

## PosNqueueHandle

<b>Type</b>	long
<b>Default</b>	0
<b>Access</b>	CG

The **PosNqueueHandle** resource specifies the input queue that is to receive event messages generated by point-of-sale devices. This resource is specified on the *PosOpen()* subroutine call.

**Note:** The **PosNqueueHandle** resource is ignored on systems that use the IBM Point of Sale Subsystem for Linux. Linux systems must use the IBM Point of Sale Subsystem input queue.

The resource value indicates which queue to place messages that are specific to the device being opened. Querying the value of this resource for device specific queues is the same as for all other device resources. Use the device descriptor of that device when making a `POS_SYS_GET_VALUES Chapter 19. PosIOctl() Requests` request.

A **PosNqueueHandle** value of 0 (zero) is used to indicate the IBM Point of Sale Subsystem input queue is to be used instead of a presentation facility queue. A value other than 0 (zero) indicates that a presentation facility queue is to be used for posting event messages. For Presentation Manager, set this resource to the value returned by the `WinCreateMsgQueue()` subroutine call. For applications for the Microsoft Windows operating system, set this resource to the value returned by the `CreateWindow()` subroutine call.

If the application closes the presentation facility queue after passing it to the `PosOpen()` subroutine call, asynchronous event messages cannot be queued and errors are logged.

#### Attention

For OS/2, the IBM Point of Sale Subsystem for OS/2 attempts to verify that the queue handle is valid, and returns a 308 `POSERR_SYS_INVALID_QUEUE` return code if it is not valid. In OS/2, an incorrect value for this resource can sometimes cause Presentation Manager to end abnormally when performing the validation. Applications must make every effort to ensure that the queue handle that is passed in is valid.

## PosNslotNumber

<b>Type</b>	char
<b>Default</b>	PosSLOT_5
<b>Access</b>	CG

The **PosNslotNumber**, **PosNportNumber**, and **PosNdeviceNumber** resources together uniquely identify each point-of-sale device found by the IBM Point of Sale Subsystem. (See “PosOpen()” on page 18-10 for an example usage.) The **PosNslotNumber** resource can have the following values:

**PosSLOT\_0 (0)**

**PosSLOT\_1 (1)**

**PosSLOT\_2 (2)**

**PosSLOT\_3 (3)**

**PosSLOT\_4 (4)**

**PosSLOT\_5 (5)**

**PosSLOT\_6 (6)**

**PosSLOT\_7 (7)**

**PosSLOT\_8 (8)**

The **PosNslotNumber** contained in the `POSM_SYS_DEVICE_ONLINE` message for each device (see “`POSM_SYS_DEVICE_ONLINE`” on page 20-22).

---

## PosAlarm Resource Set

**Class Name** PosAlarm  
**Include** <pos/alarm.h>

A copy of the PosAlarm resource set is created every time an application opens an alarm device by the *PosOpen()* subroutine call. The resource set associated with an alarm connection when it is opened remains associated with it until the alarm connection is closed.

**Note:** The alarm device is not supported on any model of the IBM 4695.

The following table gives an overview of the resources in this set.

*Table 21-3. PosAlarm Resources*

Name	Type	Default	Access
PosNalarmStatus	int	n/a	G

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## PosNalarmStatus

**Type** int  
**Default** n/a  
**Access** G

The **PosNalarmStatus** resource identifies the status of the alarm, whether it is sounding or not, and if the alarm is currently online (connected).

This resource can have the following values:

**PosALARM\_ONLINE (0x01)**

Indicates that the device is attached and online. If this bit is not set, the device is considered offline and is possibly not even attached.

**PosALARM\_SOUNDING (0x02)**

Indicates that the alarm is sounding. If this bit is not set, the alarm is not sounding. This bit is always off if the alarm is offline.

---

## PosDisplay Resource Set

**Class Name** PosDisplay  
**Include** <pos/display.h>

A copy of the PosDisplay resource set is created every time an application opens a display device by the *PosOpen()* subroutine call. The resource set associated with a display connection when it is opened remains associated with it until the display connection is closed. Changing the resources of one opened display does not affect any of the other displays.

The following table gives an overview of the resources in this set.

*Table 21-4. PosDisplay Resources*

<b>Name</b>	<b>Type</b>	<b>Default</b>	<b>Access</b>
PosNcharSize	int	PosDSP_CHAR_2x20	CGS
PosNdisplayCodePage	int	n/a	CGS
PosNdisplayCursor	unsigned int	0 (zero)	CGS
PosNdisplayMode	int	PosDSP_MODE_NORMAL	CGS
PosNdisplayLightsOn	unsigned int	0 (zero)	CGS
PosNpixelX	int	0 (zero)	CGS
PosNpixelY	int	0 (zero)	CGS

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## PosNcharSize

<b>Type</b>	int
<b>Default</b>	PosDSP_CHAR_2x20
<b>Access</b>	CGS

The **PosNcharSize** resource defines the character mode.

This resource can have the following values:

### **PosDSP\_CHAR\_2x20 (0x00)**

Two lines of 20 characters (8x16), 10 double-byte characters, or a combination.

### **PosDSP\_CHAR\_3x32 (0x01)**

Three lines of 32 characters (5x12). Double-byte characters are not allowed in this mode. Code page definition by **PosNdisplayCodePage** will be ignored in this mode.

This resource is only for the Character and Graphics Display.

## PosNdisplayCodePage

<b>Type</b>	int
<b>Default</b>	For the newer Vacuum Fluorescent Display II (VFD II): n/a  For the USB Character/Graphics Displays, the default is determined by the operating system that is loaded and the DBCS fonts that are loaded into the display. See the following table:

Windows OS language	Loaded DBCS font in the display is:	Default PosNdisplayCodePage
Japanese	Japanese	PosDSP_CP_JAPANESE
	other (none or other DBCS)	n/a
Korean	Korean	PosDSP_CP_KOREA
	other (none or other DBCS)	n/a

Traditional Chinese	Traditional Chinese	PosDSP_CP_TRAD_CHINESE
	other (none or other DBCS)	n/a
Simplified Chinese	Simplified Chinese	PosDSP_CP_SIMP_CHINESE
	other (none or other DBCS)	n/a
other	n/a	n/a

**Access** CGS

The **PosNdisplayCodePage** resource defines the character set to be used in displaying text on newer Vacuum Fluorescent Display II (VFD II) devices and newer Character/Graphics Display devices.

This resource can have the following values:

**(0x00) PosDSP\_CP\_US\_EUROPE**

Original Vacuum Fluorescent Display character set.

**(0x01) PosDSP\_CP\_KATAKANA**

Character set containing some Katakana characters.

**(0x02) PosDSP\_CP\_MULTILINGUAL\_INTL**

Code page 850.

**(0x03) PosDSP\_CP\_HUNGARY**

Code page 852.

**(0x04) PosDSP\_CP\_CYRILLIC**

Code page 855.

**(0x05) PosDSP\_CP\_TURKEY**

Code page 857.

**(0x06) PosDSP\_CP\_ISRAEL**

Code page 862.

**(0x07) PosDSP\_CP\_CANADIAN\_FRENCH**

Code page 863.

**(0x08) PosDSP\_CP\_ARABIC**

Code page 864.

**(0x09) PosDSP\_CP\_NORDIC**

Code page 865.

**(0x0A) PosDSP\_CP\_CYRILLIC\_RUSSIA**

Code page 866.

**(0x0B) PosDSP\_CP\_GREECE**

Code page 869.

**(0x80) PosDSP\_CP\_JAPANESE**

SBCS Code page 897, DBCS Code page 301. Not resident.

**(0x90) PosDSP\_CP\_KOREA**

SBCS Code page 1088, DBCS Code page 951. Not resident.

**(0xA0) PosDSP\_CP\_TRAD\_CHINESE**

SBCS Code page 1114, DBCS Code page 950. Not resident.

**(0xB0) PosDSP\_CP\_SIMP\_CHINESE**

SBCS Code page 1115, DBCS Code page 1381. Not resident.

This resource is only supported by newer VFD II devices and newer Character/Graphics Display devices. Devices that do not support this resource will ignore it.

For newer Character/Graphics Display devices, this resource is valid in **PosDSP\_MODE\_NORMAL** for **PosNdisplayMode** and **PosDSP\_CHAR\_2x20** for **PosNcharSize**.

## PosNdisplayCursor

<b>Type</b>	unsigned int
<b>Default</b>	0
<b>Access</b>	CGS

The **PosNdisplayCursor** resource specifies the beginning position on the display to begin reading or writing. The offset values can be:

Offset Value	Display
0 – 39	Alphanumeric Display
0 – 39	Operator Display
0 – 11	Shopper Display
0–39	Character and Graphics Display (in PosDSP_CHAR_2x20 mode)
0–95	Character and Graphics Display (in PosDSP_CHAR_3x32 mode)
0–39	40-Character Liquid Crystal Display II
0–39	40-Character Vacuum Fluorescent Display II
0–39	Two-Sided Vacuum Fluorescent Display II
0–39	2x20 Character VFD Customer Display

## PosNdisplayMode

<b>Type</b>	int
<b>Default</b>	PosDSP_MODE_NORMAL
<b>Access</b>	CGS

The **PosNdisplayMode** resource defines the display mode.

This resource can have the following values:

### PosDSP\_MODE\_NORMAL (0x00)

Display characters in rows (2x20 or 3x32) or double-byte characters (2x20 only), using the **PosNdisplayCursor** resource to position the text on the display.

### PosDSP\_MODE\_LOGO (0x01)

Display bitmap data using **PosNpixelX** and **PosNpixelY** to position the bitmap on the display.

### PosDSP\_MODE\_APA (0x02)

Display characters using **PosNpixelX** and **PosNpixelY** to position the characters on the display. Double-byte characters are not allowed in this mode. Code page definition by **PosNdisplayCodePage** will be ignored in this mode.

This resource is only for the Character and Graphics Display.

**PosNdisplayLightsOn**

<b>Type</b>	unsigned long
<b>Default</b>	0
<b>Access</b>	CGS

The **PosNdisplayLightsOn** resource identifies which guidance lights are to be on. Any light not requested to be switched on is switched off.

The Shopper Display has two columns, each containing 3 indicator lights. The indicator lights have values:

Top left	0x20
Middle left	0x10
Bottom left	0x08
Top right	0x04
Middle right	0x02
Bottom right	0x01

Specify the value 0x3F to illuminate all the indicator lights on the Shopper Display.

The Character and Graphics Display has two rows of 12 indicator lights. The first row of indicator lights is above the main screen area. The second row is below the main screen area. The indicator lights have the following values:

First Row: [0x80000000] [0x40000000] . . . [0x00200000] [0x00100000]

Second Row: [0x00008000] [0x00004000] . . . [0x00000020] [0x00000010]

Specify the value 0xFFFF0FFF0 to illuminate all the indicator lights on the Character and Graphics Display.

The 40-Character Vacuum Fluorescent Display II, the 2x20 Character VFD Customer Display, and the Two-Sided Vacuum Fluorescent Display II have one row of 20 indicator lights below the main screen area. The indicator lights have the following values:

Light 20: 0x02000000	Light 10: 0x00000200
Light 19: 0x01000000	Light 9: 0x00000100
Light 18: 0x00800000	Light 8: 0x00000080
Light 17: 0x00400000	Light 7: 0x00000040
Light 16: 0x00200000	Light 6: 0x00000020
Light 15: 0x00100000	Light 5: 0x00000010
Light 14: 0x00080000	Light 4: 0x00000008
Light 13: 0x00040000	Light 3: 0x00000004
Light 12: 0x00020000	Light 2: 0x00000002
Light 11: 0x00010000	Light 1: 0x00000001

**Note:** Light 20 is the left-most light and Light 1 is the right-most light.

Specify the value 0x03FF03FF to illuminate all the indicator lights on the 40-Character Vacuum Fluorescent Display II, and the Two-Sided Vacuum Fluorescent Display II.

You can set the guidance lights by specifying the value for resource **PosNdisplayLightsOn** in the resource file or by sending it on the POS\_SYS\_SET\_VALUES *PosIOctl()* request.

This resource is supported by the following displays:

- Shopper Display

- Character and Graphics Display
- 2x20 Character VFD Customer Display
- 40-Character Vacuum Fluorescent Display II
- Two-Sided Vacuum Fluorescent Display II

## PosNpixelX

**Type** int  
**Default** 0  
**Access** CGS

The **PosNpixelX** resource defines a pixel cursor horizontal location. The left column of pixels of the Character and Graphics Display is pixel location 0 (zero).

## PosNpixelY

**Type** int  
**Default** 0  
**Access** CGS

The **PosNpixelY** resource defines a pixel cursor vertical location. The top row of pixels of the Character and Graphics Display is pixel location 0.

## PosDrawer Resource Set

**Class Name** PosDrawer  
**Include** <pos/drawer.h>

A copy of the PosDrawer resource set is created every time an application opens a cash drawer device by the *PosOpen()* subroutine call. The resource set associated with a cash drawer connection when it is opened remains associated with it until the cash drawer connection is closed. Changing the resources of one opened cash drawer does not affect any of the other cash drawers.

The following table gives an overview of the resources in this set.

*Table 21-5. PosDrawer Resources*

<b>Name</b>	<b>Type</b>	<b>Default</b>	<b>Access</b>
PosNpulseWidth	int	80	CGS
PosNtillStatus	int	n/a	G

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## PosNpulseWidth

**Type** int  
**Default** 80  
**Access** CGS

The **PosNpulseWidth** resource specifies the duration of the pulse that is required to open the cash drawer till. The value is specified in terms of milliseconds, and can range from 5 ms to 1048 ms.

## PosNtillStatus

<b>Type</b>	int
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNtillStatus** resource identifies the status of the cash drawer, whether it is open or closed, and whether it is currently online (connected). This resource can have the following values:

### PosDRAWER\_ONLINE (0x01)

Indicates that the device is attached and online. If this bit is not set, the device is considered offline and might not be attached.

### PosDRAWER\_OPEN (0x04)

Indicates that the till drawer is open. If this bit is not set, the till drawer is closed. This bit is off if the device is offline.

---

## PosKeyboard Resource Set

<b>Class Name</b>	PosKeyboard
<b>Include</b>	<pos/keyboard.h>

A copy of the PosKeyboard resource set is created every time an application opens a keyboard device by the *PosOpen()* subroutine call. The resource set associated with a keyboard connection when it is opened remains associated with it until the keyboard connection is closed. Changing the resources of one opened keyboard does not affect any of the other keyboards.

The following table gives an overview of the resources in this set.

Table 21-6. PosKeyboard Resources

Name	Type	Default	Access
PosNdoubleKey01 - PosNdoubleKey60	char *	0,0	C
PosNfatFingerTimeOut	int	30	CG
PosNkeyboardClick	int	PosOFF	CGS
PosNkeyLock	int	n/a	G
PosNkeyboardLightsOn	int	PosLIGHTS_OFF	CGS
PosNnumpadLocation	int	PosLOCATION_1	CG
PosNnumpadStyle	int	PosCALCULATOR_POS	CG
PosNnumpadZero	int	PosDOUBLE_KEY	CG
PosNtoneDuration	int	2	CGS
PosNtoneFreq	int	PosMEDIUM	CGS
PosNtoneVolume	int	PosLOW	CGS
PosNtypematicDelay	int	250	CG
PosNtypematicFreq	int	1	CG

See "Resource Access Codes" on page 21-5 for descriptions of the access codes.

## PosNdoubleKey01 - PosNdoubleKey60

<b>Type</b>	char *
<b>Default</b>	0,0
<b>Access</b>	C

The double key resources specify the double key pairs for the keyboard. A double key is defined as two keys that are tied together with a 1x2 key top. When a double key is specified as resource value, it is given as a pair of decimal key switch positions. You can define up to 60 double key pairs for the keyboard. If 0,0 is used for the double key pair, no double key is defined. For example, one definition of double keys for the ANPOS Keyboard in a resource file could look like this:

```
! A sample resource definition for an ANPOS keyboard
*anpos.doubleKey01: 117,118
*anpos.doubleKey02: 119,120
*anpos.doubleKey03: 121,122
*anpos.doubleKey04: 0,0
```

```
! Double key pairs 05 through 60 would be undefined (0,0) by default.
! The definition of doubleKey04 above would not be necessary.
```

The key switch positions are shown in individual keyboard layout figures in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book.

The IBM Point of Sale Subsystem searches the resource table when the system is initialized, looking for double key definitions. It uses these definitions to build a double key table. The double key table is searched on each key scan code in order to detect any double keys defined by the application. When the IBM Point of Sale Subsystem sees the scan codes from a defined double key, it returns the preferred scan code for that double key. The other scan code, if seen, is discarded. The preferred scan code is the one associated with the first key of the double key pair.

Double keys should usually be defined in the resource file instead of being defined by the application on the *PosOpen()*. This ensures that all applications sharing the resource file will have the same double key definitions. There might be instances where you want your application to define double keys when the keyboard is opened. If you do this, recognize the possible conflict between what your application defines as double keys and what another application defines as double keys. Following is an example of how to define double keys when opening the keyboard.

```
#include <pos/pos.h>

int kbddes;
int rc = 0;
PosArg resource; /* doubleKey01 resource */

rc = PosInitialize( "myappl", "checkout", 0, 0, 0, 0 );

PosSetArg( resource, PosNdoubleKey(01), ((char *) "77,82" ) );

kbddes = PosOpen( "mykbd", PosKeyboard, &resource, 1 );
```

The following list describes the key switches that can be doubled for each keyboard. The keyboard device handler does not cross-check that double key definitions define adjacent key switch locations. The IBM Point of Sale Subsystem allows both horizontal and vertical double key tops.

#### All checkout keyboards

All keys except the numeric keypad keys, the **S1** key, the **S2** key, and the **Ctrl** key.

#### Modifiable Layout Keyboard with Card Reader

All keys except the **S1** key, the **S2** key, and the **Ctrl** key. Because the numeric keypad position is moveable, no check is made to determine if the double key switch is part of the numeric keypad.

#### ANPOS Keyboard

All key switches in the range of 75 to 128, except the numeric keypad keys, the **S1** key, the **S2** key, and the **Esc** key.

#### Retail Alphanumeric Point of Sale Keyboard with Card Reader

All key switches in the range of 75 to 128, except the **S1** key, the **S2** key, and the **Esc** key. Because the numeric keypad position is moveable, no check is made to determine if the double key switch is part of the numeric keypad.

#### All DBCS Point of Sale Keyboards

Double-byte character set (DBCS) double keys are set when the keyboard is manufactured. They cannot be changed by the application program.

The DBCS point-of-sale keyboards ignore this resource.

**Note:** Double keys operate as those in the ANPOS keyboard when the PC Point of Sale Keyboard functions as a system keyboard.

**Note:** Double keys can only be defined using the ANPOS utility for system keyboards. See "Configuring the Alphanumeric Point of Sale Keyboard" on page 3-4 for more information on the ANPOS utility.

This resource is not supported on the following keyboards:

- PC Point of Sale Keyboard (when functioning as a non-system keyboard)
- PLU Keyboard/Display-III
- Point of Sale Keyboard V
- Point of Sale Keyboard VI
- 4685 Keyboard Model K01

## PosNfatFingerTimeOut

<b>Type</b>	int
<b>Default</b>	30
<b>Access</b>	CG

The **PosNfatFingerTimeOut** resource specifies the time between keystrokes used to determine if a fat-finger condition has occurred. The value can be:

<b>0</b>	0 milliseconds
<b>10</b>	10 milliseconds
<b>20</b>	20 milliseconds
<b>30</b>	30 milliseconds
<b>40</b>	40 milliseconds

**Notes:**

1. The term *fat-finger* refers to two keys being pressed faster than the value specified using the PosNfatFingerTimeOut resource. This could occur because:
  - The operator pressed two keys at once.
  - The operator is keying faster than 25 keys per second.
  - A double key was not defined to the keyboard device handler.
2. This resource is ignored if it is specified for the system keyboard.
3. This resource is ignored if it is specified for the 50-Key Modifiable Layout Keyboard or the 50-Key Modifiable Layout Keyboard and Operator Display. The fat-finger timeout for these keyboards is 40 milliseconds, and cannot be changed.

**PosNkeyboardClick**

<b>Type</b>	int
<b>Default</b>	PosOFF
<b>Access</b>	CGS

The **PosNkeyboardClick** resource specifies whether the keyboard device handler is to initialize the keyboard so that it produces an audible click when keys are pressed. The valid values are:

**PosOFF (0x00)**

Disable keyboard click

**PosSOFT (0x01)**

Enable soft keyboard click

**PosLOUD (0x02)**

Enable loud keyboard click

This resource is ignored if it is specified for the 50-Key Modifiable Layout Keyboard, or the 50-Key Modifiable Layout Keyboard and Operator Display. It is supported by all other keyboards.

**PosNkeyLock**

<b>Type</b>	int
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNkeyLock** resource value can only be retrieved. The value of the resource indicates the current position of the keyboard lock.

This resource can have the following values:

**PosPOSITION\_UNKNOWN (0)**

The key position is unknown because the keyboard is offline.

**PosPOSITION\_MANAGER (0x01)**

The key lock is in the manager position.

**PosPOSITION\_OPERATOR (0x02)**

The key lock is in the operator position.

**PosPOSITION\_INACTIVE (0x03)**

The key lock for the DBCS point-of-sale keyboard is in the inactive position.

**PosPOSITION\_SYSTEM (0x04)**

The key lock for the DBCS point-of-sale keyboard is in the system position.

**PosNkeyboardLightsOn**

<b>Type</b>	int
<b>Default</b>	PosLIGHTS_OFF
<b>Access</b>	CGS

The **PosNkeyboardLightsOn** resource identifies which point-of-sale specific lights on the keyboard are to be switched on. This resource can have the following values:

**PosLIGHTS\_OFF (0x00)**

This is the default value that indicates that all the lights are off.

**PosLIGHTS\_WAIT (0x01)**

The left-most light on non-DBCS point-of-sale keyboards. On DBCS point-of-sale keyboards, this light means Wait.

**PosLIGHTS\_OFFLINE (0x02)**

The left-middle light on non-DBCS point-of-sale keyboards. On DBCS point-of-sale keyboards, this light means Offline.

**PosLIGHTS\_MESSAGE\_PENDING (0x04)**

The right-middle light on non-DBCS point-of-sale keyboards. On DBCS point-of-sale keyboards, this light means Message Pending.

**PosLIGHTS\_NO\_LABEL (0x08)**

The right-most light on non-DBCS point-of-sale keyboards. This is the unlabeled point-of-sale light.

**PosLIGHTS\_READY (0x08)**

This light is on DBCS point-of-sale keyboards only. This light means READY.

This resource is supported for all keyboards.

**PosNnumpadLocation**

<b>Type</b>	int
<b>Default</b>	PosLOCATION_1
<b>Access</b>	CG

The **PosNnumpadLocation** resource specifies the numeric keypad location for those keyboards that allow the keypad to be moved. The possible locations for the keypad are illustrated in the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book. The values can be:

<b>PosLOCATION_1 (0x01)</b>	Numeric keypad is at the right-most location.
<b>PosLOCATION_2 (0x02)</b>	Numeric keypad is second from the right.
<b>PosLOCATION_3 (0x03)</b>	Numeric keypad is third from the right.

This resource is supported only for the Modifiable Layout Keyboard with Card Reader and the Retail Alphanumeric Point of Sale Keyboard with Card Reader, and only when they are point-of-sale keyboards. All other keyboards ignore this request.

## PosNnumpadStyle

<b>Type</b>	int
<b>Default</b>	PosCALCULATOR_POS
<b>Access</b>	CG

The **PosNnumpadStyle** resource specifies the format you want for your numeric keypad. The values can be:

### PosTOUCHTONE\_POS (0x00)

Indicates a touchtone keypad, with the bottom right key defined as a slash (/). Places the key for numeric **1** in the upper left corner.

### PosCALCULATOR\_POS (0x01)

Indicates a calculator keypad, with the bottom right key defined as a slash (/). Places the key for numeric **1** in the lower left corner.

### PosTOUCHTONE\_STANDARD (0x02)

Indicates a touchtone keypad, with the bottom right key defined as the standard character for the country-specific keyboard. For the U.S. keyboard, this character is a period (.). Places the key for numeric **1** in the upper left corner.

### PosCALCULATOR\_STANDARD (0x03)

Indicates a calculator keypad, with the bottom right key defined as the standard character for the country-specific keyboard. For the U.S. keyboard, this character is a period (.). Places the key for numeric **1** in the lower left corner.

#### Notes:

1. The checkout keyboards only support the PosTOUCHTONE\_POS and PosCALCULATOR\_POS values.
2. This resource is only supported for point-of-sale keyboards. The operating system controls the layout of the system keyboard. For OS/2, it is possible to change the numeric pad layout by editing the OS/2 system file KEYBOARD.DCP with a hexadecimal editor. To do this you must:
  - a. **Important:**Make a backup copy of the KEYBOARD.DCP file..
  - b. Locate the correct translation table for the country, subcountry, keyboard type and code page being used. For more information about the table format, see the *IBM OS/2 2.0 Technical Library: Physical Device Driver Reference*.
  - c. Change the entries to the appropriate values for the scan codes that produce the 1, 2, 3, 7, 8, and 9 keys.
  - d. Re-IPL the system.

## PosNnumpadZero

<b>Type</b>	int
<b>Default</b>	PosDOUBLE_KEY
<b>Access</b>	CG

The **PosNnumpadZero** resource specifies whether the numeric keypad is a double key (as shipped), or has been converted to two single keys. The values can be:

### PosSINGLE\_KEY (0x01)

Define the numeric keypad 0 (zero) as a single key.

**PosDOUBLE\_KEY (0x02)**

Define the numeric keypad 0 (zero) as a double key.

When the 0 (zero) key is set as a double key, the primary key produces the scan code defined for it. The other key switch is inactive. When the 0 (zero) key is set as a single key, both key switches produce the scan code defined for the key. See the *IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages* book for the scan codes for these keys.

**Note:** Numeric keypad zero can only be defined using the ANPOS utility for system keyboards. See “Configuring the Alphanumeric Point of Sale Keyboard” on page 3-4 for more information on the ANPOS utility.

This resource is ignored if it is specified for the 50-Key Modifiable Layout Keyboard or the 50-Key Modifiable Layout Keyboard and Operator Display. It is supported by all other keyboards.

**PosNtoneDuration**

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CGS

The **PosNtoneDuration** resource controls the duration of the internal speaker. The values can be:

<b>PosON (0x01)</b>	On until switched off
<b>2-255</b>	0.2 - 25.5 seconds (0.10 times the specified number)

This resource is supported for all keyboards.

**PosNtoneFreq**

<b>Type</b>	int
<b>Default</b>	PosMEDIUM
<b>Access</b>	CGS

The **PosNtoneFreq** resource controls the frequency of the internal speaker. The frequency values can be:

<b>PosLOW (0x01)</b>	875 Hz
<b>PosMEDIUM (0x02)</b>	1.3 KHz
<b>PosHIGH (0x03)</b>	2.0 KHz

This resource is supported for all keyboards.

**PosNtoneVolume**

<b>Type</b>	int
<b>Default</b>	PosLOW
<b>Access</b>	CGS

The **PosNtoneVolume** resource controls the volume of the internal speaker. The values can be:

<b>PosLOW (0x01)</b>	Tone volume soft
----------------------	------------------

**PosHIGH (0x03)**                      Tone volume loud

This resource is supported for all keyboards.

## PosNtypematicDelay

**Type**                      int  
**Default**                  250  
**Access**                  CG

The **PosNtypematicDelay** resource indicates the delay between the time when the key is first pressed and when the key starts repeating automatically. The valid typematic delays are:

**250**    250 milliseconds  
**500**    500 milliseconds  
**750**    750 milliseconds  
**1000**  1000 milliseconds

**Notes:**

1. For OS/2, the typematic delay can only be changed by using the OS/2 System Setup for system attached keyboards. See the OS/2 documentation for more information on System Setup.
2. For the Microsoft Windows operating system, the typematic delay for a system keyboard can be changed by using the Microsoft Windows *Keyboard* applet from the Control Panel folder.

This resource is ignored if it is specified for the 50-Key Modifiable Layout Keyboard or the 50-Key Modifiable Layout Keyboard and Operator Display. It is supported for all other keyboards.

## PosNtypematicFreq

**Type**                      int  
**Default**                  1  
**Access**                  CG

The **PosNtypematicFreq** resource indicates the rate at which keys repeat automatically. Valid numbers are from 0 (zero) to 31. The formula for the typematic rates is:

$$\text{Rate} = \frac{1}{(8 + A) \cdot (2^{**}B) \cdot 0.00417}$$

A = bits 2–0 of **PosNtypematicFreq**

B = bits 4–3 of **PosNtypematicFreq**

Some of the more common typematic rates are shown in the list below.

Frequency	Typematic Rate (characters per second)
0	30
1	26.7
4	20.0
8	15.0
12	10.0

## PosKeyboard Resource Set

created on October 2, 2001

20	5.0
26	3.0
31	2.0

### Notes:

1. This resource is ignored if it is specified for the 50-Key Modifiable Layout Keyboard or the 50-Key Modifiable Layout Keyboard/Operator Display. It is supported for all other keyboards.
2. The values indicated for **PosNtypematicFreq** and **PosNtypematicDelay** apply to all keys and are set according to the values in the last rate and delay command received.
3. For OS/2, the typematic frequency (rate) can only be changed using the OS/2 System Setup for system attached keyboards. See OS/2 documentation for more information on System Setup.
4. For the Microsoft Windows operating system, the typematic frequency for a system keyboard can be changed by using the Microsoft Windows *Keyboard* applet from the Control Panel folder.

---

## PosMsr Resource Set

<b>Class Name</b>	PosMsr
<b>Include</b>	<pos/msr.h>

The PosMsr resource set does not contain any MSR specific resources. However, the class name is still used on the *PosOpen()* subroutine call to identify that a MSR is being opened.

---

## PosNvram Resource Set

<b>Class Name</b>	PosNvram
<b>Include</b>	<pos/nvram.h>

A copy of the PosNvram resource set is created every time an application opens an NVRAM device using the *PosOpen()* subroutine call. The resource set associated with an NVRAM connection when it is opened remains associated with it until the NVRAM connection is closed. Changing the resources of one opened NVRAM does not affect any of the other NVRAMs. The following table gives an overview of the resources in this set.

Table 21-7. PosNvram Resources

Name	Type	Default	Access
PosNnvramCursor	long	0	CGS
PosNnvramMode	int	PosMODE_DIRECT	CG
PosNnvramSize	long	n/a	G

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## PosNnvramCursor

<b>Type</b>	long
<b>Default</b>	0
<b>Access</b>	CGS

The **PosNvramCursor** resource specifies the position of the next read or write for this NVRAM connection. This resource can have values from 0 (zero) through the limit defined in the “list of NVRAM application address space” on page 11-2.

### PosNvramMode

**Type** int  
**Default** PosMODE\_DIRECT  
**Access** CG

The **PosNvramMode** resource specifies the operational mode for this NVRAM connection. This resource can have the following values:

**PosMODE\_DIRECT (0x00)**

Reads and writes to the NVRAM device are in direct mode.

**PosMODE\_SEQUENTIAL (0x01)**

Reads and writes to the NVRAM device are in sequential mode.

### PosNvramSize

**Type** long  
**Default** n/a  
**Access** G

The **PosNvramSize** resource specifies the size of the usable portion of the NVRAM device. This value depends on the type of hardware used, and cannot be modified by an application program. The possible values are defined in the “list of NVRAM application address space” on page 11-2.

## PosPower Resource Set

**Class Name** PosPower  
**Include** <pos/power.h>

A copy of the PosPower resource set is created every time an application opens a programmable power device using the *PosOpen()* subroutine call. The resource set associated with a programmable power connection when it is opened remains associated with it until the programmable power connection is closed. Changing the resources of one opened programmable power device does not affect any of the other programmable power devices.

The following table gives an overview of the resources in this set.

*Table 21-8. PosPower Resources*

<b>Name</b>	<b>Type</b>	<b>Default</b>	<b>Access</b>
PosNpowerAlarm	long	0	G

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

### PosNpowerAlarm

**Type** long

**Default**        0  
**Access**         G

The **PosNpowerAlarm** resource specifies the day of month and the time of day when power is to be turned on at a terminal. This resource cannot be set using the `POS_SYS_SET_VALUES Chapter 19. PosIOctl() Requests` subroutine. It can only be modified by calling the `POS_POWER_SET_ALARM Chapter 19. PosIOctl() Requests` subroutine. The resource is given as a packed BCD format value in the form **ddhhmm00**:

- dd**     A 2-digit integer from 00 to 31, representing the day of the month. If 00 (two zeros) is used to specify the day of month, the alarm setting is cleared, allowing an application to turn off power without enabling a time to turn power back on. The hour and minutes values are ignored if 00 is specified for the day of month. If the value is from 01 to 31, either of the following conditions must be true for the value to be valid:
  - If the day and time are prior to the current day and time, the day must be valid for the next calendar month.
  - If the day and time are after the current day and time, the day must fall within the current month.
- hhmm** A 4-digit integer representing the time in international format. For example, 5:30 p.m. is 1730 and 12:00 a.m. is 0000. The first 2 digits, which represent the hour, must be from 00 to 23, and the last 2 digits, which represent minutes, must be from 00 to 59.

For example, a resource value of 0x25184500 means that the alarm would be set for the 25th day of the month at 6:45 p.m.

## PosPrinter Resource Set

**Class Name**    PosPrinter  
**Include**        <pos/printer.h>

A copy of the PosPrinter resource set is created every time an application opens a printer device by using the `PosOpen()` subroutine call. The resource set associated with a printer connection when it is opened remains associated with it until the printer connection is closed. Changing the resources of one opened printer does not affect any of the other printers.

The following table gives an overview of the resources in this set.

Table 21-9. PosPrinter Resources

Name	Type	Default	Access
PosNcodePage	int	Code page in the operating system when it is loaded	GS
PosNCRWidth	short	PosCR_WIDE	G
PosNdiOrientation	short	PosPRINT_PORTRAIT	CGS
PosNDIWidth	short	PosDI_WIDE	CGS
PosNfeedDirection	short	PosFEED_FORWARD	CGS
PosNfiscalCountry	short	n/a	G
PosNfiscalNotify	short	PosFISCAL_NOTIFY_OFF	CGS

Table 21-9. PosPrinter Resources (continued)

Name	Type	Default	Access
PosNfiscalPLDStatus	short	n/a	G
PosNfiscalVersion	short	n/a	G
PosNheadParkedPosition	short	PosHOME_CENTER	CGS
PosNinterleaved	short	PosINTERLEAVED_FALSE	CGS
PosNleftMarginCR	long	PosLEFT_MARGIN_MODEL4	CGS
PosNlineFeedCR	long	PosLPI_6	CGS
PosNlineFeedDI	long	PosLPI_6	CGS
PosNlineFeedSJ	long	PosLPI_8	CGS
PosNprintAlignment	short	PosPRINT_ALIGN_LEFT	CGS
PosNprintColorMode	short	PosPAPER_NORMAL	CGS
PosNprintCRCharSetx	short	PosPRINT_NONE	G
PosNprintDICharSetx	short	PosPRINT_NONE	G
PosNprintFeatures	long	n/a	G
PosNprintMode	short	PosMODE_NORMAL	CGS
PosNprintQualityMode	short	PosHIGH_QUALITY_OFF	CGS
PosNprintStation	short	PosPRINT_STATION_CR	CGS
PosNprintStatus	long	n/a	G
PosNprintStatus2	long	n/a	G
PosNprintTabStops	unsigned char *	100,200,300,400,500	CGS
PosNprintToneDuration	short	0x03	CGS
PosNprintToneFrequency	int	2093	GS
PosNprintToneNote	int	PosPRINT_TONE_B	CGS
PosNprintToneOctave	short	PosPRINT_OCTAVE4	CGS
PosNprintToneVolume	short	PosPRINT_TONE_SOFT	CGS
PosNprintUpsideDown	short	PosPRINT_RIGHT_SIDE_UP	CGS
PosNrawPrintStatus	unsigned char *	n/a	G
PosNresumeString	unsigned char *	86 blank characters	S
PosNretryString	unsigned char *	86 single-quote characters	S

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## PosNcodePage

<b>Type</b>	int
<b>Default</b>	Code page in the operating system when it is loaded
<b>Access</b>	GS

The **PosNcodePage** resource is the code page used for printing. On Windows systems, the initial value of this resource is acquired from the operation system when the IBM Point of Sale Subsystem is loaded.

**Notes:**

1. This resource is not supported on the 4689 family of printers.
2. This resource is not support on the USB fiscal printer and fiscal printers that are set to fiscal mode (with *Pos/Oct()* POS\_PRN\_ENABLE\_FISCAL\_PRINTING).

An invalid value defaults to using codepage 858.

**Code Pages Supported:**

Value Entered	Code Page
437	CP00437
819	CP00819
850	CP00850
852	CP00852
855	CP00855
857	CP00857
858	CP00858 (default if no code page set from operating system.)
860	CP00860
861	CP00861
862	CP00862
863	CP00863
864	CP00864
865	CP00865
866	CP00866
869	CP00869
1116	CP01116
1117	CP01117
1118	CP01118
1119	CP01119
1250	CP01250
1251	CP01251
1252	CP01252
1253	CP01253
1254	CP01254
1257	CP01257

**DBCS Code Pages :** The double byte code pages should not be changed because they are downloaded to the printer (4610 TI5) or fixed in the printer.

- 932** Japan downloads available for 4610 TI5:
- Gothic Style
  - Mincho Style
    - Single-bytes characters - code page 897
    - Double-byte characters - code page 301
- 949** Korea downloads available for 4610 TI5:
- Single-byte characters - code page 1088
  - Double-byte characters - code page 951
- 950** Traditional Chinese:
- (BIG-5) with Bold SBCS impact
  - (BIG-5) with Normal SBCS impact

1381 Simplified Chinese (PRC) download for 4610 TI5

## PosNCRWidth

**Type** short  
**Default** PosCR\_WIDE  
**Access** G

The **PosNCRWidth** resource informs the applications about the size of the paper that is loaded (58mm or 80mm paper). The supported values for this resource are:

**PosCR\_WIDE (0x00)**  
 Set printer for 80mm paper

**PosCR\_NARROW (0x01)**  
 Set printer for 58mm paper

**Note:** Paper width settings are also stored in “PosNprintFeatures” on page 21-43  
**PosFEATURE\_58MM\_PAPER**

This resource is supported on Single Station SureMark printers.

## PosNdiOrientation

**Type** short  
**Default** PosPRINT\_PORTRAIT  
**Access** CGS

The **PosNdiOrientation** resource controls the orientation of printing in the DI station. This resources can be set to one of the following:

**PosPRINT\_PORTRAIT (0x0000)**  
 Characters are printed upright and the form should be inserted with the right side of the form inserted against the right paper stop of the DI station.

**PosPRINT\_LANDSCAPE (0x0001)**  
 Characters are printed 90 degrees counter-clockwise. The bottom of the form should be along the right paper stop of the DI station. Lines are printed from the bottom of the form to the top of the form (a line feed will move the print head to the left and not advance the document).

Double-high and emphasized characters are not supported.

When the impact characters are defined as 16x16, landscape printing is not available.

**Note:** This resource is supported only on the IBM 4610 SureMark Point of Sale printers that have a DI station.

## PosNDIWidth

**Type** short  
**Default** PosDI\_WIDE  
**Access** CGS

The **PosNDIWidth** resource controls the placement and width of printing on the DI station.

This resource can have the following values:

**PosDI\_WIDE (0x00)**

The print line is from the left margin to the right margin, and is 86 (47 for the IBM 4610 printers) characters wide. If a value other than PosDI\_WIDE or PosDI\_NARROW is specified, an error code of 4901 POSERR\_PRN\_INVALID\_DI\_WIDTH is returned.

**PosDI\_NARROW (0x01)**

The print line is from the center to the right margin, and is 38 characters wide.

**Note:** This resource is ignored on the IBM Model 2 printer and the IBM 4689 printers.

This resource is supported only by the following printers:

- IBM Model 3 printer
- IBM Model 3F printer
- IBM Model 3R printer
- IBM Model 4 printer
- IBM Model 4A printer
- IBM Model 4R printer
- IBM 4610 Model T11 printer
- IBM 4610 Model T12 printer
- IBM 4610 Model T13 printer
- IBM 4610 Model T14 printer
- IBM 4610 Model T15 printer

## PosNfeedDirection

<b>Type</b>	short
<b>Default</b>	PosFEED_FORWARD
<b>Access</b>	CGS

The **PosNfeedDirection** resource controls the feed direction of the DI station. If a value other than PosFEED\_FORWARD or PosFEED\_REVERSE is specified, an error code of 4904 POSERR\_PRN\_INVALID\_STATION is returned.

**Note:** This resource is ignored on the IBM Model 2 printer, the IBM 4689 printers, and on the IBM 4610 SureMark Point of Sale printers.

This resource can have the following values:

**PosFEED\_FORWARD (0x00)**

Feed document from front to top of DI station.

**PosFEED\_REVERSE (0x01)**

Feed document from top to front of DI station.

This resource is supported only by the following printers:

- IBM Model 3 printer
- IBM Model 3F printer
- IBM Model 3R printer
- IBM Model 4 printer
- IBM Model 4A printer

- IBM Model 4R printer

## PosNfiscalCountry

<b>Type</b>	short
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNfiscalCountry** resource holds the current supported country of the fiscal printer.

This resource can only be queried by the POS\_SYS\_GET\_VALUES *PosIOctl()* request.

This resource is supported only by the IBM fiscal printers.

## PosNfiscalNotify

<b>Type</b>	short
<b>Default</b>	PosFISCAL_NOTIFY_OFF
<b>Access</b>	CGS

The **PosNfiscalNotify** resource controls the notification to the application of the successful completion of a fiscal command.

This resource can have the following values:

### **PosFISCAL\_NOTIFY\_OFF (0x00)**

The fiscal device handler does not notify the application upon successfully completing a fiscal command.

### **PosFISCAL\_NOTIFY\_ON (0x01)**

The fiscal device handler does notify the application upon successfully completing a fiscal command.

If this resource is set to PosFISCAL\_NOTIFY\_ON, the fiscal device handler sends a POSM\_PRN\_FISCAL\_STATUS event message to the application when a fiscal command has completed successfully. If a fiscal command completes with an error, the fiscal device handler sends a POSM\_PRN\_FISCAL\_ERROR message automatically.

If a value other than PosFISCAL\_NOTIFY\_OFF or PosFISCAL\_NOTIFY\_ON is specified, an error code of 4910 POSERR\_PRN\_INVALID\_FISCAL\_NOTIFY is returned.

This resource is supported only by the IBM fiscal printers.

## PosNfiscalPLDStatus

<b>Type</b>	short
<b>Default</b>	PosFISCAL_PLD_FALSE
<b>Access</b>	G

The **PosNfiscalPLDStatus** resource holds the power line disturbance (PLD) information of the last IPL of the fiscal printer.

This resource can have the following values:

**PosFISCAL\_PLD\_FALSE (0x00)**

The fiscal printer was not interrupted before completing a fiscal command.

**PosFISCAL\_PLD\_TRUE (0x01)**

The fiscal printer was interrupted before completing a fiscal command.

This resource can only be queried by the `POS_SYS_GET_VALUES PosIOctl()` request.

This resource is supported only by the IBM fiscal printers.

## PosNfiscalVersion

<b>Type</b>	short
<b>Default</b>	n/a
<b>Access</b>	G

This resource can only be queried by the `POS_SYS_GET_VALUES PosIOctl()` request.

The **PosNfiscalVersion** resource holds the current version of the fiscal printer.

This resource is supported only by the IBM Model 3F fiscal printers.

## PosNheadParkedPosition

<b>Type</b>	short
<b>Default</b>	PosHOME_CENTER
<b>Access</b>	CGS

The **PosNheadParkedPosition** resource controls the location where the printer head is when the printer is in the idle state.

This resource can have the following values:

**PosHOME\_CENTER (0x00)**

The head is parked at the center, between the CR and the SJ stations.

**PosHOME\_LEFT (0x01)**

The head is parked at the left margin.

If a value other than `PosHOME_CENTER` or `PosHOME_LEFT` is specified, an error code of 4903 `POSERR_PRN_INVALID_HEAD_PARKED_POSITION` is returned.

**Note:** This resource is ignored on the IBM Model 2 printer, the IBM 4689 printers, and on the IBM 4610 SureMark Point of Sale printers.

This resource is supported only by the following printers:

- IBM Model 3 printer
- IBM Model 3F printer
- IBM Model 3R printer

- IBM Model 4 printer
- IBM Model 4A printer
- IBM Model 4R printer

## PosNinterleaved

<b>Type</b>	short
<b>Default</b>	PosINTERLEAVED_FALSE
<b>Access</b>	CGS

The **PosNinterleaved** resource controls the sequence of printing on the CR and the SJ stations.

This resource can have the following values:

### **PosINTERLEAVED\_FALSE (0x00)**

Each station is printed in the order that gives the best throughput, independent of the order received.

### **PosINTERLEAVED\_TRUE (0x01)**

Each station is printed in the order received.

The default value is PosINTERLEAVED\_FALSE which allows the printer device handler to optimize the printing to get the greatest throughput. If a value other than PosINTERLEAVED\_FALSE or PosINTERLEAVED\_TRUE is specified, an error code of 4902 POSERR\_PRN\_INVALID\_INTERLEAVED\_VALUE is returned.

**Note:** This resource is ignored on the fiscal printer, the IBM 4689 printers, and on the IBM 4610 SureMark Point of Sale printers.

This resource is supported only by the following printers:

- IBM Model 2 printer
- IBM Model 3 printer
- IBM Model 3R printer
- IBM Model 4 printer
- IBM Model 4A printer
- IBM Model 4R printer

## PosNleftMarginCR

<b>Type</b>	long
<b>Default</b>	PosLEFT_MARGIN_MODEL4
<b>Access</b>	CGS

The **PosNleftMarginCR** resource controls the left margin of the print lines on the CR station. If the most significant bit of this value is set, this resource value defines the margin in microns that can range from 0 to 70,000. The following are can also be used when setting this resource:

### **PosLEFT\_MARGIN\_NONE (0x00000000)**

Sets the left margin to zero to maximize the length of the print line. This enables compatible line lengths with the IBM Model 3 and 4 printers.

### **PosLEFT\_MARGIN\_MODEL4 (0x80000000)**

Sets the left margin for compatibility with the IBM Model 3 and Model 4 printers. When set to this value, 38 characters per line at 15 characters per

inch (30 characters per line at 12 characters per inch), and the text is centered on the print line. Text lines longer than the maximum allowable on the IBM Model 3 and Model 4 printers can be printed. When set to 15 characters per inch, a maximum of 41 characters per line can be printed. When set to 12 characters per inch, a maximum of 31 characters per line can be printed. However, any lines longer than this will wrap to the next line.

**Note:** This resource is supported only on the IBM 4610 SureMark Point of Sale printers.

## PosNlineFeedCR

<b>Type</b>	long
<b>Default</b>	PosLPI_6 (4236 microns or 1/6 inch)
<b>Access</b>	CGS

The **PosNlineFeedCR** resource controls the approximate distance that the paper is fed when a line feed character is encountered in a CR station normal mode message.

The minimum line feed distance is 0 (zero) and the maximum is 90,000 microns (30,000 for the IBM 4610 SureMark Point of Sale printers).

For all printers except the IBM 4610 SureMark Point of Sale printers, there are 72 dot rows in 1 inch, and each print line has 9 dot rows. A dot row is 353 microns in length. To calculate the number of microns for a given lines per inch value, use the following formula, where *lpi* is the number of lines per inch:

$$\text{microns} = (72 / lpi) \cdot 353$$

**Note:** This resource is ignored on the IBM 4689 printers.

This resource is supported only by the following printers:

- IBM Model 2 printer
- IBM Model 3 printer
- IBM Model 3F printer
- IBM Model 3R printer
- IBM Model 4 printer
- IBM Model 4A printer
- IBM Model 4R printer
- IBM 4610 SureMark Point of Sale printer Model TI1
- IBM 4610 SureMark Point of Sale printer Model TI2
- IBM 4610 SureMark Point of Sale printer Model TI3
- IBM 4610 SureMark Point of Sale printer Model TI4
- IBM 4610 SureMark Point of Sale printer Model TI5

## PosNlineFeedDI

<b>Type</b>	long
<b>Default</b>	PosLPI_6 (4236 microns or 1/6 inch)
<b>Access</b>	CGS

The **PosNlineFeedDI** resource controls the approximate distance that the paper is fed when a line feed character is encountered in a DI station normal mode message.

The minimum line feed distance is 0 (zero) and the maximum is 90,000 microns (30,000 for the IBM 4610 SureMark printers).

For all printers except the IBM 4610 SureMark Point of Sale printer, there are 72 dot rows in 1 inch, and each print line has 9 dot rows. A dot row is 353 microns in length. To calculate the number of microns for a given lines per inch value, use the following formula, where  $lpi$  is the number of lines per inch:

$$\text{microns} = (72 / lpi) \cdot 353$$

**Note:** This resource is ignored on the IBM 4689 printers.

This resource is supported only by the following printers:

- IBM Model 2 printer
- IBM Model 3 printer
- IBM Model 3F printer
- IBM Model 3R printer
- IBM Model 4 printer
- IBM Model 4A printer
- IBM Model 4R printer
- IBM 4610 SureMark Point of Sale printer Model T11
- IBM 4610 SureMark Point of Sale printer Model T12
- IBM 4610 SureMark Point of Sale printer Model T13
- IBM 4610 SureMark Point of Sale printer Model T14
- IBM 4610 SureMark Point of Sale printer Model T15

## PosNlineFeedSJ

<b>Type</b>	long
<b>Default</b>	PosLPI_8 (3177 microns or 1/8 inch)
<b>Access</b>	CGS

The **PosNlineFeedSJ** resource controls the approximate distance that the paper is fed when a line feed character is encountered in a SJ station normal mode message.

The minimum line feed distance is 0 (zero) and the maximum is 90,000 microns.

There are 72 dot rows in 1 inch, and each print line has 9 dot rows. A dot row is 353 microns in length. To calculate the number of microns for a given lines per inch value, use the following formula, where  $lpi$  is the number of lines per inch:

$$\text{microns} = (72 / lpi) \cdot 353$$

**Note:** This resource is ignored on the IBM Model 2 printer, the IBM 4689 printers, and on the IBM 4610 SureMark printers.

This resource is supported only by the following printers:

- IBM Model 3 printer
- IBM Model 3F printer
- IBM Model 3R printer
- IBM Model 4 printer
- IBM Model 4A printer

- IBM Model 4R printer

## PosNprintAlignment

<b>Type</b>	short
<b>Default</b>	PosPRINT_ALIGN_LEFT
<b>Access</b>	CGS

The **PosNprintAlignment** resource controls the alignment of the text in the printer at both the CR and the DI stations. The supported values for this resource are:

**PosPRINT\_ALIGN\_LEFT (0x0000)**

Left justifies the print lines.

**PosPRINT\_ALIGN\_CENTER (0x0001)**

Center justifies the print lines.

**PosPRINT\_ALIGN\_RIGHT (0x0002)**

Right justifies the print lines.

**Note:** This resource is supported only on the IBM 4610 SureMark Point of Sale printers.

## PosNprintColorMode

<b>Type</b>	short
<b>Default</b>	PosPAPER_NORMAL
<b>Access</b>	CGS

The **PosNprintColorMode** resource informs the printer of the type of paper that is loaded in the printer at CR station. The supported values for this resource are:

**PosPAPER\_NORMAL(0x00)**

Disable Color Mode Printing

**PosPAPER\_TWO\_COLOR (0x01)**

Two Color Paper

**Notes:**

1. For other valid values refer to the 4610 SureMark Point of Sale Printer *User's Guide*.
2. Not supported in the 4610 SureMark Point of Sale Printer Model T11/TI2.

## PosNprintCRCharSetx

<b>Type</b>	short
<b>Default</b>	PosPRINT_NONE
<b>Access</b>	G

The **PosNprintCRCharSetx** resources (PosNprintCRCharSet1, 2, 3, and 4) return a value that corresponds to the User-defined font that is stored on the printer.

The **PosNprintCRCharSetx** resource has the following valid values:

**PosPRINT\_NONE (0x03)**

No fonts are stored on the printer, or could not obtain font information from printer.

**PosPRINT\_PARTIAL (0x01)**

The font stored is the second part of a proportional font.

**PosPRINT\_FIXEDFONT (0x02)**

The font is a fixed size font.

**PosPRINT\_PROPORTIONAL (0x00)**

The font is a proportional font.

**Note:** Not supported in the 4610 SureMark Point of Sale Printer Models TI1/TI2.

**PosNprintDICharSetx**

<b>Type</b>	short
<b>Default</b>	PosPRINT_NONE
<b>Access</b>	G

**PosNprintDICharSetx** resources (PosNprintDICharSet1 and 2) return a value that corresponds to the User-defined font that is stored on the printer.

The **PosNprintDICharSetx** resource has the following valid values:

**PosPRINT\_NONE (0x03)**

No fonts are stored on the printer, or could not obtain font information from printer.

**PosPRINT\_PARTIAL (0x01)**

The stored font is in the second part of a proportional font

**PosPRINT\_FIXEDFONT (0x02)**

The font is a fixed size font.

**PosPRINT\_PROPORTIONAL (0x00)**

The font is a proportional font.

**Note:** Not supported in the 4610 SureMark Point of Sale Printer Models TI1/TI2.

**PosNprintFeatures**

<b>Type</b>	long
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNprintFeatures** resource indicates the current features of the printer. If a MICR reader is installed, this resource will have the PosFEATURE\_MICR\_INSTALLED bit set. If a check flipper is installed, this resource will have the PosFEATURE\_FLIPPER\_INSTALLED bit set.

This read-only resources has the following possible values:

**PosFEATURE\_MICR\_INSTALLED (0x0001)**

This bit is set if a MICR reader is installed in the printer.

**PosFEATURE\_FLIPPER\_INSTALLED (0x0002)**

This bit is set if a check flipper is installed in the printer.

**PosFEATURE\_58mm\_Paper (0x0004)**

This bit enables or disables printing on 58-mm paper.

**Note:** This resource is supported only on the IBM 4610 SureMark Point of Sale printers.

**PosNprintMode**

<b>Type</b>	short
<b>Default</b>	PosMODE_NORMAL
<b>Access</b>	CGS

The **PosNprintMode** resource controls the interpretation of the message bytes.

This resource can have the following values:

**PosMODE\_NORMAL (0x01)**

ASCII characters and double-byte characters are written to the printer.

**PosMODE\_LOGO (0x02)**

Logo data is written to the printer.

**PosMODE\_FISCAL (0x04)**

Fiscal commands are written to the printer. (Fiscal printers only)

**PosMODE\_LOAD\_MESSAGE (0x08)**

Download a pre-defined message to the printer. (IBM 4610 SureMark Point of Sale printers only)

**PosMODE\_LOAD\_LOGO (0x10)**

Download a pre-defined logo to the printer. (IBM 4689 Models 3x1 and TD5 printers and IBM 4610 SureMark Point of Sale printers only)

The default value on the:

Non-fiscal printers is PosMODE\_NORMAL

Fiscal printer is PosMODE\_FISCAL.

In normal mode, ASCII values, double-byte characters, control characters, and escape character sequences are processed by the printer device handler.

An error code of 4905 POSERR\_PRN\_INVALID\_MODE is returned if a value other than the following is specified::

- PosMODE\_NORMAL, PosMODE\_LOGO, PosMODE\_FISCAL (on fiscal printers)
- PosMODE\_LOAD\_MESSAGE (IBM 4610 SureMark Point of Sale printers only)
- PosMODE\_LOAD\_LOGO (IBM 4610 SureMark Point of Sale printers and IBM 4689-301, 3G1, 3M1, and TD5 printers only),

**Note:** This resource is supported on all the printers.

**PosNprintQualityMode**

<b>Type</b>	short
<b>Default</b>	PosHIGH_QUALITY_OFF
<b>Access</b>	CGS

The **PosNprintQualityMode** resource informs the printer. In the CR thermal station to print slower (35 LPS), which increases the print quality. The supported values for this resource are:

**PosHIGH\_QUALITY\_OFF (0x00)**

High Quality Text Off

**PosHIGH\_QUALITY\_ON (0x01)**

High Quality Text On

**Notes:**

1. Not supported in the document station.
2. Not supported in the 4610 SureMark Printer models T11/T12.

## PosNprintStation

<b>Type</b>	short
<b>Default</b>	PosPRINT_STATION_CR
<b>Access</b>	CGS

The **PosNprintStation** resource controls which station prints the data. Both PosPRINT\_STATION\_CR and PosPRINT\_STATION\_SJ can be specified by combining the values together. The data is written to each selected station for each *PosWrite()* subroutine call.

The PosPRINT\_STATION\_DI value cannot be used in combination with any other station.

This resource can have the following values:

**0 (zero)**

Turn off all printing. This allows the application to continue to send data to the printer device handler, but no data is printed.

**PosPRINT\_STATION\_CR (0x01)**

CR station. Causes the printer device handler to print only on the CR station

**PosPRINT\_STATION\_SJ (0x02)**

SJ station (Not supported on the IBM 4610 SureMark Point of Sale printers)

**PosPRINT\_STATION\_DI (0x04)**

DI station (Not supported on the IBM 4689 Model 3x1 and TD5 printers)

**PosPRINT\_STATION\_CR plus PosPRINT\_STATION\_SJ**

Any data sent to the printer device handler is printed on both the CR station and on the SJ station.

If an incorrect value is specified, an error code of 4904 POSERR\_PRN\_INVALID\_STATION is returned.

**Notes:**

1. For the IBM 4610 SureMark Point of Sale printers, the SJ station is not supported.
2. For the IBM 4689-301, 3G1, 3M1, and TD5 printers, the DI station is not supported.

This resource is supported on all printers.

## PosNprintStatus

<b>Type</b>	long
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNprintStatus** resource indicates the current status of the printer. This resource can only be queried by the `POS_SYS_GET_VALUES PosIOctl()` request.

This resource is a set of Boolean values. The TRUE state of each of these bits has the following meaning:

### **PosSTATUS\_COVER\_OPEN (0x0001)**

The cover is open. For the IBM 4610 SureMark Point of Sale printers and the IBM 4689 Model 3x1 and TD5 printers, the print station with the cover open will be indicated in the **PosNprintStatus2** resource.

### **PosSTATUS\_TRANSPORT\_ERROR (0x0002)**

The print head transport is not moving properly.

### **PosSTATUS\_SJ\_PAPER\_ERROR (0x0004)**

The SJ station is out of paper or has a paper jam.

### **PosSTATUS\_DOCUMENT\_AT\_FRONT (0x0008)**

A document is inserted at the front of the DI station.

### **PosSTATUS\_DOCUMENT\_AT\_TOP (0x0010)**

A document is inserted at the top of the DI station.

### **PosSTATUS\_DOCUMENT\_READY (0x0020)**

A document is registered in the DI station and is ready for printing.

### **PosSTATUS\_HEAD\_PARKED (0x0040)**

The head is parked in the location specified in the **PosNheadParkedPosition** resource.

### **PosSTATUS\_INSERTED\_FORWARD (0x0080)**

The document in the DI station was inserted from the front, and is registered with the top-most print line ready to print. If FALSE, this indicates that the document was inserted from the top, and is registered with the bottom-most print line ready to print. This field is valid only when **PosSTATUS\_DOCUMENT\_READY** is TRUE.

### **PosSTATUS\_ERROR\_PENDING (0x0100)**

An outstanding error is pending.

### **PosSTATUS\_DI\_FRONT\_LOAD\_ERROR (0x0200)**

A DI station print command is specified and the document is not registered.

### **PosSTATUS\_DI\_TOP\_LOAD\_ERROR (0x0400)**

A CR or SJ station print command is specified and a document in the DI station is covering the top sensor, or a DI station print command is specified and there is no document registered.

### **PosSTATUS\_PRINTER\_ONLINE (0x1000)**

The printer is online and ready to be used.

### **PosSTATUS\_MICR\_INSTALLED (0x2000)**

The printer has a MICR reader installed.

### **PosSTATUS\_CR\_PAPER\_LOW (0x4000)**

The paper in the CR station is low. (IBM 4689 printers only)

**PosSTATUS\_SJ\_PAPER\_LOW (0x8000)**

The paper in the SJ station is low. (IBM 4689 printers only)

This resource is supported on all the printers.

**PosNprintStatus2**

<b>Type</b>	long
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNprintStatus2** resource indicates additional printer status information. This resource can only be queried by the `POS_SYS_GET_VALUES PosIOctl()` request.

This resource is a set of Boolean values. The TRUE state of each of these bits has the following meaning:

**PosSTATUS2\_PAPER\_OUT (0x0001)**

One of the printer stations is out of paper. (IBM 4689 Model 3x1 and TD5 printers only)

**PosSTATUS2\_OVERHEAT (0x0002)**

The print head is overheating in one of the print stations. (IBM 4689 Model 3x1 and TD5 printers only)

**PosSTATUS2\_CR\_STATION (0x1000)**

The condition is in the CR station for the IBM 4689 Model 3x1 and TD5 printers. For the IBM 4610 SureMark Point of Sale printers, this condition means that either the CR station cover is open or the CR station is out of paper.

**PosSTATUS2\_DI\_STATION (0x2000)**

The condition is in the DI station for the IBM 4689 Model 3x1 and TD5 printers. For the IBM 4610 SureMark Point of Sale printers, the condition is that the DI station cover is open.

**PosSTATUS2\_SJ\_STATION (0x4000)**

The condition is in the SJ station. (IBM 4689 Model 3x1 and TD5 printers only)

**PosSTATUS2\_BUFFER\_HELD (0x8000)**

The printer print buffer is held, typically in response to an error. (IBM 4610 SureMark Point of Sale printers only)

**Note:** This resource is supported only on the IBM 4610 SureMark Point of Sale printers and the IBM 4689-301, 3G1, 3M1, and TD5 printers.

**PosNprintTabStops**

<b>Type</b>	unsigned char *
<b>Default</b>	100, 200, 300
<b>Access</b>	CGS

The **PosNprintTabStops** resource allows 5 tab stops to be set. The following conditions must be met:

- The `PosNprintTabStops` values must be greater than or equal to 1, and the values must be less than the number of dots for the station.

- There can be five tabs set per station and each station's tabs are set when that station is active.
- The syntax for this command is a number followed by a comma (for example: "10,20,30,40" sets tab stops at dot positions 10, 20, 30, and 40).
- Tabs must be in ascending order.
- Spaces are not allowed in the string.
- This resource is supported on Single Station SureMark printers.

## PosNprintToneDuration

<b>Type</b>	short
<b>Default</b>	0x03 (.3 seconds)
<b>Access</b>	CGS

The **PosNprintToneDuration** resource controls the duration of the internal speaker. The values can be:

**0x01 - 0xfe** 0.1 seconds times the specified number.

This resource is supported on Single Station SureMark printers.

## PosNprintToneFrequency

<b>Type</b>	int
<b>Default</b>	2093 Hz
<b>Access</b>	GS

The **PosNprintToneFrequency** resource controls the frequency of the internal speaker. The frequency values can be: 261 – 3953

### Notes:

1. Changing the value in **PosNprintToneFrequency** changes the **PosNprintToneNote** and **PosNprintToneOctave**.
2. Changing the **PosNprintToneNote** or **PosNprintToneOctave** changes the **PosNprintToneFrequency**.

This resource is supported on Single Station SureMark printers.

## PosNprintToneNote

<b>Type</b>	int
<b>Default</b>	PosPRINT_TONE_C
<b>Access</b>	CGS

The **PosNprintToneNote** resource controls the musical pitch of the internal speaker. The values can be:

<b>PosPRINT_TONE_C (0x00)</b>	musical note C
<b>PosPRINT_TONE_C_SHARP (0x01)</b>	musical note C sharp
<b>PosPRINT_TONE_D (0x02)</b>	musical note D

<b>PosPRINT_TONE_D_SHARP (0x03)</b>	musical note D sharp
<b>PosPRINT_TONE_E (0x04)</b>	musical note E
<b>PosPRINT_TONE_F (0x05)</b>	musical note F
<b>PosPRINT_TONE_F_SHARP (0x06)</b>	musical note F sharp
<b>PosPRINT_TONE_G (0x07)</b>	musical note G
<b>PosPRINT_TONE_G_SHARP (0x08)</b>	musical note G sharp
<b>PosPRINT_TONE_A (0x09)</b>	musical note A
<b>PosPRINT_TONE_A_SHARP (0x0A)</b>	musical note A sharp
<b>PosPRINT_TONE_B (0x0B)</b>	musical note B

**Notes:**

1. Changing the value in **PosNprintToneFrequency** changes the **PosNprintToneNote** and **PosNprintToneOctave**.
2. Changing the **PosNprintToneNote** or **PosNprintToneOctave** changes the **PosNprintToneFrequency**.

This resource is supported on Single Station SureMark printers.

## PosNprintToneOctave

<b>Type</b>	short
<b>Default</b>	PosPRINT_OCTAVE4
<b>Access</b>	CGS

The **PosNprintToneOctave** resource controls the octave (degrees between tones) of the internal speaker. The values can be:

<b>PosPRINT_OCTAVE1 (0x00)</b>	first octave
<b>PosPRINT_OCTAVE2 (0x01)</b>	second octave
<b>PosPRINT_OCTAVE3 (0x02)</b>	third octave
<b>PosPRINT_OCTAVE4 (0x03)</b>	fourth octave

**Notes:**

1. Changing the value in **PosNprintToneFrequency** changes the **PosNprintToneNote** and **PosNprintToneOctave**.
2. Changing the **PosNprintToneNote** or **PosNprintToneOctave** changes the **PosNprintToneFrequency**.

**Note:** This resource is supported on Single Station SureMark printers.

## PosNprintToneVolume

<b>Type</b>	short
<b>Default</b>	PosPRINT_TONE_SOFT
<b>Access</b>	CGS

The **PosNprintToneVolume** resource controls the volume of the internal speaker. The values can be:

<b>PosPRINT_TONE_SOFT (0x00)</b>	Tone volume soft
<b>PosPRINT_TONE_LOUD (0x01)</b>	Tone volume loud

This resource is supported on Single Station SureMark printers.

## PosNprintUpsideDown

<b>Type</b>	short
<b>Default</b>	PosPRINT_RIGHT_SIDE_UP
<b>Access</b>	CGS

The **PosNprintUpsideDown** resource informs the printer to switch between normal printing and printing upside down (opposite from normal printing). This command allows the data to be right side up as it comes out of the printer. The supported values for this resource are:

<b>PosPRINT_RIGHT_SIDE_UP (0x00)</b>	Disable upside down printing. Text is printed normally.
<b>PosPRINT_UPSIDE_DOWN (0x01)</b>	Enable upside down printing.

### Notes:

1. This resource is not supported on the DI Station or Model 4610 T11 and T12 printers
2. When the printer is wall mounted, the front of the printer will be pointed up. The print data will appear upside down when printed normally. This command allows the data to be right side up as it comes out of the printer. **The data must be sent to the printer last line first.**

## PosNrawPrintStatus

<b>Type</b>	unsigned char *
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNrawPrintStatus** resource returns the current status bytes of the printer. This resource returns 16 bytes. The printer status bytes are left-justified in the returned bytes. Any byte not defined by the printer will be set to 0 (zero).

This resource can only be queried by the `POS_SYS_GET_VALUES PosIOctl()` request..

This resource is supported on all the printers.

## PosNresumeString

<b>Type</b>	unsigned char *
<b>Default</b>	86 blank characters

**Access** S

The **PosNResumeString** resource specifies the overlay string to print when a `POS_PRN_RESUME_PRINTING` *Chapter 19. PosIOCtl() Requests* subroutine call is made after a printer error is detected.

The default value is a blank line.

**Model 2, Model 3, Model 3F, Model 3R, Model 4, Model 4A, and Model 4R Printers**

For the Model 2, Model 3, Model 3F, Model 3R, Model 4, Model 4A, and Model 4R printers, the character string is printed at 15 CPI on the station where the error occurred.

If the `POS_PRN_RESUME_PRINTING` request is for the DI station and **PosNDIWidth** is `PosDI_WIDE`, up to 86 single-byte characters will be printed. If fewer than 86 bytes are defined, the data will be padded with spaces before being printed. If more than 86 bytes are defined, only the first 86 will be printed.

If the `POS_PRN_RESUME_PRINTING` request is for either the CR or SJ stations, or if it is for the DI station and **PosNDIWidth** is set to `PosDI_NARROW`, only the first 38 single-byte characters will be printed. If less than 38 bytes are defined, the data will be padded with spaces before being printed. If more than 38 bytes are defined, only the first 38 will be printed.

**4689-001 and 4689-002 Printers**

For the 4689-00x printers, the character string is printed at the current CPL value on the station where the error occurred.

If the `POS_PRN_RESUME_PRINTING` request is for the DI station, up to 58 single-byte characters at 25 CPL or up to 70 single-byte characters at 30 CPL will be printed. If fewer than 58 (70 at 30 CPL) bytes are defined, the data will be padded with spaces before being printed. If more than 58 (70 at 30 CPL) bytes are defined, only the first 58 (70 at 30 CPL) will be printed.

If the `POS_PRN_RESUME_PRINTING` request is for either the CR or SJ stations, only the first 25 (30 at 30 CPL) single-byte characters will be printed. If less than 25 (30 at 30 CPL) bytes are defined, the data will be padded with spaces before being printed. If more than 25 (30 at 30 CPL) bytes are defined, only the first 25 or 30 will be printed.

**4689-3x1 and TD5 printers**

For the IBM 4689 Model 3x1 and TD5 printers, the character string is printed at the current characters per line (CPL) value on the station where the error occurred.

**4610 SureMark Point of Sale Printers (all models)**

For the 4610 SureMark Point of Sale printers, the character string is printed at the current characters per line (CPL) value on the station where the error occurred.

This resource is supported on all the printers.

## PosNretryString

<b>Type</b>	unsigned char *
<b>Default</b>	86 single-quote characters
<b>Access</b>	S

The **PosNretryString** resource specifies the overlay string to print when a `POS_PRN_RETRY_PRINTING` *Chapter 19. PosIOctl() Requests* subroutine call is made after a printer error is detected.

The default value is a line of single quote characters at 15 CPI across the width of the station that the error occurred on.

For all printers, the overlay string will only be printed if an error occurred during a command which prints data and the printer device handler has sent the command to the printer.

### **Model 2, Model 3, Model 3F, Model 3R, Model 4, Model 4A, and Model 4R Printers**

For the Model 2, Model 3, Model 3F, Model 3R, Model 4, Model 4A, and Model 4R printers, the character string is printed at 15 CPI on the station where the error occurred.

If the `POS_PRN_RETRY_PRINTING` request is for the DI station and **PosNDIWidth** is `PosDI_WIDE`, up to 86 single-byte characters will be printed. If less than 86 bytes are defined, the data will be padded with spaces before being printed. If more than 86 bytes are defined, only the first 86 will be printed.

If the `POS_PRN_RETRY_PRINTING` request is for either the CR or SJ stations, or if it is for the DI station and **PosNDIWidth** is `PosDI_NARROW`, only the first 38 single-byte characters will be printed. If less than 38 bytes are defined, the data will be padded with spaces before being printed. If more than 38 bytes are defined, only the first 38 will be printed.

### **4689-001 and 4689-002 Printers**

For the 4689-00x printers, the character string is printed at the current CPL value on the station where the error occurred.

If the `POS_PRN_RETRY_PRINTING` request is for the DI station, up to 58 single-byte characters at 25 CPL (70 single-byte characters at 30 CPL) will be printed. If fewer than 58 single-byte characters (70 single-byte characters at 30 CPL) are defined, the data will be padded with spaces before being printed. If more than 58 single-byte characters (70 at 30 CPL) are defined, only the first 58 single-byte characters (70 at 30 CPL) will be printed.

If the `POS_PRN_RETRY_PRINTING` request is for either the CR or SJ stations, only the first 25 single-byte characters (30 at 30 CPL) will be printed. If less than 25 single-byte characters (30 at 30 CPL) are defined, the data will be padded with spaces before being printed. If more than 25 single-byte characters (30 at 30 CPL) are defined, only the first 25 or 30 will be printed.

### **4689 Model 3x1 and TD5 printers**

For the IBM 4689 Model 3x1 and TD5 printers, the character string is printed at the current CPL value on the station where the error occurred.

### 4610 SureMark Point of Sale Printers (all models)

For the 4610 SureMark Point of Sale printers, only the first character of the character string is printed at the current CPL value on the station where the error occurred. This character will be printed only if the retry is for a line that has a home error. In all other instances the line will be printed without the overlay character in the first position.

This resource is supported on all the printers.

## PosRs232c Resource Set

**Class Name** PosRs232c  
**Include** <pos/rs232c.h>

The PosRs232c resource set controls a group of devices that includes:

- RS-232C devices that are attached to the serial ports on a 4693-2x2
- RS-232C devices that are attached to an IBM Feature E expansion card
- RS-232C devices that have been modified by their manufacturers to attach directly to an SIO channel
- USB devices which conform to the IBM USB Pseudo-RS-232C Interface Specification.

A copy of the PosRs232c resource set is created every time an application opens an RS-232C device using the *PosOpen()* subroutine call. The resource set is associated with it until the RS-232C device connection is closed. Changing the resources of one opened RS-232C device does not affect any other RS-232C device.

The following table gives an overview of the resources in this set.

Table 21-10. PosRs232c Resources

Name	Type	Default	Access
PosNbaudRate	int	PosBAUD_RATE_300	CG
PosNdataBits	int	PosDATA_BITS_7	CG
PosNlineMode	int	PosRECV_ENABLE I PosLOCAL_READY_ACTIVE I PosLOCAL_OK_TO_SEND_ACTIVE	CGS
PosNparity	int	PosPARITY_EVEN	CG
PosNrs232Status	int	n/a	G
PosNstopBits	int	PosSTOP_BITS_1	CG
PosNtimeoutChar	int	16	CG

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

### PosNbaudRate

**Type** int  
**Default** PosBAUD\_RATE\_300  
**Access** CG

The **PosNbaudRate** resource controls the baud rate of the communication. This resource can have the following values:

<b>PosBAUD_RATE_110 (110)</b>	110 bits per second
<b>PosBAUD_RATE_300 (300)</b>	300 bits per second
<b>PosBAUD_RATE_1200 (1200)</b>	1200 bits per second
<b>PosBAUD_RATE_2400 (2400)</b>	2400 bits per second
<b>PosBAUD_RATE_4800 (4800)</b>	4800 bits per second
<b>PosBAUD_RATE_9600 (9600)</b>	9600 bits per second

## PosNdataBits

<b>Type</b>	int
<b>Default</b>	PosDATA_BITS_7
<b>Access</b>	CG

The **PosNdataBits** resource identifies the length of the data bits. This resource can have the following values:

<b>PosDATA_BITS_5 (5)</b>	5 bits per character
<b>PosDATA_BITS_6 (6)</b>	6 bits per character
<b>PosDATA_BITS_7 (7)</b>	7 bits per character
<b>PosDATA_BITS_8 (8)</b>	8 bits per character

## PosNlineMode

<b>Type</b>	int
<b>Default</b>	PosRECV_ENABLE   PosLOCAL_READY_ACTIVE   PosLOCAL_OK_TO_SEND_ACTIVE
<b>Access</b>	CGS

The **PosNlineMode** resource identifies the state of the lines that are output by the local RS-232C device. If the device is currently acquired, any changes to the **PosNlineMode** resource causes the state of the physical output lines to change state accordingly. This resource can have the following values:

### **PosLOCAL\_READY\_ACTIVE (0x01)**

Asserts the communications line that indicates that this RS-232C device is ready. (This asserts DSR on a Feature E card and DTR on IBM 4693 Point of Sale Terminal Models 2x2.)

### **PosLOCAL\_OK\_TO\_SEND\_ACTIVE (0x02)**

Asserts the communications line that indicates that the remote RS-232C device is clear to send data. (This asserts CTS on a Feature E card and RTS on IBM 4693 Point of Sale Terminal Models 2x2.)

### **PosRECV\_ENABLE (0x20)**

Enables the receiver.

### **PosDTR\_NOT\_REQUIRED\_FOR\_XMIT (0x40)**

Indicates that IBM Point of Sale Subsystem should transmit to the device regardless of the state of DTR.

If either of the control lines (DTR or RTS) are dropped (negated) when the remote device is actively transmitting data, data might be lost or repeated.

## PosNparity

<b>Type</b>	int
<b>Default</b>	PosPARITY_EVEN
<b>Access</b>	CG

The **PosNparity** resource identifies the parity.

This resource can have the following values:

<b>PosPARITY_NONE (0)</b>	None
<b>PosPARITY_ODD (1)</b>	Odd
<b>PosPARITY_EVEN (2)</b>	Even

## PosNrs232Status

<b>Type</b>	int
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNrs232Status** resource indicates the present status of the RS-232C port.

This resource is a set of Boolean values. The TRUE state of each of these bits has the following meaning:

### **PosLOCAL\_READY\_ACTIVE (0x01)**

The signal is asserted that indicates that the local RS-232 device is ready. (DSR on a Feature E card, DTR on IBM 4693 Point of Sale Terminal Models 2x2.)

### **PosLOCAL\_OK\_TO\_SEND\_ACTIVE (0x02)**

The signal is asserted that indicates that the local RS-232 device is ready to receive data. (CTS on a Feature E card, RTS on IBM 4693 Point of Sale Terminal Models 2x2.)

### **PosREMOTE\_READY\_ACTIVE (0x04)**

The signal is asserted that indicates that the remote RS-232 device is ready. (DTR on a Feature E card, DSR on IBM 4693 Point of Sale Terminal Models 2x2.)

### **PosREMOTE\_OK\_TO\_SEND\_ACTIVE (0x08)**

The signal is asserted that indicates that the remote RS-232 device is ready to receive data. (RTS on a Feature E card, CTS on IBM 4693 Point of Sale Terminal Models 2x2.)

### **PosXMIT\_BUF\_EMPTY (0x10)**

The transmit buffer is empty.

### **PosRECV\_ENABLE (0x20)**

The receiver of the local device is enabled.

## PosNstopBits

<b>Type</b>	int
<b>Default</b>	PosSTOP_BITS_1
<b>Access</b>	CG

The **PosNstopBits** resource identifies the length of the stop bits.

This resource can have the following values:

<b>PosSTOP_BITS_1 (10)</b>	1 bit
<b>PosSTOP_BITS_1_5 (15)</b>	1.5 bits
<b>PosSTOP_BITS_2 (20)</b>	2 bits

## PosNtimeoutChar

<b>Type</b>	int
<b>Default</b>	16 character times
<b>Access</b>	CG

The **PosNtimeoutChar** resource specifies the number of character times that must elapse after a data transmission before a timeout occurs. The timeout will cause the contents of the RS-232C receive buffer to be returned to the application. (A character time is equivalent to the amount of time required for one character to be transmitted at the current baud rate.)

Values cannot be negative. The maximum number of characters depends on the baud rate and the number of data bits. The default value is when the baud rate is 300 baud and the default value for the data bits length is 7 bits.

Table 21-11. Maximum PosNtimeoutChar Value

PosNbaudRate	PosDATA_BITS_5	PosDATA_BITS_6	PosDATA_BITS_7	PosDATA_BITS_8
PosBAUD_RATE_110	102	91	83	76
PosBAUD_RATE_300	278	250	227	208
PosBAUD_RATE_600	557	501	455	417
PosBAUD_RATE_1200	1114	1002	911	835
PosBAUD_RATE_2400	2228	2005	1823	1671
PosBAUD_RATE_4800	4456	4010	3646	3342
PosBAUD_RATE_9600	8912	8021	7292	6684

---

## PosScale Resource Set

<b>Class Name</b>	PosScale
<b>Include</b>	<pos/scale.h>

A copy of the PosScale resource set is created every time an application opens a scale device by using the *PosOpen()* subroutine call. The resource set associated with a scale connection when it is opened remains associated with it until the scale connection is closed. Changing the resources of one opened scale does not affect any of the other scales.

“Scale Default Values” on page 15-4 lists the default modes of each of the supported scales. It is a cross-reference to this section which allows you to more easily determine which resources are supported by each scale.

The following table gives an overview of the resources in this set.

*Table 21-12. PosScale Resources*

<b>Name</b>	<b>Type</b>	<b>Default</b>	<b>Access</b>
PosNdisplayRequired	int	PosENABLE	CG
PosNnumWeightDigits	int	PosFOUR_WEIGHT_DIGITS	CG
PosNoperMode	int	PosUSCANADA	CG
PosNvibrationFilter	int	PosLOW	CG
PosNweightMode	int	PosENGLISH	CG
PosNzeroIndState	int	PosENABLE	CG
PosNzeroRetState	int	PosDISABLE	CG

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## PosNdisplayRequired

<b>Type</b>	int
<b>Default</b>	PosENABLE
<b>Access</b>	CG

The **PosNdisplayRequired** resource specifies whether a remote scale display is required. If the application software is capable of displaying weight on the system display, the IBM 4696 Point of Sale Scanner Scale can be run without the remote display. If you run the IBM 4696 Point of Sale Scanner Scale without the remote display, this resource should be set to PosDISABLE to avoid error messages. It is your responsibility to ensure that a system operating without a remote display meets the applicable weights and measures regulations.

If the value of this resource is PosDISABLE but there is a remote display attached to the scale device, the scale display will be completely blank. If the value of this resource is PosENABLE but there is no remote display attached to the scale device, the scale will beep six times in rapid succession and will then go offline.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	A remote scale display is not required.
<b>PosENABLE (0x01)</b>	A remote scale display is required.

This resource is supported only by the following scales:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4698 Point of Sale Scanner Model 2
- IBM USB Scale Interface Specification

## PosNnumWeightDigits

<b>Type</b>	int
<b>Default</b>	PosFOUR_WEIGHT_DIGITS
<b>Access</b>	CG

The **PosNnumWeightDigits** resource specifies the number of digits to return for an English weight (pounds). Four digits of weight data implies units of hundredths of pounds; five digits of weight data implies units of thousandths of pounds.

This resource can have the following values:

**PosFOUR\_WEIGHT\_DIGITS (0x00)**

Scale will return four weight digits when configured in English weight mode.

**PosFIVE\_WEIGHT\_DIGITS (0x01)**

Scale will return five weight digits when configured in English weight mode.

**Note:** This resource has no effect when the scale is configed in Metric weight mode. In Metric weight mode, five weight digits are always returned and the weight is in units of thousandths of kilograms.

This resource is supported only by the following scales:

- IBM USB Scale Interface Specification

## PosNoperMode

<b>Type</b>	int
<b>Default</b>	PosUSCANADA
<b>Access</b>	CG

The **PosNoperMode** resource specifies the regulations that the point-of-sale system must conform to. The weight and measures requirements in various countries differ, and require minor operational differences.

This resource can have the following values:

**PosUSCANADA (0x00)**

The point-of-sale system must conform to regulations specified by the United States NIST Handbook 44 and the Canadian Department of Consumer and Corporate Affairs, Weights and Measurement Act, Specifications SGM-1.

**PosUK (0x01)**

The point-of-sale system must conform to regulations specified by the Non-automatic Weighing Instruments (EEC Requirements) Regulations 1992 (based on OIML R 76-1).

When the value of this resource is PosUK, weight data numbers are not displayed until a *PosRead()* call is issued. When an item is placed on the scale, the decimal point (.) and the units designator (lb or kg) are displayed but blank characters appear where the numbers would be until the *PosRead()* call is issued.

This resource is supported only by the following scales:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4698 Point of Sale Scanner Model 2
- IBM USB Scale Interface Specification

## PosNvibrationFilter

<b>Type</b>	int
<b>Default</b>	PosLOW
<b>Access</b>	CG

The **PosNvibrationFilter** resource controls the scale's sensitivity to vibration. External vibrations can affect the stability of the scale. There is a programmable vibration filter that allows you to reduce the scale's sensitivity to vibration.

Reducing the scale's sensitivity to vibration increases the scale settling time slightly, so the higher vibration filter settings should be selected only when testing reveals a stability problem in the user's checkstand.

This resource can have the following values:

<b>PosLOWEST (0x00)</b>	Lowest level of vibration filtering.
<b>PosLOW (0x01)</b>	Low level of vibration filtering.
<b>PosMEDIUM (0x02)</b>	Medium level of vibration filtering.
<b>PosHIGH (0x03)</b>	High level of vibration filtering.

This resource is supported only by the following scales:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4698 Point of Sale Scanner Model 2
- IBM USB Scale Interface Specification

## PosNweightMode

<b>Type</b>	int
<b>Default</b>	PosENGLISH
<b>Access</b>	CG

The **PosNweightMode** resource determines whether the weight is returned in pounds (Avoirdupois or English system) or in kilograms (Metric system).

For the IBM 4696 scale, this resource alone causes the hardware to switch from one unit of weight to the other. For the IBM 4687 scale, switching from one unit of weight to the other requires that this resource be set to the desired mode and that the hardware switches located next to the display cable socket be set correctly for the desired unit of weight.

This resource can have the following values:

### **PosENGLISH (0x00)**

Weight is given in pounds. The PosNnumWeightDigits resource specifies the number of digits to return for an English weight (pounds). SIO-attached scales always return four digits which represent the weight of the item in hundredths of pounds. USB-attached scales return four or five digits which represent the weight of the item in hundredths or thousandths of pounds, respectively.

### **PosMETRIC (0x01)**

Weight is given in kilograms. All scales return five digits which represent the weight of the item in thousandths of kilograms.

This resource is supported by all scales.

## PosNzeroIndState

<b>Type</b>	int
<b>Default</b>	PosENABLE
<b>Access</b>	CG

The **PosNzeroIndState** resource specifies whether the center-of-zero is indicated by a light-emitting diode (LED).

This resource can have the following values:

**PosDISABLE (0x00)**

Do not indicate center-of-zero with LED.

**PosENABLE (0x01)**

Indicate center-of-zero with LED.

This resource is supported only by the following scales:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4698 Point of Sale Scanner Model 2
- IBM USB Scale Interface Specification

## PosNzeroRetState

<b>Type</b>	int
<b>Default</b>	PosDISABLE
<b>Access</b>	CG

The **PosNzeroRetState** resource specifies whether zero protection is required. With zero protection enabled, the scale will not answer weight requests if:

- A negative weight value is indicated on the display prior to placing the item for weighing on the scale. The scale must be zero-adjusted (reset to zero) before weight requests will be answered.
- An item is left on the scale for four minutes. The item must be removed, allowing the scale to return to zero, before weight requests will be answered.

If either of the above conditions exists and the value of this resource is PosENABLE, POS\_SCALE\_REQUIRES\_ZEROING will be set in the *Flags* field of the scale data buffer which is returned in response to a *PosRead()* subroutine call. The scale display will show -0- after *PosRead()* is attempted.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	Zero protection is disabled.
<b>PosENABLE (0x01)</b>	Zero protection is enabled.

This resource is supported only by the following scales:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4698 Point of Sale Scanner Model 2
- IBM USB Scale Interface Specification

---

## PosScanner Resource Set

<b>Class Name</b>	PosScanner
<b>Include</b>	<pos/scanner.h>

A copy of the PosScanner resource set is created every time an application opens a scanner device by using the *PosOpen()* subroutine call. The resource set associated with a scanner connection when it is opened remains associated with it until the scanner connection is closed. Changing the resources of one opened scanner does not affect any of the other scanners. See “Configuring the Scanner” on page 16-8 for important information about changing scanner resources.

“Scanner Default Values” on page 16-9 lists the default modes of each of the supported scanners. It is a cross-reference to this section which allows you to more easily determine which resources are supported by each scanner.

The following table gives an overview of the resources in this set.

Table 21-13. PosScanner Resources

Name	Type	Default	Access
PosNbarCodes1	long	(device dependent)	CG
PosNbarCodes2	long	(device dependent)	CG
PosNbarCodes3	long	PosLGROUP_NONE	CG
PosNbarCodes4	long	PosLGROUP_NONE	CG
PosNbarCodeProgramming	int	PosDiscrete	CG
PosNbeepLength	int	PosMEDLONG	CG
PosNbeepState	int	PosENABLE	CGS
PosNbeepVolume	int	PosHIGH	CG
PosNblinkLength	int	PosLONG	CG
PosNblockReadMode	int	1	CG
PosNblock1Type	long	PosLABEL_NO_CHECK	CG
PosNblock2Type	long	PosLABEL_NO_CHECK	CG
PosNblock3Type	long	PosLABEL_NO_CHECK	CG
PosNbVolSwitchState	int	PosENABLE	CG
PosNcheckModulo	int	PosDISABLE	CG
PosNcode128ScansPerRead	int	2	CG
PosNcode39ScansPerRead	int	2	CG
PosNdecodeAlgorithm	int	PosENABLE	CG
PosNdReadTimeout	int	PosSHORT	CG
PosNdTouchMode	int	PosDISABLE	CG
PosNeAN13ScansPerRead	int	2	CG
PosNeAN8ScansPerRead	int	2	CG
PosNiTFLength1	int	0	CG
PosNiTFLength2	int	0	CG
PosNiTFScansPerRead	int	2	CG
PosNjANTwoLabelDecode	int	PosDISABLE	CG
PosNlabelsQueued	int	n/a	G
PosNlaserSwitchState	int	PosENABLE	CG
PosNlaserTimeout	int	15 (minutes)	CG
PosNmotorTimeout	int	15 (minutes)	CG
PosNqueueAllLabels	int	PosENABLE	CGS
PosNscansPerRead	int	2	CG
PosNstoreScansPerRead	int	2	CG
PosNsupplementals	int	2	CG
PosNtransmitCheckDigit	int	PosUPCE_UPCA_CHECK_DIGIT	CG
PosNtwoLabelFlagPair1	short	0x2122	CG

Table 21-13. PosScanner Resources (continued)

Name	Type	Default	Access
PosNtwoLabelFlagPair2	short	0x2128	CG
PosNtwoLabelFlagPair3	short	0x2129	CG
PosNtwoLabelFlagPair4	short	0x2122	CG
PosNuPCAScansPerRead	int	2	CG
PosNuPCDScansPerRead	int	2	CG
PosNuPCEscansPerRead	int	2	CG
PosNuPCExpansion	int	NO_EXPANSION	CG
PosNverifyPriceChk	int	NO_VERIFY	CG

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## Scanner Model Identifiers

There are several tables in this section that list the valid values for some of the scanner resources. Below are the descriptions for the terms listed in the “Supported On” column in these tables:

<b>HHBCR</b>	Hand-Held Bar Code Reader Models 1 and 2
<b>HHBCR-1</b>	Hand-Held Bar Code Reader Model 1
<b>HHBCR-2</b>	Hand-Held Bar Code Reader Model 2
<b>4685</b>	IBM 4685 Hand-Held Bar Code Reader Models 1 and L01
<b>4686</b>	IBM 4686 Retail Point of Sale Scanner Models 1, 2, 3 and 4
<b>4686-1/2</b>	IBM 4686 Retail Point of Sale Scanner Models 1 and 2
<b>4686-3/4</b>	IBM 4686 Retail Point of Sale Scanner Models 3 and 4
<b>4696</b>	IBM 4696 Retail Point of Sale Scanner Model 1
<b>4697</b>	IBM 4697 Retail Point of Sale Scanner Model 1
<b>4698</b>	IBM 4698 Retail Point of Sale Scanner Models 1 and 2
<b>USB</b>	IBM USB Scanner Interface Specification

## PosNbarCodes1

<b>Type</b>	long
<b>Default</b>	(device dependent)
<b>Access</b>	CG

The **PosNbarCodes1** resource is one of four resources which determine what combination of bar code types a scanner will recognize. When configured in a specific mode, a scanner recognizes and returns all bar code types that are associated with that mode.

The following table lists the values that can be given for the **PosNbarCodes1** resource for both SIO- and USB-attached scanners. The default value for each supported scanner type is also indicated in the table.

Table 21-14. Valid Values for PosNbarCodes1 Resource (SIO and USB Scanners)

Resource Value	Bar Code Types	Supported On
PosLGROUP_STD_2_OF_5 (0x00000004)	Standard 2-of-5	HHBCR, 4685
PosLGROUP_INT_2_OF_5 (0x00000008)	Interleaved 2-of-5	1520, HHBCR, 4685, USB

Table 21-14. Valid Values for PosNbarCodes1 Resource (SIO and USB Scanners) (continued)

Resource Value	Bar Code Types	Supported On
PosLGROUP_UPC_D1_TO_D5 (0x00000F80)	UPC-D1, UPC-D2, UPC-D3, UPC-D4, UPC-D5	1520, USB
PosLGROUP_UPC_EAN (0x0000F000)	UPC-8, UPC-13, UPC-A, UPC-E	1520(default), 4686, 4696, 4697, 4698, USB
PosLGROUP_UPC_EAN_PLUS_5 (0x0000F001)	EAN-8/13, UPC-A/E with 5-digit supplemental	4686-1/2, 4698, USB
PosLGROUP_UPC_EAN_PLUS_2 (0x0000F002)	EAN-8/13, UPC-A/E with 2-digit supplemental	4686-1/2, 4698, USB
PosLGROUP_UPC_EAN_PLUS_2_5 (0x0000F003)	EAN-8/13, UPC-A/E with 2- and 5-digit supplemental	4686-1/2, 4698, USB
PosLGROUP_UPC_EAN_ITF (0x00000F004)	EAN-8/13, UPC-A/E, Interleaved 2-of-5	HHBCR, 4685, USB
PosLGROUP_UPC_EAN_D3 (0x0000F200)	EAN-8/13, UPC-A/E, UPC-D3	HHBCR(default), 4685(default), USB
PosLGROUP_UPC_EAN_D1_TO_D5 (0x0000FF80)	EAN-8/13, UPC-A/E, UPC-D1 - UPC-D5	4686(default), 4696(default), 4697(default), 4698(default), USB(default)
PosLGROUP_CODE_128 (0x00010000)	Code 128	HHBCR-2, 4685, USB
PosLGROUP_UPC_EAN_CODE_128 (0x0001F000)	EAN-8/13, UPC-A/E, Code 128	HHBCR-2, 4685, USB
PosLGROUP_CODE_93 (0x00020000)	Code 93	HHBCR-2, 4685, USB
PosLGROUP_UPC_EAN_CODE_93 (0x0002F000)	EAN-8/13, UPC-A/E, Code 93	HHBCR-2, 4685, USB
PosLGROUP_CODE_39 (0x00040000)	Code 39	1520, USB
PosLGROUP_UPC_EAN_CODE_39 (0x0004F000)	UPC-8, UPC-13, UPC-A, UPC-E, Code 39	HHBCR, 4685, USB
PosLGROUP_UPC_EAN_CODABAR (0x0008F000)	EAN-8/13, UPC-A/E, Codabar	HHBCR, 4685, USB
PosLGROUP_UPC_EAN_2_5_CODABAR (0x0008F003)	EAN-8/13, UPC-A/E, Codabar; 2- and 5-digit supplementals optional	HHBCR-2, 4685, USB
PosLGROUP_PLUS_2_5_ONLY (0x8000F003)	EAN-8/13, UPC-A/E; 2- and 5-digit supplementals mandatory	HHBCR-2, 4685, USB

The following table lists the values that can be given for the **PosNbarCodes1** resource for USB-attached scanners only.

Table 21-15. Valid Values for PosNbarCodes1 Resource (USB Scanners Only)

Resource Value	Bar Code Types	Supported On
PosLGROUP_ALL (0x001FFF88)	Enables all bar code types for which the scanner can be configured.	USB

See "Scanner Model Identifiers" on page 21-62 for descriptions of the terms shown in the "Supported On" column.

**Note:** The Standard 2-of-5 symbology is not supported in the IBM USB Scanner Interface Specification. In order to support legacy applications that may use this value, IBM Point of Sale Subsystem will configure the scanner to read all bar code types if Standard 2-of-5 is requested with a USB scanner device.

## PosNbarCodes2

<b>Type</b>	long
<b>Default</b>	(device dependent)
<b>Access</b>	CG

The **PosNbarCodes2** resource is one of four resources which determine what combination of bar code types a scanner will recognize. When configured in a specific mode, a scanner recognizes and returns all bar code types that are associated with that mode.

The following table lists the values that can be given for the **PosNbarCodes2** resource for both SIO- and USB-attached scanners. The default value for each supported scanner type is also indicated in the table.

Table 21-16. Valid Values for PosNbarCodes2 Resource (SIO and USB Scanners)

Resource Value	Bar Code Types	Supported On
PosLGROUP_NONE (0x00000000)	None	1520, 4686(default), 4697(default), 4698(default), USB(default)
PosLGROUP_INT_2_OF_5 (0x00000008)	Interleaved 2-of-5	1520, 4686, 4697, 4698, USB
PosLGROUP_UPC_D1_TO_D5 (0x00000F80)	UPC-D1, UPC-D2, UPC-D3, UPC-D4, UPC-D5	1520(default), USB
PosLGROUP_UPC_EAN (0x0000F000)	UPC-8, UPC-13, UPC-A, UPC-E	1520, USB
PosLGROUP_CODE_128 (0x00010000)	Code 128	4686, 4697, 4698, USB
PosLGROUP_CODE_93 (0x00020000)	Code 93	4686, 4697, USB
PosLGROUP_CODE_39 (0x00040000)	Code 39	1520, 4686, 4697, 4698, USB
PosLGROUP_CODABAR (0x00080000)	Codabar	4686, 4697, USB

The following table lists the values that can be given for the **PosNbarCodes2** resource for USB-attached scanners only.

Table 21-17. Valid Values for PosNbarCodes2 Resource (USB Scanners Only)

Resource Value	Bar Code Types	Supported On
PosLGROUP_CODE_128_ITF (0x00010008)	Code 128, Interleaved 2-of-5	USB
PosLGROUP_CODE_93_ITF (0x00020008)	Code 93, Interleaved 2-of-5	USB
PosLGROUP_CODE_93_CODE_128 (0x00030000)	Code 93, Code 128	USB
PosLGROUP_CODE_39_ITF (0x00040008)	Code 39, Interleaved 2-of-5	USB
PosLGROUP_CODE_39_CODE_128 (0x00050000)	Code 39, Code 128	USB
PosLGROUP_CODE_39_CODE_93 (0x00060000)	Code 39, Code 93	USB
PosLGROUP_CODABAR_ITF (0x00080008)	Codabar, Interleaved 2-of-5	USB

Table 21-17. Valid Values for PosNbarCodes2 Resource (USB Scanners Only) (continued)

Resource Value	Bar Code Types	Supported On
PosLGROUP_CODABAR_CODE_128 (0x00090000)	Codabar, Code 128	USB
PosLGROUP_CODABAR_CODE_93 (0x000A0000)	Codabar, Code 93	USB
PosLGROUP_CODABAR_CODE_39 (0x000C0000)	Codabar, Code 39	USB
PosLGROUP_UCC_EAN_128 (0x00100000)	UCC/EAN-128	USB
PosLGROUP_UCC_EAN_128_ITF (0x00100008)	UCC/EAN-128, Interleaved 2-of-5	USB
PosLGROUP_UCC_EAN_128_CODE_128 (0x00110000)	UCC/EAN-128, Code 128	USB
PosLGROUP_UCC_EAN_128_CODE_93 (0x00120000)	UCC/EAN-128, Code 93	USB
PosLGROUP_UCC_EAN_128_CODE_39 (0x00140000)	UCC/EAN-128, Code 39	USB
PosLGROUP_UCC_EAN_128_CODABAR (0x00180000)	UCC/EAN-128, Codabar	USB

See “Scanner Model Identifiers” on page 21-62 for descriptions of the terms shown in the “Supported On” column.

## PosNbarCodes3

<b>Type</b>	long
<b>Default</b>	(device dependent)
<b>Access</b>	CG

The **PosNbarCodes3** resource is one of four resources which determine what combination of bar code types a scanner will recognize. When configured in a specific mode, a scanner recognizes and returns all bar code types that are associated with that mode.

The following table lists the values that can be given for the **PosNbarCodes3** resource for both SIO- and USB-attached scanners. The default value for each supported scanner type is also shown.

Table 21-18. Valid Values for PosNbarCodes3 and PosNbarCodes4 Resources (SIO and USB Scanners)

Resource Value	Bar Code Types	Supported On
PosLGROUP_NONE (0x00000000)	None	1520(default), 4698(default), USB(default)
PosLGROUP_INT_2_OF_5 (0x00000008)	Interleaved 2-of-5	1520, 4698, USB
PosLGROUP_UPC_D1_TO_D5 (0x00000F80)	UPC-D1, UPC-D2, UPC-D3, UPC-D4, UPC-D5	1520, USB
PosLGROUP_UPC_EAN (0x0000F000)	UPC-8, UPC-13, UPC-A, UPC-E	1520, USB
PosLGROUP_CODE_128 (0x00010000)	Code 128	4698, USB

Table 21-18. Valid Values for PosNbarCodes3 and PosNbarCodes4 Resources (SIO and USB Scanners) (continued)

Resource Value	Bar Code Types	Supported On
PosLGROUP_CODE_39 (0x00040000)	Code 39	1520, 4698, USB

The following table lists the values that can be given for the **PosNbarCodes3** resource for USB-attached scanners only.

Table 21-19. Valid Values for PosNbarCodes3 and PosNbarCodes4 Resources (USB Scanners Only)

Resource Value	Bar Code Types	Supported On
PosLGROUP_CODE_128_ITF (0x00010008)	Code 128, Interleaved 2-of-5	USB
PosLGROUP_CODE_93 (0x00020000)	Code 93	USB
PosLGROUP_CODE_93_ITF (0x00020008)	Code 93, Interleaved 2-of-5	USB
PosLGROUP_CODE_93_CODE_128 (0x00030000)	Code 93, Code 128	USB
PosLGROUP_CODE_39_ITF (0x00040008)	Code 39, Interleaved 2-of-5	USB
PosLGROUP_CODE_39_CODE_128 (0x00050000)	Code 39, Code 128	USB
PosLGROUP_CODE_39_CODE_93 (0x00060000)	Code 39, Code 93	USB
PosLGROUP_CODABAR (0x00080000)	Codabar	USB
PosLGROUP_CODABAR_ITF (0x00080008)	Codabar, Interleaved 2-of-5	USB
PosLGROUP_CODABAR_CODE_128 (0x00090000)	Codabar, Code 128	USB
PosLGROUP_CODABAR_CODE_93 (0x000A0000)	Codabar, Code 93	USB
PosLGROUP_CODABAR_CODE_39 (0x000C0000)	Codabar, Code 39	USB
PosLGROUP_UCC_EAN_128 (0x00100000)	UCC/EAN-128	USB
PosLGROUP_UCC_EAN_128_ITF (0x00100008)	UCC/EAN-128, Interleaved 2-of-5	USB
PosLGROUP_UCC_EAN_128_CODE_128 (0x00110000)	UCC/EAN-128, Code 128	USB
PosLGROUP_UCC_EAN_128_CODE_93 (0x00120000)	UCC/EAN-128, Code 93	USB
PosLGROUP_UCC_EAN_128_CODE_39 (0x00140000)	UCC/EAN-128, Code 39	USB
PosLGROUP_UCC_EAN_128_CODABAR (0x00180000)	UCC/EAN-128, Codabar	USB

See “Scanner Model Identifiers” on page 21-62 for descriptions of the terms shown in the “Supported On” column.

## PosNbarCodes4

<b>Type</b>	long
<b>Default</b>	(device dependent)
<b>Access</b>	CG

The **PosNbarCodes4** resource is one of four resources that determines what combination of bar code types a scanner will recognize. When configured in a specific mode, a scanner will recognize and return all bar code types that are associated with that mode.

The valid and default values for this resource are the same as for the resource, “**PosNbarCodes3**” on page 21-65.

## PosNbarCodeProgramming

<b>Type</b>	long
<b>Default</b>	PosENABLE
<b>Access</b>	CG

The **PosNbarCodeProgramming** resource controls whether or not the scanner can be programmed using the manufacturer-supplied programming bar codes.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	Scanner cannot be programmed using bar codes.
<b>PosENABLE (0x01)</b>	Scanner can be programmed using bar codes.

This resource is supported only by the following scanners:

- IBM USB Scanner Interface Specification

## PosNbeepFreq

<b>Type</b>	int
<b>Default</b>	PosHIGH
<b>Access</b>	CG

The **PosNbeepFreq** resource specifies the frequency of the tone that the beeper makes upon a successful read.

This resource can have the following values:

<b>PosLOWEST (0x00)</b>	The beep of the scanner has a very low frequency.
-------------------------	---

**Note:** For all scanners that support this resource, except the IBM 4686 Retail Point of Sale Scanner Models 1 and 2, the frequency represented by PosLOWEST is equal to the frequency represented by PosLOW.

<b>PosLOW (0x01)</b>	The beep of the scanner has a low frequency.
<b>PosMEDIUM (0x02)</b>	The beep of the scanner has a medium frequency.
<b>PosHIGH (0x03)</b>	The beep of the scanner has a high frequency.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models)
- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM USB Scanner Interface Specification

## PosNbeepLength

<b>Type</b>	int
<b>Default</b>	PosSHORT
<b>Access</b>	CG

The **PosNbeepLength** resource specifies the duration of the tone that the beeper makes upon a successful read.

This resource can have the following values:

<b>PosSHORT (0x00)</b>	Use the shortest time value for the beeper duration.
<b>PosMEDLONG (0x01)</b>	Use a medium time value for the beeper duration.
<b>PosLONG (0x02)</b>	Use the longest time value for the beeper duration.

The duration of the tone ranges from 80 to 160 milliseconds.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models)
- IBM USB Scanner Interface Specification

## PosNbeepState

<b>Type</b>	int
<b>Default</b>	PosENABLE
<b>Access</b>	CGS

The **PosNbeepState** resource controls whether the beeper is enabled or disabled. When the beeper is enabled, the scanner sounds a tone when a bar code is successfully read. When it is disabled, the scanner does not sound a tone when a bar code is successfully read.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	The beeper does not sound on a successful read.
<b>PosENABLE (0x01)</b>	The beeper sounds on a successful read.

This resource is supported by all scanners.

## PosNbeepVolume

<b>Type</b>	int
<b>Default</b>	PosHIGH
<b>Access</b>	CG

The **PosNbeepVolume** resource specifies the volume of the tone that the beeper makes.

This resource can have the following values:

<b>PosLOWEST (0x00)</b>	The volume of the beep is set at its lowest level.
-------------------------	--

**Note:** For the IBM 4696 Point of Sale Scanner Scale the IBM 4697 Point of Sale Scanner, and the IBM 4698 Point of Sale Scanner, the volume represented by PosLOWEST is equal to the volume represented by PosLOW.

<b>PosLOW (0x01)</b>	The volume of the beep is set at a low level.
<b>PosMEDIUM (0x02)</b>	The volume of the beep is set at a medium level.
<b>PosHIGH (0x03)</b>	The volume of the beep is set at its highest level.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models)
- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNblinkLength

<b>Type</b>	int
<b>Default</b>	PosLONG
<b>Access</b>	CG

The **PosNblinkLength** resource controls the length of time that the scanner indicator light, if it is equipped with one, stays on after each valid bar code read.

This resource can have the following values:

<b>PosSHORT (0x00)</b>	Use the shortest time value for the scanner indicator light.
<b>PosMEDLONG (0x01)</b>	Use a moderate time value for the scanner indicator light.
<b>PosLONG (0x02)</b>	Use the longest time value for the scanner indicator light.

The duration of the blink length ranges from 500 to 1000 milliseconds.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models)
- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNblockReadMode

<b>Type</b>	int
<b>Default</b>	1
<b>Access</b>	CG

The **PosNblockReadMode** resource controls whether block read mode is enabled for the scanner. In this mode, a group of bar codes is considered to be a single block of data. A single block of data is returned to the application.

This resource can have the following values:

<b>1</b>	Scanner is set for one block read operating mode.
<b>2</b>	Scanner is set for two block read operating mode.
<b>3</b>	Scanner is set for three block read operating mode.

**Note:** It is not possible to enable a multiple block read operating mode and double touch mode at the same time.

This resource is supported only by the following scanners:

- Hand-Held Bar Code Reader Model 1
- Hand-Held Bar Code Reader Model 2

- IBM 4685 Hand-Held Bar Code Reader Model 1
- IBM 4685 Hand-Held Bar Code Reader Model L01

## PosNblock1Type

<b>Type</b>	long
<b>Default</b>	PosLABEL_NO_CHECK
<b>Access</b>	CG

The **PosNblock1Type** resource is used to instruct the scanner to perform type checking on the first bar code in a block when block read mode is active. It specifies the type of bar code that must be in the first block.

The following table lists the valid values for this resource.

Table 21-20. Valid Values for PosNblock1Type Resource

Resource Value	Bar Code Type	Supported On
PosLABEL_NO_CHECK (0x00000000)	Not checked	HHBCR, 4685
PosLABEL_CODABAR (0x00080000)	Codabar	HHBCR, 4685
PosLABEL_CODE_39 (0x00040000)	Code 39	HHBCR, 4685
PosLABEL_EAN_8 (0x00002000)	EAN-8	HHBCR, 4685
PosLABEL_EAN_13 (0x00001000)	EAN-13	HHBCR, 4685
PosLABEL_INT_2_OF_5 (0x00000008)	Interleaved 2 of 5	HHBCR, 4685
PosLABEL_STD_2_OF_5 (0x00000004)	Standard 2 of 5	HHBCR, 4685
PosLABEL_UPC_A (0x00008000)	UPC-A	HHBCR, 4685
PosLABEL_UPC_D3 (0x00000200)	UPC-D3	HHBCR, 4685
PosLABEL_UPC_E (0x00004000)	UPC-E	HHBCR, 4685
PosLABEL_CODE_93 (0x00020000)	Code 93	HHBCR-2, 4685
PosLABEL_CODE_128 (0x00010000)	Code 128	HHBCR-2, 4685
PosLABEL_EAN_8_PLUS_2 (0x00002002)	EAN-8 with a 2-digit supplemental	HHBCR-2, 4685
PosLABEL_EAN_8_PLUS_5 (0x00002001)	EAN-8 with a 5-digit supplemental	HHBCR-2, 4685
PosLABEL_EAN_13_PLUS_2 (0x00001002)	EAN-13 with a 2-digit supplemental	HHBCR-2, 4685
PosLABEL_EAN_13_PLUS_5 (0x00001001)	EAN-13 with a 5-digit supplemental	HHBCR-2, 4685
PosLABEL_UPC_A_PLUS_2 (0x00008002)	UPC-A with a 2-digit supplemental	HHBCR-2, 4685
PosLABEL_UPC_A_PLUS_5 (0x00008001)	UPC-A with a 5-digit supplemental	HHBCR-2, 4685
PosLABEL_UPC_E_PLUS_2 (0x00004002)	UPC-E with a 2-digit supplemental	HHBCR-2, 4685
PosLABEL_UPC_E_PLUS_5 (0x00004001)	UPC-E with a 5-digit supplemental	HHBCR-2, 4685

### Notes:

1. Only single block reads are allowed for UPC-D3 bar codes and UPC/EAN bar codes with supplementals.
2. When this resource is used for single-block reads, the scanner will recognize only the specified bar code type.

See "Scanner Model Identifiers" on page 21-62 for descriptions of the terms shown in the "Supported On" column.

## PosNblock2Type

<b>Type</b>	long
<b>Default</b>	PosLABEL_NO_CHECK
<b>Access</b>	CG

The **PosNblock2Type** resource is used to instruct the scanner to perform type checking on the second bar code in a block when block read mode is active. It specifies the type of bar code that should be in the second block.

The following table lists the valid values for this resource.

Table 21-21. Valid Values for PosNblock2Type Resource

Resource Value	Bar Code Type	Supported On
PosLABEL_NO_CHECK (0x00000000)	Not checked	HHBCR, 4685
PosLABEL_CODABAR (0x00080000)	Codabar	HHBCR, 4685
PosLABEL_CODE_39 (0x00040000)	Code 39	HHBCR, 4685
PosLABEL_EAN_8 (0x00002000)	EAN-8	HHBCR, 4685
PosLABEL_EAN_13 (0x00001000)	EAN-13	HHBCR, 4685
PosLABEL_INT_2_OF_5 (0x00000008)	Interleaved 2 of 5	HHBCR, 4685
PosLABEL_STD_2_OF_5 (0x00000004)	Standard 2 of 5	HHBCR, 4685
PosLABEL_UPC_A (0x00008000)	UPC-A	HHBCR, 4685
PosLABEL_UPC_E (0x00004000)	UPC-E	HHBCR, 4685
PosLABEL_CODE_93 (0x00020000)	Code 93	HHBCR-2, 4685
PosLABEL_CODE_128 (0x00010000)	Code 128	HHBCR-2, 4685

See “Scanner Model Identifiers” on page 21-62 for descriptions of the terms shown in the “Supported On” column.

## PosNblock3Type

<b>Type</b>	long
<b>Default</b>	PosLABEL_NO_CHECK
<b>Access</b>	CG

The **PosNblock3Type** resource is used to instruct the scanner to perform type checking on the third bar code in a block when block read mode is active. It specifies the type of bar code that should be in the third block.

The following table lists the valid values for this resource.

Table 21-22. Valid Values for PosNblock3Type Resource

Resource Value	Bar Code Type	Supported On
PosLABEL_NO_CHECK (0x00000000)	First block not checked	HHBCR, 4685
PosLABEL_CODABAR (0x00080000)	Codabar	HHBCR, 4685
PosLABEL_CODE_39 (0x00040000)	Code 39	HHBCR, 4685
PosLABEL_EAN_8 (0x00002000)	EAN-8	HHBCR, 4685
PosLABEL_EAN_13 (0x00001000)	EAN-13	HHBCR, 4685

Table 21-22. Valid Values for PosNblock3Type Resource (continued)

Resource Value	Bar Code Type	Supported On
PosLABEL_INT_2_OF_5 (0x00000008)	Interleaved 2 of 5	HHBCR, 4685
PosLABEL_STD_2_OF_5 (0x00000004)	Standard 2 of 5	HHBCR, 4685
PosLABEL_UPC_A (0x00008000)	UPC-A	HHBCR, 4685
PosLABEL_UPC_E (0x00004000)	UPC-E	HHBCR, 4685
PosLABEL_CODE_93 (0x00020000)	Code 93	HHBCR-2, 4685
PosLABEL_CODE_128 (0x00010000)	Code 128	HHBCR-2, 4685

See “Scanner Model Identifiers” on page 21-62 for descriptions of the terms shown in the “Supported On” column.

## PosNbVolSwitchState

<b>Type</b>	int
<b>Default</b>	PosENABLE
<b>Access</b>	CGS

The **PosNbVolSwitchState** resource controls whether the beeper volume switch is enabled or disabled. The beeper volume can be controlled by a tone volume switch on the scanner operator’s panel. The volume set by this switch is temporary and is lost if the scanner loses power. The switch can be disabled in situations in which store personnel are not authorized to alter the configured volume.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	The beeper volume switch is disabled.
<b>PosENABLE (0x01)</b>	The beeper volume switch is enabled.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNcheckModulo

<b>Type</b>	int
<b>Default</b>	PosDISABLE
<b>Access</b>	CG

The **PosNcheckModulo** resource controls whether or not the scanner checks the modulo byte of a bar code to ensure that it is correct before returning the bar code to an application.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	The scanner does not perform modulo byte.
<b>PosENABLE (0x01)</b>	The scanner does perform modulo byte.

This resource is only supported by the following scanners:

- Hand-Held Bar Code Reader Model 1
- Hand-Held bar Code Reader Model 2

- IBM 4685 Hand-Held Bar Code Reader Model 1
- IBM 4685 Hand-Held Bar Code Reader Model L01

## PosNcode128ScansPerRead

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNcode128ScansPerRead** resource controls the minimum number of scans performed for Code 128 labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the default value should be used.

This resource is supported only by the IBM 4698 Point of Sale Scanner Models 1 and 2.

## PosNcode39ScansPerRead

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNcode39ScansPerRead** resource controls the minimum number of scans performed for Code 39 labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the default value should be used.

This resource is supported only by the IBM 4698 Point of Sale Scanner Models 1 and 2.

## PosNdecodeAlgorithm

<b>Type</b>	int
<b>Default</b>	PosENABLE
<b>Access</b>	CG

The **PosNdecodeAlgorithm** resource controls the use of *The Edge*\*\* decode algorithms. These decode algorithms use a complex set of tests to assemble bar code data from damaged or truncated labels. These techniques also give faster read performance on normal bar codes. These decode algorithms can be disabled. The traditional scanning algorithm, an improved version of the one used in the IBM 4687 Point of Sale Scanner Model 2, is used when *The Edge* algorithms are disabled.

This resource can have the following values:

**PosDISABLE (0x00)** The traditional scanning algorithm is used.

**PosENABLE (0x01)**                      *The Edge* decode algorithms are used.

**Note:** *The Edge* algorithms are disabled on the IBM 4697 Point of Sale Scanner when the scanner is configured to read industrial codes (when the **PosNbarCodes2** resource is something other than PosLGROUP\_NONE).

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2

## PosNdReadTimeout

<b>Type</b>	int
<b>Default</b>	PosSHORT
<b>Access</b>	CG

The **PosNdReadTimeout** resource controls the length of the double-read timeout. Most scanners decode and recognize a bar code label several times as the bar code is passed through the scanning region. To prevent a scanner from returning data from the same bar code several times, scanners are programmed with a double-read timeout. The double-read timeout is the length of time that the scanner waits before returning the same bar code data twice.

This resource can have the following values:

<b>PosSHORT (0x00)</b>	Use the shortest time value for the double-read timeout.
<b>PosMEDLONG (0x01)</b>	Use a moderate time value for the double-read timeout.
<b>PosLONG (0x02)</b>	Use the longest time value for the double-read timeout.

The duration of the double-read timeout ranges from approximately 500 to 1000 milliseconds.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models)
- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNdTouchMode

<b>Type</b>	int
<b>Default</b>	PosDISABLE
<b>Access</b>	CG

The **PosNdTouchMode** resource controls the state of the double-touch mode of the scanner. Double-touch mode allows scanners such as the Hand-Held Bar Code Reader to read bar codes that are bigger than the reading head.

In double-touch mode, when the reading head of the scanner is placed over the first half of the label, the scanner emits a repetitive beeping noise (if the beeper is

enabled) to indicate that the data was read. When the reading head is placed over the second half of the label, the data is sent to the system unit. Putting a scanner in double-touch mode does not prevent it from reading a bar code in a single touch.

Only bar codes such as UPC-A, EAN-13, and UPC-D3 can be read using double-touch mode. For UPC-D3 bar codes, double-touch mode is automatically enabled by the scanner and cannot be switched off.

This resource can have the following values:

**PosDISABLE (0x00)** Double-touch mode is disabled.  
**PosENABLE (0x01)** Double-touch mode is enabled.

**Note:** It is not possible to enable a multiple block read operating mode and double-touch mode at the same time.

This resource is supported only by the following scanners:

- Hand-Held Bar Code Reader Model 1
- Hand-Held Bar Code Reader Model 2
- IBM 4685 Hand-Held Bar Code Reader Model 1
- IBM 4685 Hand-Held Bar Code Reader Model L01

## PosNeAN13ScansPerRead

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNeAN13ScansPerRead** resource controls the minimum number of scans performed for EAN13 labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the number of scans per read is not specified by the application.

For the IBM 4696 Point of Sale Scanner Scale, and the IBM 4698 Point of Sale Scanner, when a value is not specified for this resource, the default value will be used. For the IBM 4697 Point of Sale Scanner, when a value is not specified for this resource, the value in the **PosNscansPerRead** resource is used.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2

## PosNeAN8ScansPerRead

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNeAN8ScansPerRead** resource controls the minimum number of scans performed for EAN8 labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the number of scans per read is not specified by the application.

For the IBM 4696 Point of Sale Scanner Scale, and the IBM 4698 Point of Sale Scanners, when a value is not specified for this resource, the default value will be used. For the IBM 4697 Point of Sale Scanner, when a value is not specified for this resource, the value in the **PosNscansPerRead** resource is used.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2

## PosNiTFLength1

<b>Type</b>	int
<b>Default</b>	0
<b>Access</b>	CG

For the scanners that support Interleaved 2-of-5 bar codes, the **PosNiTFLength1** resource can be used to specify one valid length for Interleaved 2-of-5 bar codes. This value indicates the exact length of the Interleaved 2-of-5 bar codes that the scanner will read. If an Interleaved 2-of-5 bar code is not of the correct length, then the bar code is not read by the scanner.

The value of the **PosNiTFLength1** resource must be an even number from 4 to 32. (For the IBM 1520-A02, the value must be an even number from 4 to 30.) This value specifies the exact length of the bar code.

For the IBM 4686 Retail Point of Sale Scanner Models 1 and 2, a value of 0 (zero) indicates that no length checking is to be done, and Interleaved 2-of-5 bar codes of any valid length are read. For the IBM 1520 Hand-Held Scanner Model A02, the IBM 4686 Retail Point of Sale Scanner Models 3 and 4 and the IBM 4697 Point of Sale Scanner Model 1, a value of 0 (zero) is not valid.

If the scanner is not configured to read Interleaved 2-of-5 bar codes, the value of this resource is ignored.

This resource is provided because scanners are prone to errors when reading Interleaved 2-of-5 labels. Because these labels are of variable length, it is possible for a scanner to read only part of a label, but process it as though it had read the complete label. If an application is only expecting Interleaved 2-of-5 labels of a certain length, this resource ensures that the scanner does not read any partial labels.

This resource is supported only by the following scanners:

- IBM 1520 Hand-Held Scanner Model A02
- IBM 4686 Retail Point of Sale Scanner (all models)
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNiTFLength2

<b>Type</b>	int
<b>Default</b>	0
<b>Access</b>	CG

For some flat-bed scanners that support Interleaved 2-of-5 bar codes, the **PosNiTFLength2** resource can be used to indicate a second valid length for Interleaved 2-of-5 bar codes. When reading Interleaved 2-of-5 bar codes, these scanners can only read bar codes of one or two specific lengths. The **PosNiTFLength1** and **PosNiTFLength2** resources are used to specify the Interleaved 2-of-5 lengths to be recognized and read by these scanners.

The value of the **PosNiTFLength2** resource must be an even number from 4 to 32. This value specifies the exact length of the bar code. A value of 0 (zero) indicates that only Interleaved 2-of-5 bar codes of the length specified in the **PosNiTFLength1** resource will be read by the scanner.

If the scanner is not configured to read Interleaved 2-of-5 bar codes, the value of this resource is ignored.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (Models 3 and 4)
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNiTFLengthType

<b>Type</b>	int
<b>Default</b>	PosDISCRETE
<b>Access</b>	CG

The **PosNiTFLengthType** resource allows the application to choose discrete lengths or a range of lengths for Interleaved 2-of-5 bar codes.

This resource can have the following values:

<b>PosDISCRETE (0x00)</b>	Scanner recognizes only Interleaved 2-of-5 bar codes that have the length specified in <b>PosNiTFLength1</b> or <b>PosNiTFLength2</b> .
<b>PosRANGE (0x01)</b>	Scanner recognizes Interleaved 2-of-5 bar codes that have lengths ranging from the value specified in <b>PosNiTFLength1</b> to the value specified in <b>PosNiTFLength2</b> inclusive.

If the scanner is not configured to read Interleaved 2-of-5 bar codes or only **PosNiTFLength1** is specified, the value of this resource is ignored.

This resource is supported only by the following scanners:  
IBM USB Scanner Interface Specification

## PosNiTFScansPerRead

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNiTFScansPerRead** resource controls the minimum number of scans performed for Interleaved 2-of-5 labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the default value should be used.

This resource is supported only by the IBM 4698 Point of Sale Scanner Models 1 and 2.

## PosNjANTwoLabelDecode

<b>Type</b>	int
<b>Default</b>	PosDISABLE
<b>Access</b>	CG

The **PosNjANTwoLabelDecode** resource controls whether or not the scanner returns EAN/JAN-13 double labels. EAN/JAN-13 double labels are identified by the first two (prefix) digits of each label code. The valid prefix digits are specified in the **PosNtwoLabelFlagPair1**, the **PosNtwoLabelFlagPair2**, the **PosNtwoLabelFlagPair3**, and the **PosNtwoLabelFlagPair4** resources. See “PosNtwoLabelFlagPair1” on page 21-83, “PosNtwoLabelFlagPair2” on page 21-84, “PosNtwoLabelFlagPair3” on page 21-85, and “PosNtwoLabelFlagPair4” on page 21-86 for information concerning the label prefix pairs for EAN/JAN-13 double labels.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	Double EAN/JAN-13 labels are not returned by the scanner.
<b>PosENABLE (0x01)</b>	Double EAN/JAN-13 labels are returned by the scanner.

This resource is supported only by the following scanners:

- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNLabelsQueued

<b>Type</b>	int
<b>Default</b>	n/a
<b>Access</b>	G

The **PosNLabelsQueued** resource returns the number of labels currently queued for a particular device connection. The application can get these labels by issuing the *PosRead()* subroutine call for the device connection. This resource cannot be set.

This resource is supported on all scanners.

## PosNlaserSwitchState

<b>Type</b>	int
<b>Default</b>	PosENABLE
<b>Access</b>	CG

The **PosNlaserSwitchState** resource controls whether the laser power switch on the scanner unit is enabled or disabled.

When a scanner is on, its motor is running and its laser is active. When a scanner is off, its motor is not running and its laser is inactive. When the laser power switch is enabled, it can be used to turn the scanner off. When the laser power switch is disabled, it cannot be used to turn the scanner off. The laser power switch can always be used to turn the scanner on.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	The laser power switch can only be used to turn the scanner motor and laser on.
<b>PosENABLE (0x01)</b>	The laser power switch can be used to turn the scanner motor and laser both on and off.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models).
- IBM USB Scanner Interface Specification

## PosNlaserTimeout

<b>Type</b>	int
<b>Default</b>	15
<b>Access</b>	CG

The **PosNlaserTimeout** resource specifies the length of the period of inactivity that causes a laser scanner to turn off its laser.

This resource can have the following values:

<b>0</b>	The laser always stays on (IBM 4686 Retail Point of Sale Scanner, all models) or turns off after 15 minutes of inactivity (IBM 4696 Point of Sale Scanner Scale, IBM 4697 Point of Sale Scanner, and IBM 4698 Point of Sale Scanner Model 1 and 2).
<b>5</b>	The laser turns off after 5 minutes of inactivity.
<b>10</b>	The laser turns off after 10 minutes of inactivity.
<b>15</b>	The laser turns off after 15 minutes of inactivity.

**Note:** For motor-driven laser scanners, the motor is always on if the laser is on.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models)

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNmotorTimeout

<b>Type</b>	int
<b>Default</b>	15
<b>Access</b>	CG

The **PosNmotorTimeout** resource specifies the length of the period of inactivity that causes a motorized laser scanner to turn off its motor.

This resource can have the following values:

- 0** The motor is always on (IBM 4686 Retail Point of Sale Scanner, all models) or turns off after 60 minutes of inactivity (IBM 4696 Point of Sale Scanner Scale, IBM 4697 Point of Sale Scanner, and IBM 4698 Point of Sale Scanner Model 1 and 2).
- 5** The motor turns off after 5 minutes of inactivity.
- 10** The motor turns off after 10 minutes of inactivity.
- 15** The motor turns off after 15 minutes of inactivity.
- 30** The motor turns off after 30 minutes of inactivity.
- 60** The motor turns off after 60 minutes of inactivity.

**Note:** For motor-driven laser scanners, the motor is always on if the laser is on.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models)
- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNqueueAllLabels

<b>Type</b>	int
<b>Default</b>	PosENABLE
<b>Access</b>	CGS

The **PosNqueueAllLabels** resource controls whether the scanner keeps labels that are read while the scanner is acquired but locked.

Some scanners will return labels while they are locked. This resource allows the application to choose to have these labels ignored or placed in the label data queue. If the labels are queued, the application can still determine that they were read while the scanner was locked. See “Reading Scanner Data” on page 16-4 and “Processing Unexpected Scanner Data” on page 16-9 for information about how to determine that the scanner was locked when the label was read.

This resource can have the following values:

### **PosDISABLE (0x00)**

Ignore any label data that is read while the scanner is acquired and locked.

**PosENABLE (0x01)**

Place label data that is received while the scanner is acquired and locked in the label data queue.

This resource is supported on all scanners.

**PosNscansPerRead**

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNscansPerRead** resource specifies how many scans of a bar code the scanner performs before deciding that the bar code is valid and returning it to the application. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 1 to 4. Any value outside of this range is not valid. For the IBM 4697 Point of Sale Scanner, this value will only apply when the code-specific scans-per-read is unspecified. The code-specific scans-per-read are specified by the following resources:

- PosNeAN8ScansPerRead
- PosNeAN13ScansPerRead
- PosNstoreScansPerRead
- PosNuPCAScansPerRead
- PosNuPCDScansPerRead
- PosNuPCEScansPerRead

**Note:** For the IBM 4697 Point of Sale Scanner, two scans for each read are done for UPC-E and EAN-8 bar codes when either 1 or 2 is specified.

This resource is supported only by the following scanners:

- IBM 4686 Retail Point of Sale Scanner (all models)
- IBM 4697 Point of Sale Scanner Model 1

**PosNstoreScansPerRead**

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNstoreScansPerRead** resource controls the minimum number of scans performed for in-store labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the number of scans per read is not specified by the application.

For the IBM 4696 Point of Sale Scanner Scale and the IBM 4698 Point of Sale Scanners, when a value is not specified for this resource, the default value will be used. For the IBM 4697 Point of Sale Scanner, when a value is not specified for this resource, the value in the **PosNscansPerRead** resource is used.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNsupplementals

<b>Type</b>	int
<b>Default</b>	PosNO_SUPPLEMENTALS
<b>Access</b>	CG

The **PosNsupplementals** resource controls whether the scanner recognizes bar codes with supplementals. When enabled, supplementals are **optional** for all specified symbologies.

The following table lists the valid values for this resource.

Table 21-23. Valid Values for PosNsupplementals Resource

Resource Value	Description
PosNO_SUPPLEMENTALS (0x00000000)	No supplementals
PosPLUS_2_SUPPLEMENTALS (0x00000001)	2-digit supplementals
PosPLUS_5_SUPPLEMENTALS (0x00000002)	5-digit supplementals
PosPLUS_2_5_SUPPLEMENTALS (0x00000003)	2- and 5-digit supplementals
PosPLUS_CODE128_SUPPLEMENTALS (0x00000004)	Code 128 supplementals
PosPLUS_2_CODE128_SUPPLEMENTALS (0x00000005)	2-digit and Code 128 supplementals
PosPLUS_5_CODE128_SUPPLEMENTALS (0x00000006)	5-digit and Code 128 supplementals
PosALL_SUPPLEMENTALS (0x00000007)	2-digit, 5-digit and Code 128 supplementals

This resource is supported only by the IBM USB Scanner Interface.

## PosNtransmitCheckDigit

<b>Type</b>	int
<b>Default</b>	PosUPCE_UPCA_CHECK_DIGIT
<b>Access</b>	CG

The **PosNtransmitCheckDigit** resource controls whether or not check digits should be transmitted for UPC-A, UPC-E, Code 39, or Interleaved 2-of-5 (ITF) barcodes.

The following table lists the valid values for this resource:

Table 21-24. Valid Values for PosNtransmitCheckDigit Resource

Resource Value	Check Digit Transmitted for
PosNO_CHECK_DIGITS (0x00000000)	None
PosUPCA_CHECK_DIGIT (0x00000002)	UPC-A Only
PosUPCE_CHECK_DIGIT (0x00000004)	UPC-E Only
PosUPCE_UPCA_CHECK_DIGIT (0x00000006)	UPC-E Only

Table 21-24. Valid Values for PosNtransmitCheckDigit Resource (continued)

Resource Value	Check Digit Transmitted for
PosCODE39_CHECK_DIGIT (0x00000008)	Code 39 Only
PosCODE39_UPCA_CHECK_DIGIT (0x0000000A)	UPC-A Only
PosCODE39_UPCE_CHECK_DIGIT (0x0000000C)	UPC-E Only
PosCODE39_UPCE_UPCA_CHECK_DIGIT (0x0000000E)	Code 39, UPC-E, UPC-A
PosITF_CHECK_DIGIT (0x00000010)	ITF Only
PosITF_UPCA_CHECK_DIGIT (0x00000012)	ITF and UPC-A
PosITF_UPCE_CHECK_DIGIT (0x00000014)	ITF and UPC-E
PosITF_UPCE_UPCA_CHECK_DIGIT (0x00000016)	ITF, UPC-E, UPC-A
PosITF_CODE39_CHECK_DIGIT (0x00000018)	ITF and Code 39
PosITF_CODE39_UPCA_CHECK_DIGIT (0x0000001A)	ITF, UPC-A, Code 39
PosITF_CODE39_UPCE_CHECK_DIGIT (0x0000001C)	ITF, UPC-E, Code 39
PosALL_CHECK_DIGITS (0x00000001)	All

This resource is supported only by the IBM USB Scanner Interface.

## PosNtwoLabelFlagPair1

<b>Type</b>	short
<b>Default</b>	0x2122
<b>Access</b>	CG

The **PosNtwoLabelFlagPair1** resource specifies a pair of two-digit flag values used to identify EAN/JAN-13 double labels. This resource is only used when the **PosNjANTwoLabelDecode** resource is PosENABLE.

If the scanner reads an EAN/JAN-13 code which starts with one of the specified two-digit flag values, it will not beep or send the code until another EAN/JAN-13 code which starts with the other specified two-digit value is read. For example, when **PosNtwoLabelFlagPair1** is specified as 0x2122 (default), if Code1 = 21nnnn nnnnnn c is scanned, nothing happens until Code2 = 22nnnn nnnnnn c is scanned (n represents a bar code digit and c represents the bar code check digit). When Code2 is scanned, the scanner beeps once and sends Code1 and Code2 in standard EAN/JAN-13 format in separate scanner messages. The order in which the bar codes are scanned does not matter, although the bar codes will be sent to the application in the order specified by the flag pair. For instance, in the example above, if Code2 was scanned before Code1, the bar codes would still be recognized as a EAN/JAN-13 double label but Code1 would be received by the application before Code2.

Each flag value consists of two digits in the range 0x0 to 0x9. The two flag values in the pair must not be the same unless both values are 0x0. If the flag pair is specified as 0x0000, the default value for the flag pair is used. For example, the following flag pairs are not valid:

<b>0x2121</b>	both values are the same
<b>0xA121</b>	invalid digit
<b>0x211B</b>	invalid digit

There are four flag pairs defined when two-label decoding is enabled:

- **PosNtwoLabelFlagPair1**
- **PosNtwoLabelFlagPair2**
- **PosNtwoLabelFlagPair3**
- **PosNtwoLabelFlagPair4**

In addition to the restrictions placed on a single flag pair, the first two-digit flag value in each pair cannot be the same as the second two-digit flag value in any of the other flag pairs. For example, 0x2122, 0x4950, 0x2128, and 0x2822 would not be a valid flag pair combination because 0x28 appears as both a first flag value and a second flag value. It is valid, however, for two or more flag pairs to be the same. For example, 0x2122, 0x4950, 0x4950, and 0x2122 are a valid flag pair combination.

This resource is supported only by the following scanners:

- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNtwoLabelFlagPair2

<b>Type</b>	short
<b>Default</b>	0x2128
<b>Access</b>	CG

The **PosNtwoLabelFlagPair2** resource specifies a pair of two-digit flag values used to identify EAN/JAN-13 double labels. This resource is only used when the **PosNjANTwoLabelDecode** resource is PosENABLE.

If the scanner reads an EAN/JAN-13 code which starts with one of the specified two-digit flag values, it will not beep or send the code until another EAN/JAN-13 code which starts with the other specified two-digit value is read. For example, when **PosNtwoLabelFlagPair2** is specified as 0x2128 (default), if Code1 = 28nnnn nnnnn c is scanned, nothing happens until Code2 = 21nnnn nnnnn c is scanned. (n represents a bar code digit and c represents the bar code check digit). When Code2 is scanned, the scanner beeps once and sends Code1 and Code2 in standard EAN/JAN-13 format in separate scanner messages. The order in which the bar codes are scanned does not matter, although the bar codes will be sent to the application in the order specified by the flag pair. For instance, in the example above, if Code2 was scanned before Code1, the bar codes would still be recognized as a EAN/JAN-13 double label but Code1 would be received by the application before Code2.

Each flag value consists of two digits in the range 0x0 to 0x9. The two flag values in the pair must not be the same unless both values are 0x0. If the flag pair is specified as 0x0000, the default value for the flag pair is used. For example, the following flag pairs are not valid:

<b>0x2121</b>	both values are the same
<b>0xA121</b>	invalid digit
<b>0x211B</b>	invalid digit

There are four flag pairs defined when two-label decoding is enabled:

- **PosNtwoLabelFlagPair1**
- **PosNtwoLabelFlagPair2**

- **PosNtwoLabelFlagPair3**
- **PosNtwoLabelFlagPair4**

In addition to the restrictions placed on a single flag pair, the first two-digit flag value in each pair cannot be the same as the second two-digit flag value in any of the other flag pairs. For example, 0x2122, 0x4950, 0x2128, and 0x2822 would not be a valid flag pair combination because 0x28 appears as both a first flag value and a second flag value. It is valid, however, for two or more flag pairs to be the same. For example, 0x2122, 0x4950, 0x4950, and 0x2122 are a valid flag pair combination.

This resource is supported only by the following scanners:

- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

### PosNtwoLabelFlagPair3

<b>Type</b>	short
<b>Default</b>	0x2129
<b>Access</b>	CG

The **PosNtwoLabelFlagPair3** resource specifies a pair of two-digit flag values used to identify EAN/JAN-13 double labels. This resource is only used when the **PosNjANTwoLabelDecode** resource is PosENABLE.

If the scanner reads an EAN/JAN-13 code which starts with one of the specified two-digit flag values, it will not beep or send the code until another EAN/JAN-13 code which starts with the other specified two-digit value is read. For example, when **PosNtwoLabelFlagPair3** is specified as 0x2129 (default), if Code1 = 21nnnn nnnnn c is scanned, nothing happens until Code2 = 29nnnn nnnnn c is scanned. (n represents a bar code digit and c represents the bar code check digit). When Code2 is scanned, the scanner beeps once and sends both Code1 and Code2 in standard EAN/JAN-13 format in separate scanner messages. The order in which the bar codes are scanned does not matter, although the bar codes will be sent to the application in the order specified by the flag pair. For instance, in the example above, if Code2 was scanned before Code1, the bar codes would still be recognized as a EAN/JAN-13 double label but Code1 would be received by the application before Code2.

Each flag value consists of two digits in the range 0x0 to 0x9. The two flag values in the pair must not be the same unless both values are 0x0. If the flag pair is specified as 0x0000, the default value for the flag pair is used. For example, the following flag pairs are not valid:

**0x2121**  
both values are the same

**0xA121**  
invalid digit

**0x211B**  
invalid digit

There are four flag pairs defined when two-label decoding is enabled:

- **PosNtwoLabelFlagPair1**
- **PosNtwoLabelFlagPair2**

- **PosNtwoLabelFlagPair3**
- **PosNtwoLabelFlagPair4**

In addition to the restrictions placed on a single flag pair, the first two-digit flag value in each pair cannot be the same as the second two-digit flag value in any of the other flag pairs. For example, 0x2122, 0x4950, 0x2128, and 0x2822 would not be a valid flag pair combination because 0x28 appears as both a first flag value and a second flag value. It is valid, however, for two or more flag pairs to be the same. For example, 0x2122, 0x4950, 0x4950, and 0x2122 are a valid flag pair combination.

This resource is supported only by the following scanners:

- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNtwoLabelFlagPair4

<b>Type</b>	short
<b>Default</b>	0x2122
<b>Access</b>	CG

The **PosNtwoLabelFlagPair4** resource specifies a pair of two-digit flag values used to identify EAN/JAN-13 double labels. This resource is only used when the **PosNjANTwoLabelDecode** resource is PosENABLE.

If the scanner reads an EAN/JAN-13 code which starts with one of the specified two-digit flag values, it will not beep or send the code until another EAN/JAN-13 code which starts with the other specified two-digit value is read. For example, when **PosNtwoLabelFlagPair1** is specified as 0x2122 (default), if Code1 = 21nnnn nnnnn c is scanned, nothing happens until Code2 = 22nnnn nnnnn c is scanned. (n represents a bar code digit and c represents the bar code check digit). When Code2 is scanned, the scanner beeps once and sends both Code1 and Code2 in standard EAN/JAN-13 format in separate scanner messages. The order in which the bar codes are scanned does not matter, although the bar codes will be sent to the application in the order specified by the flag pair. For instance, in the example above, if Code2 was scanned before Code1, the bar codes would still be recognized as a EAN/JAN-13 double label but Code1 would be received by the application before Code2.

Each flag value consists of two digits in the range 0x0 to 0x9. The two flag values in the pair must not be the same unless both values are 0x0. If the flag pair is specified as 0x0000, the default value for the flag pair is used. For example, the following flag pairs are not valid:

**0x2121**  
both values are the same

**0xA121**  
invalid digit

**0x211B**  
invalid digit

There are four flag pairs defined when two-label decoding is enabled:

- **PosNtwoLabelFlagPair1**
- **PosNtwoLabelFlagPair2**

- **PosNtwoLabelFlagPair3**
- **PosNtwoLabelFlagPair4**

In addition to the restrictions placed on a single flag pair, the first two-digit flag value in each pair cannot be the same as the second two-digit flag value in any of the other flag pairs. For example, 0x2122, 0x4950, 0x2128, and 0x2822 would not be a valid flag pair combination because 0x28 appears as both a first flag value and a second flag value. It is valid, however, for two or more flag pairs to be the same. For example, 0x2122, 0x4950, 0x4950, and 0x2122 are a valid flag pair combination.

This resource is supported only by the following scanners:

- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNuPCAScansPerRead

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNuPCAScansPerRead** resource controls the minimum number of scans performed for UPC-A labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the number of scans per read is not specified by the application.

For the IBM 4696 Point of Sale Scanner Scale and the IBM 4698 Point of Sale Scanners, when a value is not specified for this resource, the default value will be used. For the IBM 4697 Point of Sale Scanner, when a value is not specified for this resource, the value in the **PosNscansPerRead** resource is used.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2

## PosNuPCDScansPerRead

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNuPCDScansPerRead** resource controls the minimum number of scans performed for UPC-D labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the number of scans per read is not specified by the application.

For the IBM 4696 Point of Sale Scanner Scale and the IBM 4698 Point of Sale Scanners, when a value is not specified for this resource, the default value will be used. For the IBM 4697 Point of Sale Scanner, when a value is not specified for this resource, the value in the **PosNscansPerRead** resource is used.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2

## PosNuPCEScansPerRead

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CG

The **PosNuPCEScansPerRead** resource controls the minimum number of scans performed for UPC-E labels. This is the number of scans for a single pass of an item over the scanner window.

The value of this resource has a range from 0 (zero) to 4. Any value outside of this range is not valid. A value of 0 (zero) indicates that the number of scans per read is not specified by the application.

For the IBM 4696 Point of Sale Scanner Scale and the IBM 4698 Point of Sale Scanners, when a value is not specified for this resource, the default value will be used. For the IBM 4697 Point of Sale Scanner, when a value is not specified for this resource, the value in the **PosNscansPerRead**.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2

## PosNuPCExpansion

<b>Type</b>	int
<b>Default</b>	0
<b>Access</b>	CG

The **PosNuPCExpansion** resource controls the report format for UPC-A and UPC-E labels. UPC-A, UPC-E, and EAN-13 are all part of the same numbering system. It is possible to have the scanner report all of these codes in a uniform format.

UPC-A is a 12-digit subset of EAN-13. The scanner can add a leading 0 (zero) to the UPC-A number, yielding its EAN equivalent. UPC-E is a short form of a UPC-A number. The scanner can expand UPC-E data to its UPC-A format.

This resource can have the following values:

### **PosNO\_EXPANSION (0x00)**

No expansions are performed.

**PosUPCA\_TO\_EAN13 (0x01)**

UPC-A to EAN-13 expansions are performed.

**PosUPCE\_TO\_EAN13 (0x02)**

UPC-E to EAN-13 expansions are performed.

**PosUPCA\_UPCE\_TO\_EAN13 (0x03)**

UPC-A to EAN-13 and UPC-E to EAN-13 expansions are performed.

**PosUPCE\_TO\_UPCA (0x04)**

UPC-E to UPC-A expansions are performed.

**PosUPCE\_TO\_UPCA\_TO\_EAN13 (0x05)**

UPC-E to UPC-A and UPC-A to EAN-13 expansions are performed.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosNverifyPriceChk

<b>Type</b>	int
<b>Default</b>	NO_VERIFY
<b>Access</b>	CG

The **PosNverifyPriceChk** resource controls whether the scanner verifies the price check character. UPC and EAN specifications allow for a price check character to be included in the digits encoded on in-store random weight items. The IBM 4696 Point of Sale Scanner Scale can verify the price check digit and help to guard against the poor-quality print often found on in-store labels.

This resource can have the following values:

<b>PosNO_VERIFY (0x00)</b>	Do not verify the price check character.
<b>PosVERIFY_4DIGIT (0x01)</b>	Verify price check character for 4-digit price.
<b>PosVERIFY_5DIGIT (0x02)</b>	Verify price check character for 5-digit price.

This resource is supported only by the following scanners:

- IBM 4696 Point of Sale Scanner Scale Model 1
- IBM 4697 Point of Sale Scanner Model 1
- IBM 4698 Point of Sale Scanner Models 1 and 2
- IBM USB Scanner Interface Specification

## PosTouch Resource Set

<b>Class Name</b>	PosTouch
<b>Include</b>	<pos/touch.h>

A copy of the PosTouch resource set is created every time an application opens a touch screen device by the *PosOpen()* subroutine call. The resource set associated with a touch screen connection when it is opened remains associated with it until the touch screen connection is closed. All touch screen resources except for **PosNtouchMode** can be set without first acquiring exclusive device access. However, resources specified in a resource file or as arguments to the *PosOpen* subroutine do not take effect until your application acquires exclusive access of the

device. The following table gives an overview of the resources in this set.

Table 21-25. PosTouch Resources

Name	Type	Default	Access
PosNtouchBackLightOnEvent	int	PosDISABLE	CGS
PosNtouchBrightness	int	n/a	CGS
PosNtouchClickVolume	int	PosLOW	CGS
PosNtouchContrast	int	n/a	CGS
PosNtouchEntryClick	int	PosOFF	CGS
PosNtouchExitClick	int	PosOFF	CGS
PosNtouchMaxX	int	n/a	G
PosNtouchMaxY	int	n/a	G
PosNtouchMode	int	0x01	CGS
PosNtouchScreenSaverTime	int	1800	CGS
PosNtouchToneDuration	int	2	CGS
PosNtouchToneFreq	int	PosMEDIUM	CGS
PosNtouchToneVolume	int	PosLOW	CGS

See “Resource Access Codes” on page 21-5 for descriptions of the access codes.

## PosNtouchBackLightOnEvent

<b>Type</b>	int
<b>Default</b>	PosDISABLE
<b>Access</b>	CGS

The **PosNtouchBackLightOnEvent** resource controls the generation of an event message when the screen is touched to refresh the display.

This resource can have the following values:

<b>PosDISABLE (0x00)</b>	An event message is not generated.
<b>PosENABLE (0x01)</b>	An event message is generated.

## PosNtouchBrightness

<b>Type</b>	int
<b>Default</b>	n/a
<b>Access</b>	CGS

The **PosNtouchBrightness** resource controls the brightness setting of the touch screen display on models that support programatic brightness control. The brightness can be set to 8 different levels, numbered 0 (zero) through 7.

The default brightness value for each LCD touch screen is set at the time of manufacture, and thus can vary from one display to another. Your application can determine the current brightness setting of the physical device by getting the value of the **PosNtouchBrightness** resource.

This resource can have the following values:

**0-7** LCD brightness value.

This resource is supported only by the following touch screens:

- IBM 4695 Point of Sale Distributed Touch Terminal Model 032
- IBM 4695 Point of Sale Integrated Touch Terminal Model 331
- IBM 4695 Point of Sale Integrated Touch Terminal Model 342

## PosNtouchClickVolume

<b>Type</b>	int
<b>Default</b>	PosLOW
<b>Access</b>	CGS

The **PosNtouchClickVolume** resource controls the volume of the internal speaker for the **PosNtouchEntryClick** and **PosNtouchExitClick** resources.

This resource can have the following values:

<b>PosLOW (0x01)</b>	The click volume is set at a low level.
<b>PosHIGH (0x03)</b>	The click volume is set at its highest level.

## PosNtouchContrast

<b>Type</b>	int
<b>Default</b>	n/a
<b>Access</b>	CGS

The **PosNtouchContrast** resource controls the contrast setting of the touch screen display on models that support programatic contrast control. The contrast can be set to 64 different levels, numbered 0 (zero) through 63.

The default contrast value for each LCD touch screen is set at the time of manufacture, and thus can vary from one display to another. Your application can determine the current contrast setting of the physical device by getting the value of the **PosNtouchContrast** resource.

This resource can have the following values:

**0-63** LCD contrast value.

This resource is supported only by the following touch screens:

- 4695 Point of Sale Distributed Touch Terminal Model 002
- 4695 Point of Sale Distributed Touch Terminal Model 012
- 4695 Point of Sale Distributed Touch Terminal Model 022
- 4695 Point of Sale Integrated Touch Terminal Model 201
- 4695 Point of Sale Integrated Touch Terminal Model 211
- 4695 Point of Sale Integrated Touch Terminal Model 321
- 4695 Point of Sale Integrated Touch Terminal Model 322

## PosNtouchEntryClick

<b>Type</b>	int
<b>Default</b>	PosOFF
<b>Access</b>	CGS

The **PosNtouchEntryClick** resource controls whether audible feedback is generated when the screen is touched (touch-down occurs).

This resource can have the following values:

**PosOFF (0x00)**

Audible feedback is not generated

**PosON (0x01)** Audible feedback is generated

## PosNtouchExitClick

**Type** int

**Default** PosOFF

**Access** CGS

The **PosNtouchExitClick** resource controls whether audible feedback is generated when the screen is no longer being touched (lift-off occurs).

This resource can have the following values:

**PosOFF (0x00)**

Audible feedback is not generated

**PosON (0x01)** Audible feedback is generated

## PosNtouchMaxX

**Type** int

**Default** n/a

**Access** G

The **PosNtouchMaxX** resource contains the largest x-coordinate value that will be reported to the application.

## PosNtouchMaxY

**Type** int

**Default** n/a

**Access** G

The **PosNtouchMaxY** resource contains the largest y-coordinate value that will be reported to the application.

## PosNtouchMode

**Type** int

**Default** 0x01

**Access** CGS

The **PosNtouchMode** resource determines which touch actions cause an event message to be generated.

This resource can have any combination of the following values:

**PosTOUCH\_MODE\_TRACKING (0x01)**

Touch event messages are generated when the touch display becomes

active (touch-down), when the touch coordinates change, and when the touch display becomes inactive (lift-off).

**PosTOUCH\_MODE\_ENTER (0x02)**

Touch event messages are generated when the touch display is first pressed.

**PosTOUCH\_MODE\_EXIT (0x04)**

Touch event messages are generated when the touch display is no longer pressed.

**PosNtouchScreenSaverTime**

<b>Type</b>	int
<b>Default</b>	1800
<b>Access</b>	CGS

The **PosNtouchScreenSaverTime** resource controls how many seconds the touch screen can remain inactive before the LCD back light dims.

This resource can have the following values:

<b>PosOFF (0x00)</b>	Dim the LCD back light immediately.
<b>1-65535 (0x0001-0xFFFF)</b>	Seconds of inactivity before the LCD back light dims. Values of less than 10 are not recommended and for some touch devices will have no effect.

**PosNtouchToneDuration**

<b>Type</b>	int
<b>Default</b>	2
<b>Access</b>	CGS

The **PosNtouchToneDuration** resource controls the tone duration of the internal speaker.

This resource can have the following values:

<b>PosON (0x01)</b>	On until switched off
<b>2-255</b>	0.2 - 25.5 seconds (0.10 times the specified number)

**PosNtouchToneFreq**

<b>Type</b>	int
<b>Default</b>	PosMEDIUM
<b>Access</b>	CGS

The **PosNtouchToneFreq** resource controls the tone frequency of the internal speaker.

This resource can have the following values:

<b>PosLOW (0x01)</b>	The tone frequency is set at a low level.
<b>PosMEDIUM (0x02)</b>	The tone frequency is set at a medium level.
<b>PosHIGH (0x03)</b>	The tone frequency is set at its highest level.

## PosNtouchToneVolume

<b>Type</b>	int
<b>Default</b>	PosLOW
<b>Access</b>	CGS

The **PosNtouchToneVolume** resource controls the tone volume of the internal speaker.

This resource can have the following values:

<b>PosLOW (0x01)</b>	The tone volume is set at a low level.
<b>PosHIGH (0x03)</b>	The tone volume is set at its highest level.

created on October 2, 2001

---

## **Part 4. Appendixes**

created on October 2, 2001

---

## Appendix A. Data Types and Macros

The data types and macros in this section are defined in the `pos.h` header file.

---

### Data Types

The following is a complete list of the data types that have been defined for the functions described in this book.

Type	Meaning
<b>size_t</b>	Standard C Library unsigned integer type that can represent the size of the largest data object that can be declared.
<b>ssize_t</b>	This data type is defined in the header file <code>pos.h</code> .
<b>PosArg</b>	A structure that defines a resource and its value. A list of these are passed to the application programming interface subroutine calls. This structure has the following elements: <b>name (char *)</b> The name of the resource <b>value (long)</b> The value of the resource
<b>PosArgPtr</b>	A pointer to the structure <b>PosArg</b> .
<b>POSQMSG</b>	A structure that defines a IBM Point of Sale Subsystem event message. It has the following elements: <b>pid (long)</b> Process identifier <b>msg (unsigned long)</b> Event message identifier <b>mp1 (unsigned long)</b> Message parameter 1 <b>mp2 (unsigned long)</b> Message parameter 2

---

### Macros

The following is a complete list of the macros that are described in the header `pos.h` file. Additional macros for parsing messages are in the `helper.h` file.

Macro	Meaning
<b>PosGetValues</b>	This macro calls the <i>PosIOctl()</i> subroutine with the <code>POS_SYS_GET_VALUES</code> request. The complete definition of this macro is: <pre>#define PosGetValues(device, args, nargs) \ PosIOctl(device, POS_SYS_GET_VALUES, args, nargs)</pre>
<b>PosNumber</b>	This macro determines the size of a fixed-size array and returns the size. The complete definition of this macro is: <pre>#define PosNumber(arr) \ ((unsigned int) (sizeof(arr) / sizeof(arr[0])))</pre>
<b>PosSetArg</b>	This macro assigns a value to a resource in a fixed-size argument list array. The complete definition of this macro is: <pre>#define PosSetArg(arg, n, d) \ ((arg).name = (n), (arg).value = (long)(d))</pre>

**PosSetValues** This macro calls the *PosIOctl()* subroutine with the POS\_SYS\_SET\_VALUES request. The complete definition of this macro is:

```
#define PosSetValues(device, args, nargs) \  
PosIOctl(device, POS_SYS_SET_VALUES, args, nargs)
```

---

## Appendix B. Trace Codes

For tracing with the OS/2 system trace facility, or with the built-in trace facility in the IBM Point of Sale Subsystem for Windows and the IBM Point of Sale Subsystem for Linux, a major trace code of 93 (0x5D) is used for all trace events. Within this major trace code, the IBM Point of Sale Subsystem uses minor trace codes in the following ranges:

Range	Meaning
<b>1 (0x01) through 31 (0x1F)</b>	IBM Point of Sale Subsystem API subroutines
<b>32 (0x20) through 63 (0x3F)</b>	SIO (Serial Input/Output) messages
<b>64 (0x40) through 71 (0x47)</b>	Error trace events
<b>72 (0x48) through 96 (0x60)</b>	Other trace events

The following are the minor trace codes used by the IBM Point of Sale Subsystem:

Minor Code	Meaning
<b>1 (0x01)</b>	PosInitialize() subroutine entry
<b>2 (0x02)</b>	PosOpen() subroutine entry
<b>3 (0x03)</b>	PosClose() subroutine entry
<b>4 (0x04)</b>	PosRead() subroutine entry
<b>5 (0x05)</b>	PosWrite() subroutine entry
<b>6 (0x06)</b>	PosIOCtl() subroutine entry
<b>7 (0x07)</b>	Exit list processing entry point
<b>17 (0x11)</b>	PosInitialize() subroutine exit results
<b>18 (0x12)</b>	PosOpen() subroutine exit results
<b>19 (0x13)</b>	PosClose() subroutine exit results
<b>20 (0x14)</b>	PosRead() subroutine exit results
<b>21 (0x15)</b>	PosWrite() subroutine exit results
<b>22 (0x16)</b>	PosIOCtl() subroutine exit results
<b>23 (0x17)</b>	Exit list processing exit results
<b>32 (0x20)</b>	SIO EC level
<b>34 (0x22)</b>	SIO error message
<b>35 (0x23)</b>	Incoming SIO message
<b>36 (0x24)</b>	Outgoing SIO message
<b>43 (0x2B)</b>	Incoming SIO I-Frame message
<b>44 (0x2C)</b>	Outgoing SIO I-Frame message
<b>64 (0x40)</b>	Error Event
<b>72 (0x48)</b>	IBM Point of Sale Subsystem DLL initialization/exit (OS/2 only)
<b>73 (0x49)</b>	IBM Point of Sale Subsystem C run-time initialization/exit (OS/2 only)
<b>74 (0x4A)</b>	IBM Point of Sale Subsystem FFST/2 initialization/exit (OS/2 only)
<b>75 (0x4B)</b>	IBM Point of Sale Subsystem heap initialization/exit (OS/2 only)
<b>76 (0x4C)</b>	IBM Point of Sale Subsystem semaphore initialization/exit (OS/2 only)



---

## Appendix C. Error Messages

This appendix contains IBM Point of Sale Subsystem error messages.

**Note:** The error messages listed in this appendix are available only on OS/2.

---

**AIP0001**      **Error %d occurred at location %d. See error log for details.**

**Explanation:** An unexpected error occurred.

**User Action:** Submit the error log details to IBM.

---

**AIP0030**      **ANPOS Utility: Incorrect numpadZero value %ld.**

**Explanation:** An incorrect numeric keypad zero value was specified in the resource file.

**User Action:** Make sure a correct value is specified for the **numpadZero** resource. The correct values are **DOUBLE\_KEY** or **SINGLE\_KEY**.

---

**AIP0032**      **ANPOS Utility: Invalid numpadStyle value %d.**

**Explanation:** An invalid numeric keypad style value was specified in the resource file.

**User Action:** Make sure a correct value is specified for the **numpadStyle** resource. The correct values are: **TOUCHTONE\_POS**, **CALCULATOR\_POS**, **TOUCHTONE\_STANDARD** or **CALCULATOR\_STANDARD**.

---

**AIP0033**      **ANPOS Utility: Invalid keyboard click value %d.**

**Explanation:** An invalid keyboard key click value was specified in the resource file.

**User Action:** Make sure a correct value is specified for the **keyboardClick** resource. The correct values are: **OFF**, **SOFT**, or **LOUD**.

---

**AIP0034**      **ANPOS Utility: Invalid double key: %s.**

**Explanation:** An invalid double key value was specified in the resource file.

**User Action:** Make sure a correct value is specified for each **doubleKeyXX** resource.

---

**AIP0035**      **ANPOS Utility: Error opening KBD\$: %d.**

**Explanation:** The DosOpen call for the KBD\$ device driver failed for an unexpected reason.

**User Action:** Contact your IBM service representative.

---

---

**AIP0036**      **ANPOS Utility: Error closing KBD\$: %d.**

**Explanation:** The DosClose call for the KBD\$ device driver failed for an unexpected reason.

**User Action:** Contact your IBM service representative.

---

**AIP0037**      **ANPOS Utility: Error setting double keys: %d.**

**Explanation:** The command to the ANPOS keyboard failed. The return code indicates the operating system error.

**User Action:** Contact your IBM service representative.

---

**AIP0039**      **ANPOS Utility: No %s code update file.**

**Explanation:** No code update file was found. This is not an error condition unless there is a necessary code update file for the microcode level of the keyboard.

**User Action:** Contact your IBM service representative.

---

**AIP0040**      **ANPOS Utility: Error opening code update file: %s.**

**Explanation:** An error occurred reading the code update file.

**User Action:** Make sure that the file is not corrupted. If the problem persists, contact your IBM service representative.

---

**AIP0041**      **ANPOS Utility: Not a point-of-sale keyboard.**

**Explanation:** The keyboard is not responding to point-of-sale keyboard specific commands.

**User Action:** If it is a point-of-sale keyboard, unplug it and plug it in again to reset. If the problem persists, contact your IBM service representative.

---

**AIP0042**      **ANPOS Utility: Point of Sale KBD\$ device driver not found.**

**Explanation:** The file KBD01.SYS, or KBDO2.SYS is not the IBM Point of Sale Subsystem version.

**User Action:** Replace the file with the one provided with the IBM Point of Sale Subsystem.

---

## Error Messages

created on October 2, 2001

---

**AIP0050      Translate Table Utility: Error opening  
KEYBOARD.DCP: %d.**

**Explanation:** The DosOpen call for the  
KEYBOARD.DCP file failed for an unexpected reason.

**User Action:** Contact your IBM service representative.

---

**AIP0051      Translate Table Utility: Error opening  
KEYBOARD.DCP: %d.**

**Explanation:** The DosOpen call for the  
KEYBOARD.DCP file failed for an unexpected reason.

**User Action:** Contact your IBM service representative.

---

**AIP0052      Translate Table Utility: Error opening  
KEYBOARD.DCP: %d.**

**Explanation:** The DosOpen call for the  
KEYBOARD.DCP file failed for an unexpected reason.

**User Action:** Contact your IBM service representative.

---

## Appendix D. Error Codes

Error Messages, Numeric Order . . . . .	D-4
301 POSERR_SYS_OS_ERROR. . . . .	D-4
302 POSERR_SYS_NOT_INITIALIZED . . . . .	D-4
303 POSERR_SYS_INVALID_DESCRIPTOR . . . . .	D-4
304 POSERR_SYS_ALREADY_INITIALIZED . . . . .	D-4
305 POSERR_SYS_MEMORY_ALLOCATION . . . . .	D-4
306 POSERR_SYS_HW_ERROR . . . . .	D-5
307 POSERR_SYS_INVALID_DEVICE . . . . .	D-5
308 POSERR_SYS_INVALID_QUEUE. . . . .	D-5
309 POSERR_SYS_TOO_MANY_DEVICES . . . . .	D-5
311 POSERR_SYS_FUNCTION_NOT_SUPPORTED . . . . .	D-5
312 POSERR_SYS_BUFFER_TOO_SMALL . . . . .	D-5
313 POSERR_SYS_ACQUIRED_BY_OTHER . . . . .	D-5
314 POSERR_SYS_ALREADY_ACQUIRED . . . . .	D-5
315 POSERR_SYS_NOT_ACQUIRED. . . . .	D-6
316 POSERR_SYS_INVALID_REQUEST. . . . .	D-6
317 POSERR_SYS_DEVICE_OFFLINE . . . . .	D-6
318 POSERR_SYS_INVALID_LENGTH . . . . .	D-6
319 POSERR_SYS_INVALID_CLASS_DEVICE_COMBO . . . . .	D-6
320 POSERR_SYS_DATA_DISCARDED . . . . .	D-6
321 POSERR_SYS_INTERNAL_ERROR. . . . .	D-7
325 POSERR_SYS_INVALID_NARGS. . . . .	D-7
326 POSERR_SYS_INVALID_SLOT . . . . .	D-7
327 POSERR_SYS_UNSUPPORTED_ADAPTER . . . . .	D-7
328 POSERR_SYS_INVALID_PORT . . . . .	D-7
329 POSERR_SYS_TIMEOUT . . . . .	D-7
330 POSERR_SYS_INVALID_NAME . . . . .	D-7
331 POSERR_SYS_INVALID_CLASS . . . . .	D-7
332 POSERR_SYS_INTERRUPTED . . . . .	D-7
334 POSERR_SYS_INVALID_ADDRESS. . . . .	D-8
335 POSERR_SYS_LOCKED_NO_DATA_READ . . . . .	D-8
336 POSERR_SYS_INVALID_FILE . . . . .	D-8
337 POSERR_SYS_SERVICE_NOT_AVAILABLE . . . . .	D-8
4101 POSERR_NVRAM_NOT_ENOUGH_ROOM . . . . .	D-8
4102 POSERR_NVRAM_INVALID_CURSOR . . . . .	D-8
4103 POSERR_NVRAM_EOF . . . . .	D-8
4104 POSERR_NVRAM_INVALID_MODE . . . . .	D-9
4105 POSERR_NVRAM_INVALID_LENGTH_RECORD . . . . .	D-9
4106 POSERR_NVRAM_INVALID_DATA_CRC . . . . .	D-9
4401 POSERR_DSP_INVALID_CURSOR . . . . .	D-9
4402 POSERR_DSP_INVALID_MODE. . . . .	D-9
4403 POSERR_DSP_INVALID_SIZE . . . . .	D-9
4404 POSERR_DSP_UNSUPPORTED_BITMAP . . . . .	D-9
4405 POSERR_DSP_BAD_BITMAP . . . . .	D-9
4406 POSERR_DSP_INVALID_CODE_PAGE . . . . .	D-10
4701 POSERR_KBD_INVALID_FREQUENCY . . . . .	D-10
4702 POSERR_KBD_INVALID_DURATION . . . . .	D-10
4703 POSERR_KBD_INVALID_VOLUME . . . . .	D-10
4705 POSERR_KBD_INVALID_DOUBLE_KEY . . . . .	D-10
4706 POSERR_KBD_INVALID_FAT_FINGER_TIMEOUT . . . . .	D-10
4708 POSERR_KBD_INVALID_KEYBOARD_CLICK . . . . .	D-10
4709 POSERR_KBD_INVALID_NUMPAD_STYLE . . . . .	D-10
4710 POSERR_KBD_INVALID_NUMPAD_ZERO . . . . .	D-10

4711	POSERR_KBD_INVALID_TYEMATIC_DELAY . . . . .	D-11
4712	POSERR_KBD_INVALID_TYEMATIC_FREQ . . . . .	D-11
4713	POSERR_KBD_INVALID_NUMPAD_LOCATION . . . . .	D-11
4901	POSERR_PRN_INVALID_DI_WIDTH. . . . .	D-11
4902	POSERR_PRN_INVALID_INTERLEAVED_VALUE . . . . .	D-11
4903	POSERR_PRN_INVALID_HEAD_PARKED_POSITION . . . . .	D-11
4904	POSERR_PRN_INVALID_STATION . . . . .	D-11
4905	POSERR_PRN_INVALID_MODE . . . . .	D-12
4906	POSERR_PRN_INVALID_CR_LF_DISTANCE . . . . .	D-12
4907	POSERR_PRN_INVALID_SJ_LF_DISTANCE. . . . .	D-12
4908	POSERR_PRN_INVALID_DI_LF_DISTANCE. . . . .	D-12
4909	POSERR_PRN_INVALID_FEED_DIRECTION . . . . .	D-12
4910	POSERR_PRN_INVALID_FISCAL_NOTIFY . . . . .	D-12
4911	POSERR_PRN_INVALID_DI_ORIENTATION . . . . .	D-12
4912	POSERR_PRN_INVALID_LEFT_MARGIN_CR . . . . .	D-12
4913	POSERR_PRN_INVALID_PRINT_ALIGNMENT . . . . .	D-12
4914	POSERR_PRN_INCORRECT_DATA_FORMAT . . . . .	D-12
4915	POSERR_PRN_LOGO_EXISTS . . . . .	D-13
4916	POSERR_PRN_UDC_CHARACTER_EXISTS . . . . .	D-13
4918	POSERR_PRN_INVALID_QUALITY_MODE . . . . .	D-13
4919	POSERR_PRN_INVALID_UPSIDE_DOWN_MODE . . . . .	D-13
4920	POSERR_PRN_INVALID_TABSTOPS_FORMAT . . . . .	D-13
4921	POSERR_PRN_INVALID_COLOR_MODE. . . . .	D-13
4922	POSERR_PRN_INVALID_TONE_VOLUME . . . . .	D-13
4923	POSERR_PRN_INVALID_TONE_DURATION . . . . .	D-14
4924	POSERR_PRN_INVALID_TONE_NOTE . . . . .	D-14
4925	POSERR_PRN_INVALID_TONE_OCTAVE . . . . .	D-14
4926	POSERR_PRN_INVALID_TONE_FREQUENCY. . . . .	D-14
4927	POSERR_PRN_INVALID_CODE_PAGE . . . . .	D-14
5401	POSERR_CDR_INVALID_PULSE_WIDTH. . . . .	D-14
5702	POSERR_SCAN_INVALID_BAR_CODES_1 . . . . .	D-14
5703	POSERR_SCAN_INVALID_BAR_CODES_2 . . . . .	D-14
5704	POSERR_SCAN_INVALID_BEEP_FREQ . . . . .	D-14
5705	POSERR_SCAN_INVALID_BEEP_LENGTH . . . . .	D-15
5706	POSERR_SCAN_INVALID_BEEP_STATE . . . . .	D-15
5707	POSERR_SCAN_INVALID_BEEP_VOLUME . . . . .	D-15
5708	POSERR_SCAN_INVALID_BLINK_LENGTH. . . . .	D-15
5709	POSERR_SCAN_INVALID_BLOCK_READ_MODE . . . . .	D-15
5710	POSERR_SCAN_INVALID_BLOCK_1_TYPE. . . . .	D-15
5711	POSERR_SCAN_INVALID_BLOCK_2_TYPE. . . . .	D-15
5712	POSERR_SCAN_INVALID_BLOCK_3_TYPE. . . . .	D-15
5713	POSERR_SCAN_INVALID_CHECK_MODULO . . . . .	D-15
5714	POSERR_SCAN_INVALID_D_READ_TIMEOUT . . . . .	D-15
5715	POSERR_SCAN_INVALID_D_TOUCH_MODE . . . . .	D-16
5716	POSERR_SCAN_INVALID_ITF_LENGTH_1 . . . . .	D-16
5717	POSERR_SCAN_INVALID_ITF_LENGTH_2 . . . . .	D-16
5718	POSERR_SCAN_INVALID_LASER_TIMEOUT . . . . .	D-16
5719	POSERR_SCAN_INVALID_MOTOR_TIMEOUT. . . . .	D-16
5720	POSERR_SCAN_INVALID_LASER_SWITCH_STATE . . . . .	D-16
5721	POSERR_SCAN_INVALID_SCANS_PER_READ . . . . .	D-16
5722	POSERR_SCAN_LABEL_TOO_SHORT . . . . .	D-16
5725	POSERR_SCAN_INVALID_BVOL_SWITCH_STATE . . . . .	D-16
5726	POSERR_SCAN_INVALID_DECODE_ALGORITHM . . . . .	D-17
5727	POSERR_SCAN_INVALID_EAN13_SCANS_PER_READ . . . . .	D-17
5728	POSERR_SCAN_INVALID_EAN8_SCANS_PER_READ. . . . .	D-17
5729	POSERR_SCAN_INVALID_STORE_SCANS_PER_READ . . . . .	D-17

5730	POSERR_SCAN_INVALID_UPCA_SCANS_PER_READ	D-17
5731	POSERR_SCAN_INVALID_UPCD_SCANS_PER_READ	D-17
5732	POSERR_SCAN_INVALID_UPCE_SCANS_PER_READ	D-17
5733	POSERR_SCAN_INVALID_UPC_EXPANSION	D-17
5734	POSERR_SCAN_INVALID_VERIFY_PRICE_CHK	D-17
5735	POSERR_SCAN_INVALID_QUEUE_ALL_INDICATOR	D-17
5736	POSERR_SCAN_CONFIGURATION_ERROR	D-18
5737	POSERR_SCAN_2_LABEL_FLAG_CONFIG_ERROR	D-18
5738	POSERR_SCAN_INVALID_2_LABEL_DECODE_STATE	D-18
5739	POSERR_SCAN_INVALID_FLAG_PAIR_COMBINATION	D-18
5740	POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_1	D-18
5741	POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_2	D-18
5742	POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_3	D-18
5743	POSERR_SCAN_INVALID_2_LABEL_FLAG_PAIR_4	D-18
5744	POSERR_SCAN_INVALID_CODE39_SCANS_PER_READ	D-19
5745	POSERR_SCAN_INVALID_INT2OF5_SCANS_PER_READ	D-19
5746	POSERR_SCAN_INVALID_CODE128_SCANS_PER_READ	D-19
5747	POSERR_SCAN_INVALID_BAR_CODES_3	D-19
5748	POSERR_SCAN_INVALID_BAR_CODES_4	D-19
5750	POSERR_SCAN_INVALID_ITF_LENGTH_TYPE	D-19
5751	POSERR_SCAN_INVALID_SUPPLEMENTALS	D-19
5752	POSERR_SCAN_INVALID_BARCODE_PROG_STATE	D-19
5753	POSERR_SCAN_INVALID_XMIT_CHECK_DIGIT	D-19
5901	POSERR_RS232_INVALID_BAUD_RATE	D-19
5902	POSERR_RS232_INVALID_STOP_BITS	D-20
5903	POSERR_RS232_INVALID_PARITY	D-20
5904	POSERR_RS232_INVALID_DATA_BITS	D-20
5905	POSERR_RS232_INVALID_TIMEOUT_CHAR	D-20
5907	POSERR_RS232_PREV_NOT_COMPLETE	D-20
6201	POSERR_SCALE_INVALID_OPERATIONS_MODE	D-20
6202	POSERR_SCALE_INVALID_REMOTE_DISPLAY_STATE	D-20
6203	POSERR_SCALE_INVALID_VIBRATION_FILTER	D-20
6204	POSERR_SCALE_INVALID_WEIGHT_MODE	D-20
6205	POSERR_SCALE_INVALID_ZERO_INDICATOR_STATE	D-20
6206	POSERR_SCALE_INVALID_ZERO_RETURN_STATE	D-21
6207	POSERR_SCALE_ZERO_SCALE_FAILED	D-21
6208	POSERR_SCALE_INVALID_CLEAR_SCREEN_REQUEST	D-21
6209	POSERR_SCALE_CONFIGURATION_ERROR	D-21
6211	POSERR_SCALE_INVALID_NUM_WEIGHT_DIGITS	D-21
6401	POSERR_POWER_INVALID_DAY	D-21
6402	POSERR_POWER_INVALID_HOUR	D-21
6403	POSERR_POWER_INVALID_MINUTES	D-21
6701	POSERR_TOUCH_INVALID_BACKLIGHT_ON	D-21
6702	POSERR_TOUCH_INVALID_CLICK_VOLUME	D-22
6703	POSERR_TOUCH_INVALID_CONTRAST	D-22
6704	POSERR_TOUCH_INVALID_ENTRY_CLICK	D-22
6705	POSERR_TOUCH_INVALID_EXIT_CLICK	D-22
6706	POSERR_TOUCH_INVALID_MODE	D-22
6707	POSERR_TOUCH_INVALID_SCREEN_SAVER_TIME	D-22
6708	POSERR_TOUCH_INVALID_TONE_DURATION	D-22
6709	POSERR_TOUCH_INVALID_TONE_FREQUENCY	D-22
6710	POSERR_TOUCH_INVALID_TONE_VOLUME	D-22
6711	POSERR_TOUCH_INVALID_BRIGHTNESS	D-23

Table D-1 on page D-4 lists the general components of the system and the assigned number range for errors generated by, or pertaining to that component area. This

table can be used as a quick reference when you are presented with an error code in either decimal, or hexadecimal notation.

Table D-1. Error Code Reference

Component	Hexadecimal Range	Decimal Range
System	01xx	03xx
NVRAM	10xx	41xx
Display	11xx	44xx
Keyboard	12xx	47xx
Programmable power	19xx	64xx
Printer	13xx	49xx
MSR	14xx	52xx
Cash Drawer/Alarm	15xx	54xx
Scanner	16xx	57xx
RS-232C	17xx	59xx
Scale	18xx	62xx
Touch	1Axx	67xx

---

## Error Messages, Numeric Order

This section contains a list of error messages in numeric order.

### 301 POSERR\_SYS\_OS\_ERROR

**Explanation:** The IBM Point of Sale Subsystem received an operating system error.

### 302 POSERR\_SYS\_NOT\_INITIALIZED

**Explanation:** The application has not performed a successful *PosInitialize()* subroutine call. An application must successfully issue the *PosInitialize()* subroutine call before issuing any other IBM Point of Sale Subsystem subroutine calls. Check your application program logic.

### 303 POSERR\_SYS\_INVALID\_DESCRIPTOR

**Explanation:** The *devdes* parameter is not a valid device descriptor. Only device descriptors returned by *PosOpen()* can be used. Check your application program logic.

### 304 POSERR\_SYS\_ALREADY\_INITIALIZED

**Explanation:** The application has already successfully processed the *PosInitialize()* subroutine. Check your application program logic.

### 305 POSERR\_SYS\_MEMORY\_ALLOCATION

**Explanation:** The IBM Point of Sale Subsystem has not been able to acquire the memory it requires to operate.

### 306 POSERR\_SYS\_HW\_ERROR

**Explanation:** The IBM Point of Sale Subsystem has detected an error with one of the devices, or other hardware attached to the system. Test the hardware device.

### 307 POSERR\_SYS\_INVALID\_DEVICE

**Explanation:** A specific device could not be determined from the information given to the *PosOpen()* subroutine. Possible problems are:

1. The value of the **PosNdeviceNumber** resource was not recognized as a valid device number for the IBM Point of Sale Subsystem.
2. The **PosNdeviceNumber** resource was missing, misspelled, or not valid.

### 308 POSERR\_SYS\_INVALID\_QUEUE

**Explanation:** The value for the **PosNqueueHandle** resource is not valid. See “PosNqueueHandle” on page 21-14 for the allowed values.

### 309 POSERR\_SYS\_TOO\_MANY\_DEVICES

**Explanation:** The IBM Point of Sale Subsystem limit for device descriptors for each process has already been reached. The IBM Point of Sale Subsystem cannot start any more devices for this process until one or more devices are closed.

### 311 POSERR\_SYS\_FUNCTION\_NOT\_SUPPORTED

**Explanation:** The application has requested a function that is not supported for the device descriptor. See the Related Information section of the individual device chapters for the valid subroutine calls for the device.

### 312 POSERR\_SYS\_BUFFER\_TOO\_SMALL

**Explanation:** This error can occur when issuing a *PosRead()* subroutine call. The application specified a value for *nbyte* that was too small for the data being read. When this error occurs, data is not put into the application’s buffer. You should obtain the required size buffer and retry the read operation. The required size can be found in the event message that indicated that data is available.

### 313 POSERR\_SYS\_ACQUIRED\_BY\_OTHER

**Explanation:** The application issued a `POS_SYS_ACQUIRE_DEVICE` *PosIOctl()* request in order to acquire device connection, but another device descriptor has already acquired the device connection. The acquired device connection must be released before this `POS_SYS_ACQUIRE_DEVICE` *PosIOctl()* request can be successful.

### 314 POSERR\_SYS\_ALREADY\_ACQUIRED

**Explanation:** The application is trying to acquire a device connection that it has already acquired. Check your application program logic.

### 315 POSERR\_SYS\_NOT\_ACQUIRED

**Explanation:** The application issued a subroutine call that requires an acquired device connection. The application must first successfully issue the `POS_SYS_ACQUIRE_DEVICE PosIOctl()` request. Check your application program logic.

### 316 POSERR\_SYS\_INVALID\_REQUEST

**Explanation:** The *request* parameter of the `PosIOctl()` subroutine is not valid for the device associated with the device descriptor. See the related information section of the individual device chapters for the valid requests for the device.

### 317 POSERR\_SYS\_DEVICE\_OFFLINE

**Explanation:** The device is not currently connected to the system unit. Your application might have tried to open the device prior to receiving the `POSM_SYS_DEVICE_ONLINE` event message.

### 318 POSERR\_SYS\_INVALID\_LENGTH

**Explanation:** The application specified an incorrect value for *nbyte* when it issued a `PosWrite()` subroutine call.

For the `PosWrite()` subroutine call to an NVRAM device, *nbyte* contains a value that is not valid. The maximum length of the data that can be written to the NVRAM device at a single time varies based on the type of workstation and the value of the **PosNnvramMode** resource when you opened the device.

For the `PosWrite()` subroutine call to an RS-232C connection, the length of the data must be less than 248 bytes.

For the `PosWrite()` subroutine call to a Model 2, Model 3, or Model 4 printer, the length of the data must be less than 1025 bytes.

For the `PosWrite()` subroutine call to a IBM 4689-301 printer, the length of the data must be less than 8,193 bytes.

For the `PosWrite()` subroutine call to a SureMark 4610 printer, the length of the data must be less than 16,385 bytes.

### 319 POSERR\_SYS\_INVALID\_CLASS\_DEVICE\_COMBO

**Explanation:** The values of the *class* parameter and the **PosNdeviceNumber** resource given to the `PosOpen()` subroutine were both valid individually, but not valid in combination with each other. For example, a *class* parameter of "PosScanner" and a **PosNdeviceNumber** resource value of `PosDEVICE_OPERATOR_DISPLAY_A` will cause this error to be returned (assuming there are no other errors). Check your application program logic.

### 320 POSERR\_SYS\_DATA\_DISCARDED

**Explanation:** A device was closed when data was available. The data is discarded. The data is not logged.

### 321 POSERR\_SYS\_INTERNAL\_ERROR

**Explanation:** The IBM Point of Sale Subsystem has discovered an internal logic error. Please contact your IBM service representative.

### 325 POSERR\_SYS\_INVALID\_NARGS

**Explanation:** The *nargs* parameter is a negative number. It must be 0 (zero), or a positive number. Check your application program logic.

### 326 POSERR\_SYS\_INVALID\_SLOT

**Explanation:** An incorrect value was specified for the **PosNslotNumber** resource. Make sure that you typed the value correctly. See “PosNslotNumber” on page 21-15 for the allowed values.

### 327 POSERR\_SYS\_UNSUPPORTED\_ADAPTER

**Explanation:** The **PosNslotNumber** resource specified a slot containing an unsupported adapter. See “PosNslotNumber” on page 21-15 for the allowed values.

### 328 POSERR\_SYS\_INVALID\_PORT

**Explanation:** An incorrect value was specified for the **PosNportNumber** resource. Make sure that you typed the value correctly. See “PosNportNumber” on page 21-14 for the allowed values.

### 329 POSERR\_SYS\_TIMEOUT

**Explanation:** The timeout specified by the **PosNreadTimeout** resource has expired while waiting for input from the IBM Point of Sale Subsystem input queue. Data did not become available for reading from the input queue before the timeout expired.

### 330 POSERR\_SYS\_INVALID\_NAME

**Explanation:** The length of the *name* parameter was 0 (zero) or was too long. Use a *name* parameter value that has a valid length. See “PosOpen()” on page 18-10, or “PosInitialize()” on page 18-5 for valid lengths.

### 331 POSERR\_SYS\_INVALID\_CLASS

**Explanation:** The *class* parameter given to the *PosOpen()* subroutine is not valid. Possible problems are:

1. The length of the *class* parameter is 0 (zero), or is too long.
2. The length of the *class* parameter is valid, but the class is not recognized by the IBM Point of Sale Subsystem.

### 332 POSERR\_SYS\_INTERRUPTED

**Explanation:** A system call was interrupted, or could not be completed for one of the following reasons:

1. The application called the *PosRead()* subroutine for the IBM Point of Sale Subsystem input queue (device descriptor zero) and was waiting for input to become available (the value of the **PosNreadTimeout**

resource was non-zero). While the thread was waiting, an exception occurred, such as the user pressing Ctrl-Break.

2. The application called the *PosRead()* subroutine for the IBM Point of Sale Subsystem input queue (device descriptor zero) and was waiting for input to become available (the value of the **PosNreadTimeout** resource was non-zero). While the thread was waiting, another thread in the same process called the *PosRead()*, or *PosIOCtl()* subroutines for the input queue. In this case, the input queue was already busy and could not process the request.

### 334 POSERR\_SYS\_INVALID\_ADDRESS

**Explanation:** The IBM Point of Sale Subsystem detected an address that is not valid. This error is set if the buffer, the control block, or the string pointed to is not valid, or if the process does not have access to the entire length of the control block, or buffer.

The entire length of structures of known sizes are validated before being used. If the structure is an output parameter, then the structure is verified to be writeable. If the structure is of an unknown size, only the first byte of the structure is validated.

### 335 POSERR\_SYS\_LOCKED\_NO\_DATA\_READ

**Explanation:** The application issued a *PosRead()* subroutine call when the target device is in a locked state and there is no data available to be returned.

### 336 POSERR\_SYS\_INVALID\_FILE

**Explanation:** The *file* parameter given to the *PosInitialize()* subroutine is not valid. The origin of the problem might be that the length of the *file* parameter is 0 (zero), or is too long.

### 337 POSERR\_SYS\_SERVICE\_NOT\_AVAILABLE

**Explanation:** The IBM Point of Sale Subsystem for Windows is not running. This error code is available only on the Microsoft Windows operating system.

### 4101 POSERR\_NVRAM\_NOT\_ENOUGH\_ROOM

**Explanation:** This error can occur when issuing a *PosWrite()* subroutine call to an NVRAM device. It indicates that there is not enough room in the NVRAM device to write all of the data that was requested to be written.

### 4102 POSERR\_NVRAM\_INVALID\_CURSOR

**Explanation:** An incorrect value was specified for the **PosNvramCursor** resource. Valid values for this resource are from 0 (zero) through the limit defined in the "list of NVRAM application address space" on page 11-2.

### 4103 POSERR\_NVRAM\_EOF

**Explanation:** The application attempted to read past the end-of-file marker while accessing the NVRAM device in sequential mode.

## 4104 POSERR\_NVRAM\_INVALID\_MODE

**Explanation:** An incorrect value was specified for the **PosNnvramMode** resource. Make sure that you typed the value correctly. See “PosNnvramMode” on page 21-31 for the allowed values.

## 4105 POSERR\_NVRAM\_INVALID\_LENGTH\_RECORD

**Explanation:** This error can occur when issuing a *PosRead()* subroutine call to an NVRAM device. It indicates that the record’s length, as indicated by the first 2 bytes of the NVRAM record, is not valid. This situation can occur if the cursor is positioned to a location that is not the beginning of a record, or if the length bytes have been corrupted and thus contain data that is not valid.

## 4106 POSERR\_NVRAM\_INVALID\_DATA\_CRC

**Explanation:** This error can occur when issuing a *PosRead()* subroutine call to an NVRAM device. This error indicates that the record’s data failed a CRC check. The result of this means that you are sure that the data currently present in this record is not exactly the same as that from which it was written. You know how long the record is, but you do not know what part of the data is wrong. The record will be copied to the application’s buffer, but you should be cautious about using the data.

## 4401 POSERR\_DSP\_INVALID\_CURSOR

**Explanation:** An incorrect value was specified for the **PosNdisplayCursor** resource. Make sure that you typed the value correctly. See “PosNdisplayCursor” on page 21-19 for the allowed values.

## 4402 POSERR\_DSP\_INVALID\_MODE

**Explanation:** An incorrect value was specified for the **PosNdisplayMode** resource. Make sure that you typed the value correctly. See “PosNdisplayMode” on page 21-19 for the allowed values.

## 4403 POSERR\_DSP\_INVALID\_SIZE

**Explanation:** An incorrect value was specified for the **PosNcharSize** resource. Make sure that you typed the value correctly. See “PosNcharSize” on page 21-17 for the allowed values.

## 4404 POSERR\_DSP\_UNSUPPORTED\_BITMAP

**Explanation:** The type field of the bitmap data (first 4 bytes) does not contain a valid value. See “Writing Bitmaps to the Display” on page 8-5 for more information.

## 4405 POSERR\_DSP\_BAD\_BITMAP

**Explanation:** See “Writing Bitmaps to the Display” on page 8-5 for more information.

**4406 POSERR\_DSP\_INVALID\_CODE\_PAGE**

**Explanation:** An incorrect value was specified for the **PosNdisplayCodePage** resource. See “PosNdisplayCodePage” on page 21-17 for the allowed values.

**4701 POSERR\_KBD\_INVALID\_FREQUENCY**

**Explanation:** An incorrect value was specified for the **PosNtoneFreq** resource. Make sure that you typed the value correctly. See “PosNtoneFreq” on page 21-28 for the allowed values.

**4702 POSERR\_KBD\_INVALID\_DURATION**

**Explanation:** An incorrect value was specified for the **PosNtoneDuration** resource. Make sure that you typed the value correctly. See “PosNtoneDuration” on page 21-28 for the allowed values.

**4703 POSERR\_KBD\_INVALID\_VOLUME**

**Explanation:** An incorrect value was specified for the **PosNtoneVolume** resource. Make sure that you typed the value correctly. See “PosNtoneVolume” on page 21-28 for the allowed values.

**4705 POSERR\_KBD\_INVALID\_DOUBLE\_KEY**

**Explanation:** An incorrect value was specified for the **PosNdoubleKey01 - PosNdoubleKey60** resource. Make sure that you typed the value correctly. See “PosNdoubleKey01 - PosNdoubleKey60” on page 21-23 for the allowed values.

**4706 POSERR\_KBD\_INVALID\_FAT\_FINGER\_TIMEOUT**

**Explanation:** An incorrect value was specified for the **PosNfatFingerTimeOut** resource. Make sure that you typed the value correctly. See “PosNfatFingerTimeOut” on page 21-24 for the allowed values.

**4708 POSERR\_KBD\_INVALID\_KEYBOARD\_CLICK**

**Explanation:** An incorrect value was specified for the **PosNkeyboardClick** resource. Make sure that you typed the value correctly. See “PosNkeyboardClick” on page 21-25 for the allowed values.

**4709 POSERR\_KBD\_INVALID\_NUMPAD\_STYLE**

**Explanation:** An incorrect value was specified for the **PosNnumpadStyle** resource. Make sure that you typed the value correctly. See “PosNnumpadStyle” on page 21-27 for the allowed values.

**4710 POSERR\_KBD\_INVALID\_NUMPAD\_ZERO**

**Explanation:** An incorrect value was specified for the **PosNnumpadZero** resource. Make sure that you typed the value correctly. See “PosNnumpadZero” on page 21-27 for the allowed values.

## 4711 POSERR\_KBD\_INVALID\_TYPEMATIC\_DELAY

**Explanation:** An incorrect value was specified for the **PosNtypematicDelay** resource. Make sure that you typed the value correctly. See “PosNtypematicDelay” on page 21-29 for the allowed values.

## 4712 POSERR\_KBD\_INVALID\_TYPEMATIC\_FREQ

**Explanation:** An incorrect value was specified for the **PosNtypematicFreq** resource. Make sure that you typed the value correctly. See “PosNtypematicFreq” on page 21-29 for the allowed values.

## 4713 POSERR\_KBD\_INVALID\_NUMPAD\_LOCATION

**Explanation:** An incorrect value was specified for the **PosNnumpadLocation** resource. Make sure that you typed the value correctly. See “PosNnumpadLocation” on page 21-26 for the allowed values.

## 4901 POSERR\_PRN\_INVALID\_DI\_WIDTH

**Explanation:** An incorrect value was specified for the **PosNDIWidth** resource. Make sure that you typed the value correctly. See “PosNDIWidth” on page 21-35 for the allowed values.

## 4902 POSERR\_PRN\_INVALID\_INTERLEAVED\_VALUE

**Explanation:** An incorrect value was specified for the **PosNinterleaved** resource. Make sure that you typed the value correctly. See “PosNinterleaved” on page 21-39 for the allowed values.

## 4903 POSERR\_PRN\_INVALID\_HEAD\_PARKED\_POSITION

**Explanation:** An incorrect value was specified for the **PosNheadParkedPosition** resource. Make sure that you typed the value correctly. See “PosNheadParkedPosition” on page 21-38 for the allowed values.

## 4904 POSERR\_PRN\_INVALID\_STATION

**Explanation:** The possible reasons for this error are:

1. An incorrect value was specified for the **PosNprintStation** resource. Make sure that you typed the value correctly. See “PosNprintStation” on page 21-45 for the allowed values.
2. The application has issued a *PosWrite()* subroutine call with the **PosNprintStation** resource equal to PosPRINT\_STATION\_SJ and the **PosNprintMode** resource equal to PosMODE\_LOGO. See “PosNprintMode” on page 21-44 and “PosNprintStation” on page 21-45 for the allowed values.
3. The application has issued a *PosWrite()* subroutine call with the **PosNprintStation** resource equal to PosPRINT\_STATION\_DI, the **PosNdiOrientation** resource equal to PosPRINT\_LANDSCAPE\_ORIENTATION, and the **PosNprintMode** resource equal to PosMODE\_LOGO. See “PosNprintMode” on page 21-44, “PosNprintStation” on page 21-45, and “PosNdiOrientation” on page 21-35 for the allowed values.

**4905 POSERR\_PRN\_INVALID\_MODE**

**Explanation:** An incorrect value was specified for the **PosNprintMode** resource. Make sure that you typed the value correctly. See “PosNprintMode” on page 21-44 for the allowed values.

**4906 POSERR\_PRN\_INVALID\_CR\_LF\_DISTANCE**

**Explanation:** An incorrect value was specified for the **PosNlineFeedCR** resource. Make sure that you typed the value correctly. See “PosNlineFeedCR” on page 21-40 for the allowed values.

**4907 POSERR\_PRN\_INVALID\_SJ\_LF\_DISTANCE**

**Explanation:** An incorrect value was specified for the **PosNlineFeedSJ** resource. Make sure that you typed the value correctly. See “PosNlineFeedSJ” on page 21-41 for the allowed values.

**4908 POSERR\_PRN\_INVALID\_DI\_LF\_DISTANCE**

**Explanation:** An incorrect value was specified for the **PosNlineFeedDI** resource. Make sure that you typed the value correctly. See “PosNlineFeedDI” on page 21-40 for the allowed values.

**4909 POSERR\_PRN\_INVALID\_FEED\_DIRECTION**

**Explanation:** An incorrect value was specified for the **PosNfeedDirection** resource. Make sure that you typed the value correctly. See “PosNfeedDirection” on page 21-36 for the allowed values.

**4910 POSERR\_PRN\_INVALID\_FISCAL\_NOTIFY**

**Explanation:** An incorrect value was specified for the **PosNfiscalNotify** resource. Make sure that you typed the value correctly. See “PosNfiscalNotify” on page 21-37 for the allowed values.

**4911 POSERR\_PRN\_INVALID\_DI\_ORIENTATION**

**Explanation:** An incorrect value was specified for the **PosNdiOrientation** resource. Make sure that you typed the value correctly. See “PosNdiOrientation” on page 21-35 for the allowed values.

**4912 POSERR\_PRN\_INVALID\_LEFT\_MARGIN\_CR**

**Explanation:** An incorrect value was specified for the **PosNleftMarginCR** resource. Make sure that you typed the value correctly. See “PosNleftMarginCR” on page 21-39 for the allowed values.

**4913 POSERR\_PRN\_INVALID\_PRINT\_ALIGNMENT**

**Explanation:** An incorrect value was specified for the **PosNprintAlignment** resource. Make sure that you typed the value correctly. See “PosNprintAlignment” on page 21-42 for the allowed values.

**4914 POSERR\_PRN\_INCORRECT\_DATA\_FORMAT**

**Explanation:** An incorrect data format was provided on the *PosWrite()* subroutine call during one of the following commands:

- Writing a logo (See “Writing Data in Logo Mode” on page 12-18 for more information). Writing a logo with an invalid parameter for the dot density.
- Downloading a pre-defined message (See “Writing Data in Download Message Mode (4610 SureMark Printers Only)” on page 12-19 for more information.)
- Downloading a pre-defined logo (See “Writing Data in Download Logo Mode” on page 12-20 for more information.)

#### 4915 POSERR\_PRN\_LOGO\_EXISTS

**Explanation:** The logo is already defined in the printer. For the IBM 4689 Model 3x1 and TD5 printers, erase the logo using the keypad on the printer. retest the IBM Point of Sale Subsystem, and then retry the command. See “Writing Data in Download Logo Mode” on page 12-20 for more information.

#### 4916 POSERR\_PRN\_UDC\_CHARACTER\_EXISTS

**Explanation:** The UDC character is already defined in the printer. For the IBM 4689 Model 3x1 and TD5 printers, erase the UDC characters using the keypad on the printer and then retry the command. See “User-Defined Characters” on page 12-44 for more information.

#### 4918 POSERR\_PRN\_INVALID\_QUALITY\_MODE

**Explanation:** An incorrect value was specified for the **PosNprintQualityMode** resource. Make sure that you typed the value correctly. See “PosNprintQualityMode” on page 21-44 for the allowed values.

#### 4919 POSERR\_PRN\_INVALID\_UPSIDE\_DOWN\_MODE

**Explanation:** An incorrect value was specified for the **PosNprintUpsideDown** resource. Make sure that you typed the value correctly. See “PosNprintUpsideDown” on page 21-50 for the allowed values.

#### 4920 POSERR\_PRN\_INVALID\_TABSTOPS\_FORMAT

**Explanation:** An incorrect value was specified for the **PosNprintTabStops** resource. Make sure that you typed the value correctly. See “PosNprintTabStops” on page 21-47 for the allowed values.

#### 4921 POSERR\_PRN\_INVALID\_COLOR\_MODE

**Explanation:** An incorrect value was specified for the **PosNprintColorMode** resource. Make sure that you typed the value correctly. See “PosNprintColorMode” on page 21-42 for the allowed values.

#### 4922 POSERR\_PRN\_INVALID\_TONE\_VOLUME

**Explanation:** An incorrect value was specified for the **PosNprintToneVolume** resource. Make sure that you typed the value correctly. See “PosNprintToneVolume” on page 21-49 for the allowed values.

**4923 POSERR\_PRN\_INVALID\_TONE\_DURATION**

**Explanation:** An incorrect value was specified for the **PosNprintToneDuration** resource. Make sure that you typed the value correctly. See “PosNprintToneDuration” on page 21-48 for the allowed values.

**4924 POSERR\_PRN\_INVALID\_TONE\_NOTE**

**Explanation:** An incorrect value was specified for the **PosNprintToneNote** resource. Make sure that you typed the value correctly. See “PosNprintToneNote” on page 21-48 for the allowed values.

**4925 POSERR\_PRN\_INVALID\_TONE\_OCTAVE**

**Explanation:** An incorrect value was specified for the **PosNprintToneOctave** resource. Make sure that you typed the value correctly. See “PosNprintToneOctave” on page 21-49 for the allowed values.

**4926 POSERR\_PRN\_INVALID\_TONE\_FREQUENCY**

**Explanation:** An incorrect value was specified for the **PosNprintToneFrequency** resource. Make sure that you typed the value correctly. See “PosNprintToneFrequency” on page 21-48 for the allowed values.

**4927 POSERR\_PRN\_INVALID\_CODE\_PAGE**

**Explanation:** An incorrect value was specified for the **PosNcodePage** resource. Make sure that you typed the value correctly. See “PosNcodePage” on page 21-33 for the allowed values.

**5401 POSERR\_CDR\_INVALID\_PULSE\_WIDTH**

**Explanation:** An incorrect value was specified for the **PosNpulseWidth** resource. Make sure that you typed the value correctly. See “PosNpulseWidth” on page 21-21 for the allowed values.

**5702 POSERR\_SCAN\_INVALID\_BAR\_CODES\_1**

**Explanation:** An incorrect value was specified for the **PosNbarCodes1** resource. Make sure that you typed the value correctly. See “PosNbarCodes1” on page 21-62 for the allowed values.

**5703 POSERR\_SCAN\_INVALID\_BAR\_CODES\_2**

**Explanation:** An incorrect value was specified for the **PosNbarCodes2** resource. Make sure that you typed the value correctly. See “PosNbarCodes2” on page 21-64 for valid values.

**5704 POSERR\_SCAN\_INVALID\_BEEP\_FREQ**

**Explanation:** An incorrect value was specified for the **PosNbeepFreq** resource. Make sure that you typed the value correctly. See “PosNbeepFreq” on page 21-67 for valid values.

**5705 POSERR\_SCAN\_INVALID\_BEEP\_LENGTH**

**Explanation:** An incorrect value was specified for the **PosNbeepLength** resource. Make sure that you typed the value correctly. See “PosNbeepLength” on page 21-67 for valid values.

**5706 POSERR\_SCAN\_INVALID\_BEEP\_STATE**

**Explanation:** An incorrect value was specified for the **PosNbeepState** resource. Make sure that you typed the value correctly. See “PosNbeepState” on page 21-68 for the allowed values.

**5707 POSERR\_SCAN\_INVALID\_BEEP\_VOLUME**

**Explanation:** An incorrect value was specified for the **PosNbeepVolume** resource. Make sure that you typed the value correctly. See “PosNbeepVolume” on page 21-68 for valid values.

**5708 POSERR\_SCAN\_INVALID\_BLINK\_LENGTH**

**Explanation:** An incorrect value was specified for the **PosNblinkLength** resource. Make sure that you typed the value correctly. See “PosNblinkLength” on page 21-69 for valid values.

**5709 POSERR\_SCAN\_INVALID\_BLOCK\_READ\_MODE**

**Explanation:** An incorrect value was specified for the **PosNblockReadMode** resource. Make sure that you typed the value correctly. See “PosNblockReadMode” on page 21-69 for the allowed values.

**5710 POSERR\_SCAN\_INVALID\_BLOCK\_1\_TYPE**

**Explanation:** An incorrect value was specified for the **PosNblock1Type** resource. Make sure that you typed the value correctly. See “PosNblock1Type” on page 21-70 for the allowed values.

**5711 POSERR\_SCAN\_INVALID\_BLOCK\_2\_TYPE**

**Explanation:** An incorrect value was specified for the **PosNblock2Type** resource. Make sure that you typed the value correctly. See “PosNblock2Type” on page 21-71 for the allowed values.

**5712 POSERR\_SCAN\_INVALID\_BLOCK\_3\_TYPE**

**Explanation:** An incorrect value was specified for the **PosNblock3Type** resource. Make sure that you typed the value correctly. See “PosNblock3Type” on page 21-71 for the allowed values.

**5713 POSERR\_SCAN\_INVALID\_CHECK\_MODULO**

**Explanation:** An incorrect value was specified for the **PosNcheckModulo** resource. Make sure that you typed the value correctly. See “PosNcheckModulo” on page 21-72 for the allowed values.

**5714 POSERR\_SCAN\_INVALID\_D\_READ\_TIMEOUT**

**Explanation:** An incorrect value was specified for the **PosNdReadTimeout**

resource. Make sure that you typed the value correctly. See “PosNdReadTimeout” on page 21-74 for allowed values.

## 5715 POSERR\_SCAN\_INVALID\_D\_TOUCH\_MODE

**Explanation:** An incorrect value was specified for the **PosNdTouchMode** resource. Make sure that you typed the value correctly. See “PosNdTouchMode” on page 21-74 for the allowed values.

## 5716 POSERR\_SCAN\_INVALID\_ITF\_LENGTH\_1

**Explanation:** An incorrect value was specified for the **PosNiTFLength1** resource. Make sure that you typed the value correctly. See “PosNiTFLength1” on page 21-76 for valid values.

## 5717 POSERR\_SCAN\_INVALID\_ITF\_LENGTH\_2

**Explanation:** An incorrect value was specified for the **PosNiTFLength2** resource. Make sure that you typed the value correctly. See “PosNiTFLength2” on page 21-77 for valid values.

## 5718 POSERR\_SCAN\_INVALID\_LASER\_TIMEOUT

**Explanation:** An incorrect value was specified for the **PosNlaserTimeout** resource. Make sure that you typed the value correctly. See “PosNlaserTimeout” on page 21-79 for valid values.

## 5719 POSERR\_SCAN\_INVALID\_MOTOR\_TIMEOUT

**Explanation:** An incorrect value was specified for the **PosNmotorTimeout** resource. Make sure that you typed the value correctly. See “PosNmotorTimeout” on page 21-80 for valid values.

## 5720 POSERR\_SCAN\_INVALID\_LASER\_SWITCH\_STATE

**Explanation:** An incorrect value was specified for the **PosNlaserSwitchState** resource. Make sure that you typed the value correctly. See “PosNlaserSwitchState” on page 21-79 for valid values.

## 5721 POSERR\_SCAN\_INVALID\_SCANS\_PER\_READ

**Explanation:** An incorrect value was specified for the **PosNscansPerRead** resource. Make sure that you typed the value correctly. See “PosNscansPerRead” on page 21-81 for valid values.

## 5722 POSERR\_SCAN\_LABEL\_TOO\_SHORT

**Explanation:** The raw data received from the scanner either contained no label data, or the length of the label data is less than the expected length for the label type.

## 5725 POSERR\_SCAN\_INVALID\_BVOL\_SWITCH\_STATE

**Explanation:** An incorrect value was specified for the **PosNbVolSwitchState** resource. Make sure that you typed the value correctly. See “PosNbVolSwitchState” on page 21-72 for valid values.

**5726 POSERR\_SCAN\_INVALID\_DECODE\_ALGORITHM**

**Explanation:** An incorrect value was specified for the **PosNdecodeAlgorithm** resource. Make sure that you typed the value correctly. See “PosNdecodeAlgorithm” on page 21-73 for valid values.

**5727 POSERR\_SCAN\_INVALID\_EAN13\_SCANS\_PER\_READ**

**Explanation:** An incorrect value was specified for the **PosNeAN13ScansPerRead** resource. Make sure that you typed the value correctly. See “PosNeAN13ScansPerRead” on page 21-75 for valid values.

**5728 POSERR\_SCAN\_INVALID\_EAN8\_SCANS\_PER\_READ**

**Explanation:** An incorrect value was specified for the **PosNeAN8ScansPerRead** resource. Make sure that you typed the value correctly. See “PosNeAN8ScansPerRead” on page 21-75 for valid values.

**5729 POSERR\_SCAN\_INVALID\_STORE\_SCANS\_PER\_READ**

**Explanation:** An incorrect value was specified for the **PosNstoreScansPerRead** resource. Make sure that you typed the value correctly. See “PosNstoreScansPerRead” on page 21-81 for valid values.

**5730 POSERR\_SCAN\_INVALID\_UPCA\_SCANS\_PER\_READ**

**Explanation:** An incorrect value was specified for the **PosNuPCAScansPerRead** resource. Make sure that you typed the value correctly. See “PosNuPCAScansPerRead” on page 21-87 for valid values.

**5731 POSERR\_SCAN\_INVALID\_UPCD\_SCANS\_PER\_READ**

**Explanation:** An incorrect value was specified for the **PosNuPCDScansPerRead** resource. Make sure that you typed the value correctly. See “PosNuPCDScansPerRead” on page 21-87 for valid values.

**5732 POSERR\_SCAN\_INVALID\_UPCE\_SCANS\_PER\_READ**

**Explanation:** An incorrect value was specified for the **PosNuPCEScansPerRead** resource. Make sure that you typed the value correctly. See “PosNuPCEScansPerRead” on page 21-88 for valid values.

**5733 POSERR\_SCAN\_INVALID\_UPC\_EXPANSION**

**Explanation:** An incorrect value was specified for the **PosNuPCExpansion** resource. Make sure that you typed the value correctly. See “PosNuPCExpansion” on page 21-88 for valid values.

**5734 POSERR\_SCAN\_INVALID\_VERIFY\_PRICE\_CHK**

**Explanation:** An incorrect value was specified for the **PosNverifyPriceChk** resource. Make sure that you typed the value correctly. See “PosNverifyPriceChk” on page 21-89 for valid values.

**5735 POSERR\_SCAN\_INVALID\_QUEUE\_ALL\_INDICATOR**

**Explanation:** An incorrect value was specified for the

**PosNqueueAllLabels** resource. Make sure that you typed the value correctly. See “PosNqueueAllLabels” on page 21-80 for valid values.

## 5736 POSERR\_SCAN\_CONFIGURATION\_ERROR

**Explanation:** The current scanner device configuration does not match the configuration specified by the scanner device driver.

This error code applies to the following scanners:

- IBM 4696 Point of Sale Scanner Scale
- IBM 4697 Point of Sale Scanner
- IBM 4698 Point of Sale Scanner

## 5737 POSERR\_SCAN\_2\_LABEL\_FLAG\_CONFIG\_ERROR

**Explanation:** The current scanner device two-label flag pair configuration does not match the two-label flag pair configuration specified by the scanner device driver.

This error code applies to the following scanners:

- IBM 4697 Point of Sale Scanner
- IBM 4698 Point of Sale Scanner

## 5738 POSERR\_SCAN\_INVALID\_2\_LABEL\_DECODE\_STATE

**Explanation:** The value for the **PosNjANTwoLabelDecode** resource is not valid. See “PosNjANTwoLabelDecode” on page 21-78 for valid values.

## 5739 POSERR\_SCAN\_INVALID\_FLAG\_PAIR\_COMBINATION

**Explanation:** The flag pairs specified by **PosNtwoLabelFlagPair1**, **PosNtwoLabelFlagPair2**, **PosNtwoLabelFlagPair3**, and **PosNtwoLabelFlagPair4** are not valid in combination. The pairs must be specified such that no first flag value matches any of the second flag values. See “PosNtwoLabelFlagPair1” on page 21-83 for information on specifying these flag pairs.

## 5740 POSERR\_SCAN\_INVALID\_2\_LABEL\_FLAG\_PAIR\_1

**Explanation:** The value for the **PosNtwoLabelFlagPair1** resource is not valid. See “PosNtwoLabelFlagPair1” on page 21-83 for valid values.

## 5741 POSERR\_SCAN\_INVALID\_2\_LABEL\_FLAG\_PAIR\_2

**Explanation:** The value for the **PosNtwoLabelFlagPair2** resource is not valid. See “PosNtwoLabelFlagPair2” on page 21-84 for valid values.

## 5742 POSERR\_SCAN\_INVALID\_2\_LABEL\_FLAG\_PAIR\_3

**Explanation:** The value for the **PosNtwoLabelFlagPair3** resource is not valid. See “PosNtwoLabelFlagPair3” on page 21-85 for valid values.

## 5743 POSERR\_SCAN\_INVALID\_2\_LABEL\_FLAG\_PAIR\_4

**Explanation:** The value for the **PosNtwoLabelFlagPair4** resource is not valid. See “PosNtwoLabelFlagPair4” on page 21-86 for valid values.

**5744 POSERR\_SCAN\_INVALID\_CODE39\_SCANS\_PER\_READ**

**Explanation:** The value for the **PosNcode39ScansPerRead** resource is not valid. See “PosNcode39ScansPerRead” on page 21-73 for valid values.

**5745 POSERR\_SCAN\_INVALID\_INT2OF5\_SCANS\_PER\_READ**

**Explanation:** The value for the **PosNiTFScansPerRead** resource is not valid. See “PosNiTFScansPerRead” on page 21-78 for valid values.

**5746 POSERR\_SCAN\_INVALID\_CODE128\_SCANS\_PER\_READ**

**Explanation:** The value for the **PosNcode128ScansPerRead** resource is not valid. See “PosNcode128ScansPerRead” on page 21-73 for valid values.

**5747 POSERR\_SCAN\_INVALID\_BAR\_CODES\_3**

**Explanation:** An incorrect value was specified for the **PosNbarCodes3** resource. Make sure that you typed the value correctly. See “PosNbarCodes3” on page 21-65 for the allowed values.

**5748 POSERR\_SCAN\_INVALID\_BAR\_CODES\_4**

**Explanation:** An incorrect value was specified for the **PosNbarCodes4** resource. Make sure that you typed the value correctly. See “PosNbarCodes4” on page 21-66 for valid values.

**5750 POSERR\_SCAN\_INVALID\_ITF\_LENGTH\_TYPE**

**Explanation:** An incorrect value was specified for the **PosNiTFLengthType** resource. Make sure that you typed the value correctly. See “PosNiTFLengthType” on page 21-77 for valid values.

**5751 POSERR\_SCAN\_INVALID\_SUPPLEMENTALS**

**Explanation:** An incorrect value was specified for the **PosNsupplementals** resource. Make sure that you typed the value correctly. See “PosNsupplementals” on page 21-82 for valid values.

**5752 POSERR\_SCAN\_INVALID\_BARCODE\_PROG\_STATE**

**Explanation:** An incorrect value was specified for the **PosNbarCodeProgramming** resource. Make sure that you typed the value correctly. See “PosNbarCodeProgramming” on page 21-67 for valid values.

**5753 POSERR\_SCAN\_INVALID\_XMIT\_CHECK\_DIGIT**

**Explanation:** An incorrect value was specified for the **PosNtransmitCheckDigit** resource. Make sure that you typed the value correctly. See “PosNtransmitCheckDigit” on page 21-82 for valid values.

**5901 POSERR\_RS232\_INVALID\_BAUD\_RATE**

**Explanation:** An incorrect value was specified for the **PosNbaudRate** resource. Make sure that you typed the value correctly. See “PosNbaudRate” on page 21-53 for the allowed values.

**5902 POSERR\_RS232\_INVALID\_STOP\_BITS**

**Explanation:** An incorrect value was specified for the **PosNstopBits** resource. Make sure that you typed the value correctly. See “PosNstopBits” on page 21-55 for the allowed values.

**5903 POSERR\_RS232\_INVALID\_PARITY**

**Explanation:** An incorrect value was specified for the **PosNparity** resource. Make sure that you typed the value correctly. See “PosNparity” on page 21-55 for the allowed values.

**5904 POSERR\_RS232\_INVALID\_DATA\_BITS**

**Explanation:** An incorrect value was specified for the **PosNdataBits** resource. Make sure that you typed the value correctly. See “PosNdataBits” on page 21-54 for the allowed values.

**5905 POSERR\_RS232\_INVALID\_TIMEOUT\_CHAR**

**Explanation:** An incorrect value was specified for the **PosNtimeoutChar** resource. Make sure that you typed the value correctly. See “PosNtimeoutChar” on page 21-56 for the allowed values.

**5907 POSERR\_RS232\_PREV\_NOT\_COMPLETE**

**Explanation:** The application has issued a *PosWrite()* subroutine call when the previous *PosWrite()* has not completed. Wait for event message POSM\_RS232\_XMIT\_COMPLETE, or POSM\_RS232\_XMIT\_ABORT before trying the write operation again.

**6201 POSERR\_SCALE\_INVALID\_OPERATIONS\_MODE**

**Explanation:** An incorrect value was specified for the **PosNoperMode** resource. Make sure that you typed the value correctly. See “PosNoperMode” on page 21-58 for valid values.

**6202 POSERR\_SCALE\_INVALID\_REMOTE\_DISPLAY\_STATE**

**Explanation:** An incorrect value was specified for the **PosNdisplayRequired** resource. Make sure that you typed the value correctly. See “PosNdisplayRequired” on page 21-57 for valid values.

**6203 POSERR\_SCALE\_INVALID\_VIBRATION\_FILTER**

**Explanation:** An incorrect value was specified for the **PosNvibrationFilter** resource. Make sure that you typed the value correctly. See “PosNvibrationFilter” on page 21-58 for valid values.

**6204 POSERR\_SCALE\_INVALID\_WEIGHT\_MODE**

**Explanation:** An incorrect value was specified for the **PosNweightMode** resource. Make sure that you typed the value correctly. See “PosNweightMode” on page 21-59 for valid values.

**6205 POSERR\_SCALE\_INVALID\_ZERO\_INDICATOR\_STATE**

**Explanation:** An incorrect value was specified for the **PosNzeroIndState**

resource. Make sure that you typed the value correctly. See “PosNzeroIndState” on page 21-59 for valid values.

## 6206 POSERR\_SCALE\_INVALID\_ZERO\_RETURN\_STATE

**Explanation:** An incorrect value was specified for the **PosNzeroRetState** resource. Make sure that you typed the value correctly. See “PosNzeroRetState” on page 21-60 for valid values.

## 6207 POSERR\_SCALE\_ZERO\_SCALE\_FAILED

**Explanation:** The POS\_SCALE\_ZERO\_SCALE *PosIOctl()* request was not successful.

## 6208 POSERR\_SCALE\_INVALID\_CLEAR\_SCREEN\_REQUEST

**Explanation:** The POS\_SCALE\_CLEAR\_SCREEN *PosIOctl()* request is not valid in the current scale configuration. The **PosNoperMode** resource must be PosUK.

## 6209 POSERR\_SCALE\_CONFIGURATION\_ERROR

**Explanation:** The current scale device configuration does not match the configuration specified by the scale device handler. This error code does not apply to the IBM 4687 Point of Sale Scanner Model 2.

## 6211 POSERR\_SCALE\_INVALID\_NUM\_WEIGHT\_DIGITS

**Explanation:** An incorrect value was specified for the **PosNnumWeightDigits** resource. Make sure that you typed the value correctly. See “PosNnumWeightDigits” on page 21-57 for valid values.

## 6401 POSERR\_POWER\_INVALID\_DAY

**Explanation:** The application specified a value that is not valid for the day of the month. See “PosNpowerAlarm” on page 21-31 for more information.

## 6402 POSERR\_POWER\_INVALID\_HOUR

**Explanation:** The application specified a value that is not valid for the hour. The value must be from 00 to 23. See “PosNpowerAlarm” on page 21-31 for more information.

## 6403 POSERR\_POWER\_INVALID\_MINUTES

**Explanation:** The application specified a value that is not valid for the minutes. The value must be from 00 to 59. See “PosNpowerAlarm” on page 21-31 for more information.

## 6701 POSERR\_TOUCH\_INVALID\_BACKLIGHT\_ON

**Explanation:** An incorrect value was specified for the **PosNtouchBackLightOnEvent** resource. Make sure that you typed the value correctly. See “PosNtouchBackLightOnEvent” on page 21-90 for the allowed values.

**6702 POSERR\_TOUCH\_INVALID\_CLICK\_VOLUME**

**Explanation:** An incorrect value was specified for the **PosNtouchClickVolume** resource. Make sure that you typed the value correctly. See “PosNtouchClickVolume” on page 21-91 for the allowed values.

**6703 POSERR\_TOUCH\_INVALID\_CONTRAST**

**Explanation:** An incorrect value was specified for the **PosNtouchContrast** resource. Make sure that you typed the value correctly. See “PosNtouchContrast” on page 21-91 for the allowed values.

**6704 POSERR\_TOUCH\_INVALID\_ENTRY\_CLICK**

**Explanation:** An incorrect value was specified for the **PosNtouchEntryClick** resource. Make sure that you typed the value correctly. See “PosNtouchEntryClick” on page 21-91 for the allowed values.

**6705 POSERR\_TOUCH\_INVALID\_EXIT\_CLICK**

**Explanation:** An incorrect value was specified for the **PosNtouchExitClick** resource. Make sure that you typed the value correctly. See “PosNtouchExitClick” on page 21-92 for the allowed values.

**6706 POSERR\_TOUCH\_INVALID\_MODE**

**Explanation:** An incorrect value was specified for the **PosNtouchMode** resource. Make sure that you typed the value correctly. See “PosNtouchMode” on page 21-92 for the allowed values.

**6707 POSERR\_TOUCH\_INVALID\_SCREEN\_SAVER\_TIME**

**Explanation:** An incorrect value was specified for the **PosNtouchScreenSaverTime** resource. Make sure that you typed the value correctly. See “PosNtouchScreenSaverTime” on page 21-93 for the allowed values.

**6708 POSERR\_TOUCH\_INVALID\_TONE\_DURATION**

**Explanation:** An incorrect value was specified for the **PosNtouchToneDuration** resource. Make sure that you typed the value correctly. See “PosNtouchToneDuration” on page 21-93 for the allowed values.

**6709 POSERR\_TOUCH\_INVALID\_TONE\_FREQUENCY**

**Explanation:** An incorrect value was specified for the **PosNtouchToneFreq** resource. Make sure that you typed the value correctly. See “PosNtouchToneFreq” on page 21-93 for the allowed values.

**6710 POSERR\_TOUCH\_INVALID\_TONE\_VOLUME**

**Explanation:** An incorrect value was specified for the **PosNtouchToneVolume** resource. Make sure that you typed the value correctly. See “PosNtouchToneVolume” on page 21-94 for the allowed values.

## **6711 POSERR\_TOUCH\_INVALID\_BRIGHTNESS**

**Explanation:** An incorrect value was specified for the **PosNtouchBrightness** resource. Make sure that you typed the value correctly. See “PosNtouchBrightness” on page 21-90 for the allowed values.



---

## Appendix E. IBM Model 4A Font Download

When the IBM Point of Sale Subsystem recognizes a Model 4A printer that it has not used before, it will automatically download to the printer one of the character sets described below:

<b>File</b>	<b>Description</b>
<b>M4A00850.FON</b>	The single-byte characters are the same as those in the Model 4 printer. The characters in this font file are used for the following code pages: <ul style="list-style-type: none"><li>• 437</li><li>• 850</li><li>• 852</li><li>• 857</li><li>• 860</li><li>• 861</li><li>• 862</li><li>• 863</li><li>• 864</li><li>• 865</li></ul>
<b>M4A00932.FON</b>	The double-byte characters for Japan. This character set is a subset of the characters in code page 932. For the supported characters, see the <i>IBM Point of Sale Subsystem: Installation, Keyboards, and Code Pages</i> publication.
<b>M4A00949.FON</b>	The double-byte characters for Korea. This character set is a subset of the characters in code page 949. For the supported characters, see the <i>IBM Point of Sale Subsystem: Installation, Keyboards, and Code Pages</i> publication.

---

### Manually Downloading Characters

If at a later time, you want to use another character set, or if you want to define your own characters and download them to the Model 4A printer, you can do so manually. The program to download a new character set to the Model 4A printer is AIPM4ALD.EXE. This program is in the \BIN directory where the IBM Point of Sale Subsystem is installed.

Use the following syntax to run the AIPM4ALD program:

```
AIPM4ALD -Sslot -Pport -Fx:\dir\font-file [ -Q ]
```

The parameters for the AIPM4ALD program are:

<b>Parameter</b>	<b>Description</b>
<b>-S slot</b>	The slot number (in decimal) of the IBM Model 4A printer to which the new character set is to be downloaded.
<b>-P port</b>	The port number (in decimal) of the IBM Model 4A printer to which the new character set is to be downloaded.

<b>-F</b> <i>x:\dir\font-file</i>	The name of the file containing the characters to be downloaded to the IBM Model 4A printer. A fully-qualified path name must be specified when this program is run manually.
<b>-Q</b>	Quiet Mode. Do not display progress information
<b>-V</b>	Verify Characters. Verify each character by printing it on the CR station after all the characters have been downloaded.

---

## Font File Format

There are three types of records that are allowed in a Model 4A font file. These are:

- Comment Record
- Keyword Record
- Character Definition Record

A *comment record* can be either a blank line, or a line with an exclamation point (!) as the first non-blank character.

A *keyword record* is a line with one of the keywords defined below followed by one or more blank characters and ending with a value for the keyword.

A *character definition record* is a line of hexadecimal ASCII characters. Each pair of characters defines one byte of the character definition. The first two pairs of characters identify the character being defined and the remaining characters define the character.

## Keywords

To define characters to the IBM Point of Sale Subsystem for use with the Model 4A printer, the following keywords and their corresponding values must be defined in the font file along with the definition of each character. If any of these keywords are omitted, unpredictable results can occur when your application is using the printer.

<b>Keyword</b>	<b>Description</b>
<b>CODEPAGE</b>	The code page this font file defines.
<b>HEIGHT</b>	Defines the double-byte character height. The valid values are 9 or 16.
<b>MODE</b>	Defines this code page to be a single-byte, or a double-byte code page. A value of 0 (zero) indicates this is a single-byte code page and a value of 1 (one) indicates this is a double-byte code page.
<b>SPACE</b>	The number of dot columns to be added to each double-byte character. The valid values are 0 (zero) through 16.
<b>WIDTH</b>	Defines the double-byte character width. The valid values are 10 through 16.

## Character Definition Record

Each *character definition record* defines one character for the Model 4A printer. Each character on the line must be a hexadecimal ASCII character. Each pair of characters defines one byte of the character definition. The first two pairs of characters identify the character being defined and the remaining characters define the character.

### Character Identifier

The first two pairs of characters that identify the character being defined must be between 0x0000 and 0x00FF, or between 0x8000 and 0xFFFF. The characters in the range 0x0000 through 0x00FF define the single-byte characters and there must be 44 hexadecimal ASCII characters (22 bytes). The characters in the range 0x8000 through 0xFFFF define the double-byte characters and the number of hexadecimal ASCII characters depends upon the width of the character being defined. The number of characters in the line can be calculated by multiplying the character width, as defined by the WIDTH keyword, by 4 and adding 4. The additional four characters are for the character identifier.

**Note:** If you define double-byte characters in your font file, you must define the character 0x8000 to be all zeros. This is because the IBM Point of Sale Subsystem uses that character for padding double-byte character print lines when positioning the print head over characters already printed.

### Character Definition

Each dot column on the Model 4A printer has nine print wires for single-byte characters and between nine to sixteen print wires for double-byte characters.

The definition of each character follows the character identifier on each line. Each pair of hexadecimal ASCII values defines one byte of the character. For the Model 4A printer, each pair of bytes (four characters) defines which print wires are turned on for each dot column to be printed. Within each pair, the first two hexadecimal ASCII characters (first byte) defines which of the lower eight print wires will be turned on and the second two hexadecimal ASCII characters (second byte) defines which of the upper eight print wires will be turned on. Within each byte, the least significant bit represents the top print wire.

For single-byte characters, the least significant bit of the first byte in the pair of dot column bytes represents print wire number nine. For single-byte characters, this is the only bit used in the first byte. For double-byte characters, the number of significant bits is determined by the LENGTH keyword. If the LENGTH keyword is set to 9, the double-byte character definition is the same as the single-byte character. If the LENGTH keyword is set to 16, then all eight bits of the first byte is used to define the print wires 9 through 16.

### Character Definition Restrictions

The following should be considered when defining characters for the Model 4A printer:

- When defining characters, the same print wire should not be turned on in consecutive dot columns.
- For single-byte characters, the spacing between characters is included in the character definition.
- For double-byte characters, the spacing between characters can be either included in the character definition, or defined by the SPACE keyword.



## Appendix F. Downloading fonts

USB Character/Graphics Display . . . . .	F-1
Downloading Fonts to the Display . . . . .	F-1
IBM 4689 SurePOS Receipt Journal Printer . . . . .	F-2
Converting Printer Font Files . . . . .	F-2
Downloading Fonts to the Printer . . . . .	F-2
4610 SureMark Point of Sale Printer Model TI5, TM7, and TF7 . . . . .	F-3
Converting Printer Font Files . . . . .	F-3
Downloading Fonts to the Printer . . . . .	F-4
Other 4610 SureMark Point of Sale Printer Models . . . . .	F-4
Converting Printer Font Files . . . . .	F-4
Downloading Fonts to the Printer . . . . .	F-5

This section describes how to download fonts for the USB Character/Graphics display, the 4689 SurePos Receipt Journal Printer, and models of the 4610 SureMark POS printer.

### USB Character/Graphics Display

To download fonts to the USB Character/Graphics Display, you will need the following:

- The display font download program, AIPFNVFD.EXE, which comes with the IBM Point of Sale Subsystem for Windows.
- Font files for the USB Character/Graphics Display, which can be found in the IBM Point of Sale Subsystem for Windows BIN directory

Font File Name	Font File Description
jpm16.bin	Japan, Mincho style
korm16.bin	Korea, Mincho style
sbc0001.bin	SBCS fonts, updated version
Prcmz16.bin	PRC Simplified Chinese
Rocbg16b.bin	Taiwan Traditional Chinese

### Downloading Fonts to the Display

Font files for the USB Character/Graphics Display must be downloaded manually using the AIPFNVFD.EXE program

```
AIPFNVFD -Sslot -Pport {-Ndevice-number} -Fx:\dir\font-file {-Q}
```

**-Sslot** The slot number (in decimal) of the display for which the download is intended.

**-Pport** The port number (in decimal) of the display for which the download is intended.

**-Ndevice-number**

An indication of the device number of the display for which the download is intended. Valid values are:

- A : device number 2A (default)
- B : device number 2B
- C : device number 2C

- D : device number 2D

**-Fx:\dir\font-file**

Fully qualified path name for the font file to be downloaded to the display.

- Q** Quiet mode. No progress information is displayed

---

## IBM 4689 SurePOS Receipt Journal Printer

To download fonts to the IBM 4689 SurePOS Receipt Journal Printer, you will need the following:

- The 4689 Printer font download program, AIPFNTRJ.EXE, which comes with the IBM Point of Sale Subsystem for Windows.
- The 4689 Printer font conversion program, 4689CNVT.EXE, and IBM 4689 Printer font files, which can be downloaded from the IBM Retail Store Solutions Support site: [www.ibm.com/solutions/retail/store](http://www.ibm.com/solutions/retail/store):
  1. Select **Support** under Retail Store Solutions.
  2. Select **Other Systems and Adapters** under SurePOS Peripherals.
  3. Select **4689** under IBM POS Printers.
  4. Select **Font Support Files** to download the font files.

The download contains the following:

File Name	File Description
4689JPNM.EXE	Japan, Mincho style font (diskette image)
4689JPNG.EXE	Korea, Mincho style font (diskette image)
4689JPNU.EXE	Japan, user-defined fonts (diskette image)
CONVERT.EXE	Contains: <ul style="list-style-type: none"> <li>• 4689CNVT.EXE : conversion program</li> <li>• jpngs932.cfg: Japan, gothic style + user-defined fonts font configuration file</li> <li>• jpnms932.cfg: Japan, mincho style + user-defined fonts font configuration file</li> <li>• readme.txt: the directions given here</li> </ul>
readme.txt	The information presented in this section.

## Converting Printer Font Files

Convert the downloaded files using the 4689CNVT.EXE program.

4689CNVT configuration-file

**configuration-file**

One of the configuration files contained in CONVERT.EXE

**Notes:**

1. The font files must be in the same directory as the conversion program.
2. The output font file will have the same name as the supplied configuration file, but will have a file extension of .FON.

## Downloading Fonts to the Printer

Font files for the IBM 4689 SurePOS Receipt Journal Printer must be downloaded manually using the AIPFNTRJ.EXE program:

AIPFNTRJ -Sslot -Pport -Fx:\dir\font-file {-Q}

**-Sslot :**

The slot number (in decimal) of the printer for which the download is intended.

**-Pport :**

The port number (in decimal) of the printer for which the download is intended.

**-Fx:\dir\font-file :**

Fully qualified path name for the font file to be downloaded to the printer.

**-Q :** Quiet mode. No progress information is displayed.

---

## 4610 SureMark Point of Sale Printer Model TI5, TM7, and TF7

To download fonts to the IBM SureMark Point of Sale Printer Model TI5, TM7, and TF7, you will need the following:

- The 4610 SureMark Point of Sale Printer font download program, AIPFNT46.EXE, which comes with the IBM Point of Sale Subsystem for Windows.
- Font files for the IBM 4610 SureMark Point of Sale Printer Model TI5 and the IBM 4610 SureMark Point of Sale Printer Model TI5 font conversion program, 4610CNVT.EXE, which can be downloaded from the IBM Retail Store Solutions Support site: [www.ibm.com/solutions/retail/store](http://www.ibm.com/solutions/retail/store)
  1. Select **Support** under Retail Store Solutions
  2. Select **IBM SureMark Printer** under SurePOS Peripherals.
  3. Select **DBCS Models** under Downloads. From this page, you can download fonts for Japanese (Gothic and Micho styles plus user-defined fonts), Korean, and Chinese (Simplified and Traditional).
  4. Select **Driver Font Utility Package** to download the font conversion program and font configuration files.

This package contains the following:

File Name	File Description
4610CNVT.EXE	Font conversion program
chnm1381.cfg	Chinese Simplified (PRC) font - Code Page 1381
chnm950b.cfg	Traditional Chinese (BIG-5) with Bold SBCS impact (Taiwan) fonts - Code Page 950
jpngs932.cfg	Japanese Gothic Style font - Code Page 932
jpnms932.cfg	Japanese Mincho Style font - Code Page 932
korms949.cfg	Korean font - Code Page 949

## Converting Printer Font Files

Convert the downloaded files using the 4610CNVT.EXE program.

4610CNVT configuration-file

**configuration-file**

One of the configuration files contained in CONVERT.EXE.

**Notes:**

1. The font files must be in the same directory as the conversion program.

- The output font file will have the same name as the supplied configuration file, but will have a file extension of .FON .

## Downloading Fonts to the Printer

Font files for the IBM 4610 SureMark Point of Sale Printer Model T15 must be downloaded manually using the AIPFNT46.EXE program:

```
AIPFNT46 -Sslot -Pport -Fx:\dir\font-file {-Q}
```

- Sslot** The slot number (in decimal) of the printer for which the download is intended
- Pport** The port number (in decimal) of the printer for which the download is intended.
- Fx:\dir\font-file**  
Fully qualified path name for the font file to be downloaded to the printer.
- Q :** Quiet mode. No progress information is displayed.

---

## Other 4610 SureMark Point of Sale Printer Models

You will need the following to download fonts to the IBM SureMark Point of Sale Printers (other than the Model T15, TF7, and TM7). The 4610 models described in this section are T11, T12, T13, T14, TF6, TM6, and TN3.

- The 4610 SureMark Point of Sale Printer font download programs, which come with the IBM Point of Sale Subsystem:
  - AIPFNT46.EXE - for Windows and OS/2
  - aipfnt46 - for Linux
- For the Windows operating system, you will also need:
  - The 4610 SureMark Point of Sale Printer font conversion program, AIPFNTCT.EXE, which comes with the IBM Point of Sale Subsystem for Windows and with POS Device Diagnostics. You can also download this program, as well as the font files for the IBM 4610 SureMark Point of Sale Printer, from the IBM Retail Store Solutions Support site: [www.ibm.com/solutions/retail/store](http://www.ibm.com/solutions/retail/store).
  - 1. Select **Support** under Retail Store Solutions
  - 2. Select **IBM SureMark Printer** under SurePOS Peripherals.
  - 3. Select **Proportional Font Converter Package** to download the font conversion program.
  - 4. Select **IBM True Type Fonts** to download fonts.
  - 5. Select **Fonts and Logos Utility Diskette** to download a utility to create and edit fonts.

## Converting Printer Font Files

On Windows operating systems, convert the downloaded font files using the AIPFNTCT.EXE program. If you installed POS Device Diagnostics, find the shortcut to AIPFNTCT.EXE in **Start->Programs>Point of Sale Subsystem**.

### Notes:

- User-defined fonts created using the Fonts and Logos Utility do not require conversion.
- Some true type fonts may be on a Windows system; these files can also be converted to 4610 printer fonts using AIPFNTCT.EXE.

## Downloading Fonts to the Printer

You must manually download Font files for the IBM 4610 SureMark Point of Sale Printers using the IBM 4610 SureMark Point of Sale Printer font download program AIPFNT46.EXE (aipfnt46 on Linux). Use the following syntax to run the program:

```
aipfnt46 -Sslot -Pport -Ffontfilename -Q -Cx
```

- S slot**            The slot number (in decimal) of the printer for which the download is intended.
- P port**            The port number (in decimal) of the printer for which the download is intended.
- F fontfilename**   Fully qualified path name for the font file to be downloaded to the printer.
- Q**                 Quiet mode. No progress information is displayed.
- C x**                User-defined code page to be used for the proportional/user-defined fonts; valid values are:
  - 1**            One of the two proportional font locations; default, code page 1; CR station for user-defined
  - 2**            CR station for user-defined
  - 3**            One of the two proportional font locations; code page 3; CR station for user-defined
  - 4**            CR station for user-defined
  - 5**            DI station for user-defined
  - 6**            DI station for user-defined

### Notes:

1. Proportional fonts require two memory locations. For example, a proportional font in location 1 also fills location 2.
2. All fonts are erased when code page 1 is specified. Specify code page 1 first and then subsequent code pages.
3. The logfile, aipfnt46.log, contains the results of the last font download. On Windows and OS/2 systems, this file is in the LOG directory located in the IBM Point of Sale Subsystem root directory (default C:\POS); on Linux systems, this file is located in the /var/log directory.



---

## Appendix G. IBM 4610 Printer Firmware Update

At start-up, if the IBM Point of Sale Subsystem finds any of the following microcode update files:

- **aip46mc.hex**, for IBM 4610 SureMark Point of Sale printer models TI1 and TI2
- **aip46mch.hex**, for IBM 4610 SureMark Point of Sale printer models TI3 and TI4
- **aip46mcd.hex**, for IBM 4610 SureMark Point of Sale printer model TI5

Any attached 4610 SureMark Point of Sale printer is updated with the latest microcode:

- On Windows and OS/2 systems, the microcode update files must be in the BIN directory located in the IBM Point of Sale Subsystem root directory (default C:\POS);
- On Linux systems, microcode update files must be in the /share/pos directory located in the directory specified during Point of Sale Subsystem for Linux installation (default /usr).
- A log file named aipfld46.log is created when the firmware download program is executed. On Windows and OS/2 systems, this file is in the LOG directory located in the IBM Point of Sale Subsystem root directory (default C:\POS); on Linux systems, this file is located in the /var/log directory.

---

### Manually Updating Firmware

New microcode for the IBM 4610 SureMark Point of Sale printers can also be downloaded manually. The program to download new microcode is AIPFLD46.EXE (aipfld46 on Linux). The program is in the \BIN directory where the IBM Point of Sale Subsystem is installed. To download the microcode, ensure that the file containing the microcode data is located in the same directory as the program.

Use the following syntax to run the AIPFLD46 program:

```
aipfld46 -Sslot -Pport -Fmicrocodefilename [ -Q ]
```

The parameters for the AIPFLD46 program are:

Parameter	Description
<b>-S</b> <i>slot</i>	The slot number (in decimal) of the IBM 4610 SureMark Point of Sale printer for which the microcode download is intended.
<b>-P</b> <i>port</i>	The port number (in decimal) of the IBM 4610 SureMark Point of Sale printer for which the microcode download is intended.
<b>-F</b> <i>microcodefilename</i>	The name of the file containing the microcode update data for the IBM 4610 SureMark Point of Sale printer. A fully qualified path name must be specified when this program is run manually.
<b>-Q</b>	Quiet Mode. Do not display progress information.

**Example:**

## IBM 4610 Printer Firmware

created on October 2, 2001

```
slot = 8 (hex 8) (USB Subsystem)
port = 17 (hex 11)
file in c:\pos\bin = aip46mch.hex
```

The printer in the example above is an IBM 4610 SureMark Model TI3 printer. Enter the following on the command line:

```
aipfld46 -s8 -p17 -fc:\pos\bin\aip46mch.hex
```

---

## Appendix H. Firmware Update Utility for USB Devices

There is a mechanism to automatically update the firmware for all IBM point-of-sale USB devices. The firmware update occurs during subsystem initialization. It updates the USB devices based on Engineering Change (EC) levels in flash files. If the EC level for the device is lower than indicated in the file, the firmware is automatically updated.

The firmware update activity (successes and errors) is logged in a text log file called `aipflash.log` which is created in the `pos\log\` directory



## Appendix I. Using the IBM SureBase UPS Utility

The UPS utility notifies you that a condition relating to the uninterruptible power supply in the IBM SureBase has occurred. This utility can also check the SureBase UPS status at any time.

Use the following steps to start the UPS user interface:

1. From the Start menu, select **Programs**.
2. Select **IBM Point of Sale Subsystem**.
3. Select **aipupsui**.

The normal state of the UPS is that ac power is available. Under normal conditions, the UPS window is displayed showing an ac plug and a battery. See Figure I-1.

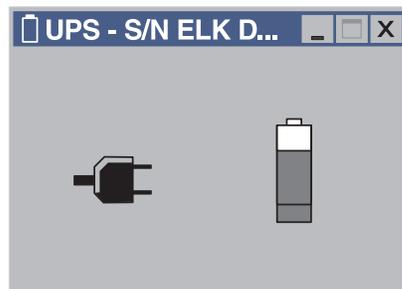


Figure I-1. Normal operation. Battery is charging or fully charged.

The normal condition for the UPS is fully charged but when the battery power goes to less than 25%, the UPS UI utility window is forced into the foreground one time indicating that the system will be operational for a limited amount of time.

The battery is shown 25% shaded to indicate the charge level and <25% is written under the battery. See Figure I-2.

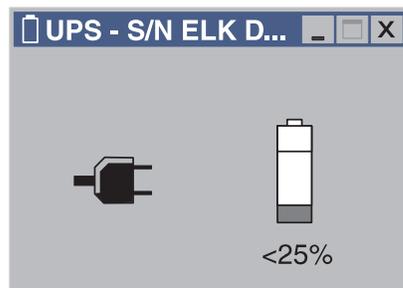
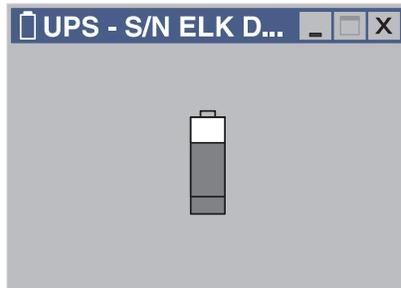


Figure I-2. Normal operation. Battery is less than 25% charged.

When the UPS UI utility is activated during a loss of ac power, the UPS UI window is forced into the foreground one time. The window only displays the battery bitmap at this point. See Figure I-3 on page I-2.



*Figure I-3. UPS is active due to the loss of A/C power.*

If the battery needs to be replaced, the UPS UI utility window is forced into the foreground and a screen is displayed with a flashing **black** battery. This window is displayed until it is closed or minimized.

In the event that the UPS becomes overloaded, the UPS UI utility window is forced into the foreground and displays a flashing **red** battery. To reduce the load on the UPS, turn off and unplug one or more of the less needed devices connected to the UPS.

---

## Appendix J. 4820 SurePoint Solution Touch Screen Calibration

This appendix describes the SurePoint Solution touch screen calibration for models with USB and RS-485 connections.

---

### USB Calibration

The IBM POS Device diagnostics tool allows you to calibrate the 4820 display when it is attached to the SurePOS 700 Series system unit. This tool installs as part of the POSS for Windows software.

---

### RS-485 Touch Screen Calibration Utility

The touch screen calibration tool, AIPTUTIL, calibrates a 4820 SurePoint Solution touch display. After installing and opening the tool, you can select from three tabs: **Calibrate**, **Click Settings**, and **Hardware**.

### Properties and methods of the tool

After installation, the configuration tool generates a dialog box, which prompts the user to calibrate the touchscreen for each video display setting. Thereafter, a monitoring process generates this dialog box whenever the screen resolution changes.

#### Configuration File

The configuration file, TOUCHSET.CFG, is a plain text file. This file stores the current screen settings and can be edited using any text editor. AIPTUTIL reads from and writes to this file. The following example shows the file contents:

```
ConnectionType = <string> !e.g. RS485
!  
RemindCalibrate = <integer> !1=True, 0=False  
Calibrated = <integer> !1=True, 0=False
```

#### ConnectionType

Specifies the type of connection for the touch screen. The only valid value for this field is RS-485; this value is also the default value.

#### RemindCalibrate

Specifies if the monitoring process is to display a reminder when the screen resolution changes. A value of 1 denotes that the reminder is on; a value of 0 denotes that the reminder is off.

#### Calibrated

Specifies if the user performed a first calibration after the installation of the tool. This field is used by the monitoring process. A value of 1 denotes that the calibration was performed and a value of 0 denotes that the calibration was not performed.

#### Notes:

1. Entries after the exclamation (!) character are remarks and will not be parsed by the utility.
2. The first **ConnectionType** entry will be the selected connection type.
3. The space characters before and after the = character are optional.

## Using the Touch Screen Calibration Utility

The touch screen calibration tool allows you to calibrate the 4820 SurePoint Solution touch display. After installing and opening the tool, the system prompts you to calibrate the display, see Figure J-1.



Figure J-1. Example of Initial Installation Window

created on October 2, 2001

If you select to calibrate, you can select from three tabs: Calibrate, Click Settings, and Hardware, as shown in Figure J-2.

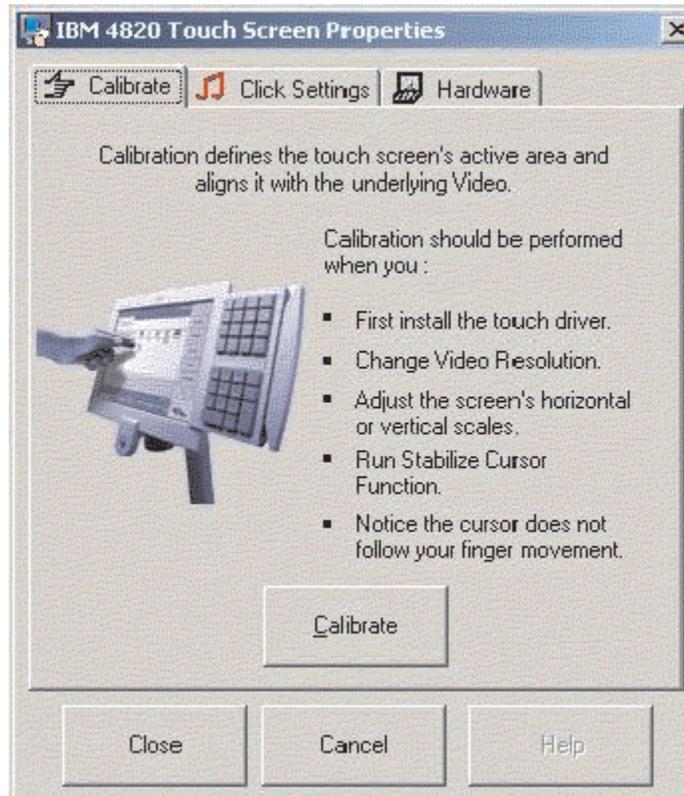


Figure J-2. Example of Calibrate Window

As shown in Figure J-3, the Click Settings tab allows you to set the touch click settings.



Figure J-3. Example of Click Settings Window

The Hardware tab allows you to restore the hardware default settings.



Figure J-4. Example of the Hardware Window

Whenever you change the screen resolution, a dialog box (Figure J-5) appears.



Figure J-5. Example of the reminder to calibrate



---

## Appendix K. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product program or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering the subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

---

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States, other countries or both:

C Set ++	IBM	SAA
FFST	Micro Channel	SureMark
FFST/2	Operating System/2	SurePoint
First Failure Support Technology	OS/2	VisualAge
First Failure Support Technology/2	Presentation Manager	

Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

| Other company, product, and service names may be trademarks or service marks  
| of others.

| Red Hat is a trademark of Red Hat Corporation.

---

## Glossary

This glossary defines terms and abbreviations used in this manual.

### A

**active.** Able to communicate on the network. A token-ring network adapter is active if it is able to transmit and receive on the network. Operational. Pertaining to a node, or device that is connected or is available for connection to another node or device. Currently transmitting or receiving.

**adapter.** In the point-of-sale terminal, a circuit card that, with its associated software, enables the terminal to use a function or feature. In a LAN, within a communicating device, a circuit card that, with its associated software and/or microcode, enables the device to communicate over the network.

**address.** In data communication, the IEEE-assigned unique code, or the unique locally administered code assigned to each device, or workstation connected to a network. A character, group of characters, or a value that identifies a register, a particular part of storage, a data source, or a data link. The value is represented by one or more characters. To refer to a device, or an item of data by its address. The location in the storage of a computer where data is stored.

**address space.** The complete range of addresses that is available to a programmer.

**all points addressable (APA).** In computer graphics, pertaining to the ability to address and display or not display each picture element (pel) on a display surface.

**alphanumeric.** Pertaining to a character set containing letters, digits, and other characters, such as punctuation marks.

**Alphanumeric point-of-sale keyboard (ANPOS keyboard).** This keyboard consists of a section of alphanumeric keys, a programmable set of point-of-sale keys, a numeric keypad, and system function keys.

**American National Standard Code for Information Interchange (ASCII).** The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphics characters.

**American National Standards Institute (ANSI).** An organization for the purpose of establishing voluntary industry standards.

**ANPOS keyboard.** Alphanumeric Point of Sale Keyboard.

**ANSI.** American National Standards Institute.

**APA.** all points addressable.

**API.** Application program interface.

**application program.** A program written for or by a user that applies to the user's own work. A program written for or by a user that applies to a particular application. A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

**application program interface (API).** The formally defined programming language interface that is between an IBM system control program or a licensed program and the user of the program.

**array.** An arrangement of elements in one or more dimensions.

**ASCII.** American National Standard Code for Information Interchange.

**asynchronous.** Pertaining to two or more processes that do not depend upon the occurrence of specific events such as timing signals. Without regular time relationship; unexpected or unpredictable with respect to the execution of program instructions.

**attach.** To connect a device physically. To make a device a part of a network logically. Compare with *connect*.

**attaching device.** Any device that is physically connected to a network and can communicate over the network.

### B

**backup.** Pertaining to a system, device, file, or facility that can be used in the event of a malfunction or the loss of data.

**backup copy.** A copy, usually of a program or of a library member, that is kept in case the original or the working copy is unintentionally altered or destroyed.

**bar code.** A code representing characters by sets of parallel bars of varying thickness and separation that are read optically by transverse scanning.

**baud.** The rate at which signal conditions are transmitted per second. Contrast with *bits per second (bps)*.

**BCD.** Binary-coded decimal notation.

**binary.** Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. Pertaining to a selection, choice, or condition that has two possible different values or states.

**binary-coded decimal notation (BCD).** A binary-coded notation in which each of the decimal digits is represented by a binary numeral. For example, in binary-coded decimal notation that uses the weights 8, 4, 2, 1, the number "twenty three" is represented by 0010 0011. In the pure binary numeration system, its representation is 10111.

**bit.** Either of the binary digits: a 0 or 1.

**bit map.** A coded representation in which each bit or group of bits represents or corresponds to an item; for example, a configuration of bits in main storage in which each bit indicates whether a peripheral device or a storage block is available or in which each group of bits corresponds to one pixel of a display image.

**bits per second (bps).** The rate at which bits are transmitted per second. Contrast with *baud*.

**block size.** The minimum size that frames are grouped into for retransmission. The number of data elements (such as bits, bytes, characters, or records) that are recorded or transmitted as a unit.

**bps.** Bits per second.

**Bps.** Bytes per second.

**buffer.** A portion of storage used to hold input or output data temporarily. A routine or storage used to compensate for a difference in data rate or time of occurrence of events, when transferring data from one device to another.

**byte.** A string that consists of a number of bits, treated as a unit, and representing a character. A binary character operated upon as a unit and usually shorter than a computer word. A string that consists of a particular number of bits, usually 8, that is treated as a unit, and that represents a character. A group of 8 adjacent binary digits that represent one extended binary-coded decimal interchange code (EBCDIC). See *n-bit byte*.

## C

**C.** A high-level programming language designed to optimize run time, size, and efficiency.

**call.** The action of bringing a function or subprogram into effect, usually by specifying the entry conditions and jumping to an entry point.

**card reader.** See *magnetic stripe reader, (MSR)*.

**cash drawer.** A drawer at a point-of-sale terminal that can be programmed to open automatically. See *till*.

**channel.** A functional unit, controlled by a host computer, that handles the transfer of data between processor storage and local peripheral equipment. A path along which signals can be sent. The portion of a storage medium that is accessible to a given reading or writing station.

**clear.** To delete data from a screen or from memory.

**code page.** A particular assignment of hexadecimal identifiers to graphic characters.

**code point.** A 1-byte code representing one of 256 potential characters.

**command.** A request for performance of an operation or execution of a program. A character string from a source external to a system that represents a request for system action.

**Common User Access (CUA).** The portion of the IBM Systems Application Architecture (SAA) that defines the basic elements of the end-user interface and how to use them. These elements include screen layout, menu presentation and selection techniques, keyboard layout and use, and display options.

**compile.** To translate all or part of a program expressed in a high-level language into a computer program expressed in an intermediate language, an assembly language, or a machine language. To prepare a machine language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one computer instruction for each symbolic statement, or both, as well as performing the function of an assembler. To translate a source program into an executable program (an object program). To translate a program written in a high-level programming language into a machine language program.

**compiler.** A program that decodes instructions written as pseudo codes and produces a machine language program to be executed at a later time. Contrast with *interpretive routine*. Synonymous with *compiling program*.

**compiling program.** Synonym for compiler.

**component.** Any part of a network other than an attaching device, such as an IBM 8228 Multistation Access Unit. Hardware or software that is part of a functional unit.

**configuration.** The group of devices, options, and programs that make up a data processing system or network as defined by the nature, number, and chief characteristics of its functional units. More specifically, the term may refer to a hardware configuration or a software configuration. See also *system configuration*.

**configuration file.** The collective set of definitions that describes a configuration.

created on October 2, 2001

**connect.** In a LAN, to physically join a cable from a station to an access unit or network connection point. Contrast with *attach*.

**constant.** String or numeric value that does not change throughout program execution.

**control character.** A character whose occurrence in a particular context initiates, modifies, or stops a control operation. A control character may be recorded for use in a subsequent action, and it may have a graphic representation in some circumstances.

**CRC.** Cyclic redundancy check.

**cursor.** A movable point of light (or a short line) that indicates where the next character is to be entered on the display screen.

**customer receipt.** An itemized list of merchandise purchased and paid for by the customer.

**customize.** To tailor a program or store system through option selection.

**cyclic redundancy check (CRC).** Synonym for *frame check sequence (FCS)*.

## D

**data.** A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human or automatic means. Any representations such as characters or analog quantities to which meaning is or might be assigned.

**data circuit-terminating equipment (DCE).** In a data station, the equipment that provides the signal conversion and coding between the data terminal equipment (DTE) and the line.

**data communication.** Transfer of information between functional units by means of data transmission according to a protocol. The transmission, reception, and validation of data.

**data file.** A collection of related data records organized in a specific manner; for example, a payroll file (one record for each employee, showing such information as rate of pay and deductions) or an inventory file (one record for each inventory item, showing such information as cost, selling price, and number in stock.) See also *data set, file*.

**data processing system.** A network, including computer systems and associated personnel, that accepts information, processes it according to a plan, and produces the desired results.

**data set.** Logically related records treated as a single unit. See also *file*.

**data terminal equipment (DTE).** That part of a data station that serves as a data source, data receiver, or both. Equipment that sends or receives data, or both.

**data transmission.** The conveying of data from one place for reception elsewhere by means of telecommunications.

**data type.** The mathematical properties and internal representation of data and functions.

**DBCS.** Double-byte character set.

**DCE.** Data circuit-terminating equipment.

**default.** Pertaining to an attribute, value, or option that is assumed when none is explicitly specified.

**default value.** The value the system supplies when the user does not specify a value.

**device.** A mechanical, electrical, or electronic contrivance with a specific purpose. An input/output unit such as a terminal, display, or printer. See also *attaching device*.

**device connection.** The connection between an application and a hardware device created by the IBM Point of Sale Subsystem when the application opens a device.

**device descriptor.** An identifier that represents a device to the IBM Point of Sale Subsystem application programming interface. This identifier is created by the IBM Point of Sale Subsystem when the application opens a device.

**device driver.** The code needed to attach and use a device on a computer or a network.

**device handler.** In the OS/2 operating system, the component of a device driver that communicates directly with the application.

**digital.** Pertaining to data in the form of digits. Contrast with *analog*. Pertaining to data consisting of numerical values or discrete units.

**direct file.** A file in which records are assigned specific record positions. No matter what order the records are put in a direct file, they always occupy the assigned position. A direct file is the same as a random file except that a direct file contains no delimiting characters, such as quotes enclosing string fields.

**directory.** A table of identifiers and references that correspond to items of data. An index that a control program uses to locate one or more blocks of data that are stored in separate areas of a data set in direct access storage.

**disabled.** Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Pertaining to the state in which a

transmission control unit or audio response unit cannot accept incoming calls on a line.

**disk.** A round, flat plate coated with a magnetic substance that is used to store computer data. See also *integrated disk*, *fixed disk*.

**Disk Operating System (DOS).** An operating system for computer systems that use disks and diskettes for auxiliary storage of programs and data.

**display.** A visual presentation of data. A device that presents visual information to the point-of-sale terminal operator and to the customer, or to the display station operator.

**distributed.** Physically separate but connected by cables.

**DLL.** See *dynamic link library*.

**DOS.** Disk Operating System.

**double-byte character set (DBCS).** A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with single-byte character set.

**driver.** Software component that controls a device.

**DTE.** Data terminal equipment.

**dump.** To write at a particular instant the contents of storage, or part of storage, onto another data medium for the purpose of safeguarding or debugging the data. Data that has been dumped.

**duplex.** In data communication, pertaining to a simultaneous two-way independent transmission in both directions. Synonymous with *full-duplex*. contrast with *half-duplex*.

**dynamic link library (DLL).** In the OS/2 and Windows operating systems, the delayed connection of a library to a routine until load time or run time.

## E

**EAN.** European article number.

**enabled.** On a LAN, pertaining to an adapter or device that is active, operational, and able to receive frames from the network. Pertaining to a state of a processing unit that allows the occurrence of certain types of interruptions. Pertaining to the state in which a transmission control unit or an audio response unit can accept incoming calls on a line.

**end-of-file.** An internal label, immediately following the last record of a file, signaling the end of that file.

**error message.** A message that is issued because an error has been detected.

**escape character.** Code extension character used, in some cases, with one or more succeeding characters to indicate by some convention that the coded representation following the character or the group of characters are to be interpreted according to a different code or different character set.

**European article number (EAN).** A number that is assigned to and encoded on an article of merchandise for scanning in some countries.

**event.** Processing unit containing price changes and item file updates. All records in an event share common characteristics such as type of change and event due date. An occurrence of significance to a task; for example, the completion of an asynchronous operation, such as an I/O operation.

**exception.** An abnormal condition such as an I/O error encountered in processing a data set or a file. See also *overflow exception* and *underflow exception*.

**exit.** To execute an instruction or statement within a portion of a program in order to terminate the execution of that portion. **Note:** Such portions of programs include loops, routines, subroutines, and modules.

**expansion board.** In an IBM Personal Computer, a panel containing microchips that a user can install in an expansion slot to add memory or special features. Synonymous with *expansion card*, *extender card*.

**expansion card.** Synonym for *expansion board*.

**extender card.** Synonym for *expansion board*.

## F

**fat-finger.** When two keys are pressed faster than the value specified using the PosNfatFingerTimeout resource. This could occur under any of the following conditions: 1) Two keys on the keyboard were pressed at the same time. 2) The operator is keying faster than 25 keys per second. 3) A double key is not defined to the keyboard device handler.

**field.** On a data medium or a storage medium, a specified area used for a particular category of data; for example, a group of character positions used to enter or display wage rates on a panel.

**FIFO.** First-in–first-out.

**file.** A named set of records stored or processed as a unit. For example, an invoice may form a record and the complete set of such records may form a file. See also *data file* and *data set*.

created on October 2, 2001

**file name.** A name assigned or declared for a file. The name used by a program to identify a file.

**first-in–first-out (FIFO).** A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

**fixed disk (drive).** In a personal computer system unit, a disk storage device that reads and writes on rigid magnetic disks. It is faster and has a larger storage capacity than a diskette and is permanently installed.

**flag.** A character or indicator that signals the occurrence of some condition, such as the setting of a switch, or the end of a word.

**frame.** The unit of transmission in some LANs, including the IBM Token-Ring Network and the IBM PC Network. It includes delimiters, control characters, information, and checking characters. On a token-ring network, a frame is created from a token when the token has data appended to it. On a token-bus network (IBM PC Network), all frames including the token frame contain a preamble, start delimiter, control address, optional data and checking characters, end delimiter, and are followed by a minimum silence period. A housing for machine elements. In synchronous data link control (SDLC), the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag.

**frequency.** The rate of signal oscillation, expressed in hertz (cycles per second).

**full-duplex.** Synonym for *duplex*.

**function.** A specific purpose of an entity, or its characteristic action. A subroutine that returns the value of a single variable. In data communications, a machine action such as a carriage return or line feed.

## G

**GCGID.** See *Graphic Character Global Identifier*.

**global.** Pertaining to that which is defined in one subdivision of a computer program and used in at least one other subdivision of that computer program.

**Graphic Character Set Global Identifier (GCGID).** A 4- to 8-character identifier assigned to a registered graphic character in an IBM registry.

**group.** A set of related records that have the same value for a particular field in all records. A collection of users who can share access authorities for protected resources. A list of names that are known together by a single name.

## H

**half-duplex.** In data communication, pertaining to transmission in only one direction at a time. Contrast with *duplex*.

**hardware.** Physical equipment as opposed to programs, procedures, rules, and associated documentation.

**hertz (Hz).** A unit of frequency equal to one cycle per second. **Note:** In the United States, line frequency is 60Hz or a change in voltage polarity 120 times per second; in Europe, line frequency is 50Hz or a change in voltage polarity 100 times per second.

**hexadecimal notation.** Notation for the base-16 number system using the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F to represent values from 0 to 15 (decimal).

**hot key.** The key combination used to change from one session to another on a workstation.

**hot plug.** To connect a USB I/O device to the universal serial bus without powering the host system down.

**hot unplug.** To disconnect a USB I/O device from the universal serial bus without powering the host system down.

**Hz.** See *hertz*.

## I

**IBM Disk Operating System (DOS).** A disk operating system based on MS-DOS.<sup>3</sup>

**identifier.** String of characters used to name elements of a program, such as variable names, reserved words, and user-defined function names.

**inactive.** Not operational. Pertaining to a node or device not connected or not available for connection to another node or device. In the IBM Token-Ring Network, pertaining to a station that is only repeating frames or tokens, or both.

**information (I) frame.** A frame in I format used for numbered information transfer. See also *supervisory frame*, *unnumbered frame*.

**initialize.** In a LAN, to prepare the adapter (and adapter support code, if used) for use by an application program.

**initial program load (IPL).** The initialization procedure that causes an operating system to begin operation.

---

3. MS-DOS is a trademark of the Microsoft Corporation.

**input device.** Synonym for *input unit*.

**input/output device.** See *I/O device*.

**input/output (I/O).** Pertaining to a device whose parts can perform an input process and an output process at the same time. Pertaining to a functional unit or channel involved in an input process, output process, or both, concurrently or not, and to the data involved in such a process.

**input unit.** A device in a data processing system that is used to enter data into the system. Synonymous with *input device*.

**instance.** An occurrence of a particular device or object. For example, two instances of the `PosDisplay` class name can be `Shopper1` and `Shopper2`, where these instances refer to the same physical display.

**integrated.** Arranged together as one unit.

**integrated disk.** An integral part of the processor that is used for magnetically storing files, application programs, and diagnostics. Synonymous with *disk*.

**interaction.** A basic unit used to record system activity, consisting of the acceptance of a line of terminal input, processing of the line, and a response, if any.

**interface.** A shared boundary between two functional units, defined by functional characteristics, common physical interconnection characteristics, signal characteristics, and other characteristics as appropriate. A shared boundary. An interface may be a hardware component to link two devices or a portion of storage or registers accessed by two or more computer programs. Hardware, software, or both, that links systems, programs, or devices.

**interleave.** To insert segments of one program into another program so that the two programs can, in effect, be executed at the same time.

**International Organization for Standardization (ISO).** An organization of national standards bodies from various countries established to promote development of standards to facilitate international exchange of goods and services, and develop cooperation in intellectual, scientific, technological, and economic activity.

**interpretive routine.** A routine that decodes instructions written as pseudocodes and immediately executes the instructions. Contrast with *compile*.

**I/O.** Input/output.

**I/O device.** Equipment for entering and receiving data from the system.

**IPL.** Initial program load.

**ISO.** International Organization for Standardization.

**item.** One member of a group. In a store, one unit of a commodity, such as one box, one bag, or one can. Usually an item is the smallest unit of a commodity to be sold.

## J

**JIS.** Japanese Industrial Standard

**JUCC.** Japanese Unified Cash Card

## K

**keyboard.** A group of numeric keys, alphabetic keys, special character keys, or function keys used for entering information into the terminal and into the system.

**kHz.** Kilohertz. See also *hertz*.

**kilohertz (kHz).** A thousand hertz. See also *hertz*.

## L

**LED.** Light-emitting diode.

**lift-off.** When a pointing device is removed from a touch-sensitive surface.

**light-emitting diode (LED).** A semiconductor chip that gives off visible or infrared light when activated.

**line.** On a terminal, one or more characters entered before a return to the first printing or display position.

**link .** The combination of physical media, protocols, and programming that connects devices on a network. In computer programming, the part of a program, in some cases a single instruction or an address, that passes control and parameters between separate portions of the computer program. To interconnect items of data or portions of one or more computer programs.

**listing.** A printout, usually prepared by a language translator, that lists the source code.

**load.** In computer programming, to enter data into memory or working registers.

**lock.** To disable a device, such as a scanner or MSR, so that it cannot receive input. See also *unlock*.

**logging.** The chronological recording of events occurring in a system or a subsystem for accounting or data collection purposes.

**logical connection.** In a network, devices that can communicate or work with one another because they share the same protocol. See also *physical connection*.

created on October 2, 2001

**logon (n).** The procedure for starting up a point-of-sale terminal or store controller for normal sales operations by sequentially entering the correct security number and transaction number. Synonymous with *sign-on*.

**log on (v).** To initiate a session. In SNA products, to initiate a session between an application program and a logical unit (LU). Synonymous with *sign-on*.

**loop.** A set of instructions that may be executed repeatedly while a certain condition prevails. See also *store loop*. A closed unidirectional signal path connecting input/output devices to a network.

## M

**macro.** An instruction that causes the execution of a predefined sequence of instructions in the same source language.

**magnetic stripe.** The magnetic material (similar to recording tape) on merchandise tickets, credit cards, and employee badges. Information is recorded on the stripe for later "reading" by the magnetic stripe reader (MSR) or magnetic wand reader attached to the point-of-sale terminal.

**magnetic stripe reader (MSR).** A device that reads coded information from a magnetic stripe on a card, such as a credit card, as it passes through a slot in the reader.

**make scan code.** The hardware scan code received by the keyboard device driver when a key on the keyboard is physically pressed.

**Mb.** Megabit.

**MB.** Megabyte.

**megabit (Mb).** A unit of measure for throughput. 1 megabit = 1,048,576 bits.

**megabyte (MB).** A unit of measure for data. 1 megabyte = 1,048,576 bytes.

**memory.** Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing.

**message.** An arbitrary amount of information whose beginning and end are defined or implied. A group of characters and control bit sequences transferred as an entity. In telecommunication, a combination of characters and symbols transmitted from one point to another. A logical partition of the user device's data stream to and from the adapter. See also *error message*, *operator message*.

**microcode.** One or more microinstructions. A code, representing the instructions of an instruction set, that is implemented in a part of storage that is not

program-addressable. To design, write, and also test one or more microinstructions.

**microprocessor.** An integrated circuit that accepts coded instructions for execution. The instructions may be entered, integrated, or stored internally.

**migration.** Installation of a new version of a release of a program to replace an earlier version or release.

**modem (MODulator/DEModulator).** A device that converts digital data from a computer to an analog signal that can be transmitted in a telecommunication line, and converts the analog signal received to data for the computer.

**modulo check.** A function designed to detect most common input errors by performing a calculation on values entered into a system by an operator or scanning device.

**monitor.** A functional unit that observes and records selected activities for analysis within a data processing system. Possible uses are to show significant departures from the norm, or to determine levels of utilization of particular functional units. Software or hardware that observes, supervises, controls, or verifies operations of a system.

**MSR.** Magnetic stripe reader.

## N

**name.** An alphanumeric term that identifies a data set, statement, program, or cataloged procedure.

**n-bit byte.** A string that consists of n bits.

**network.** A configuration of data processing devices and software connected for information interchange. An arrangement of nodes and connecting branches. Connections are made between data stations.

**noise.** A disturbance that affects a signal and that can distort the information carried by the signal. Random variations of one or more characteristics of any entity, such as voltage, current, or data. Loosely, any disturbance tending to interfere with normal operation of a device or system.

**nonvolatile random access memory (NVRAM).** Random access memory that retains its contents after electrical power is shut off.

**NVRAM.** nonvolatile random access memory.

## O

**offline.** Operation of a functional unit without the control of a computer or control unit.

**online.** Operation of a functional unit that is under the continual control of a computer or control unit. The term also describes a user's access to a computer using a terminal.

**open.** To make an adapter ready for use. A break in an electrical circuit. To make a file ready for use.

**operating system.** Software that controls the execution of programs. An operating system may provide services such as resource allocation, scheduling, input/output control, and data management. Examples are IBM DOS and IBM OS/2.

**Operating System/2 (OS/2).** A set of programs that control the operation of high-speed large-memory IBM Personal Computers (such as the IBM Personal System/2 computer, Models 50 and above), providing multitasking and the ability to address up to 16 MB of memory. Contrast with *Disk Operating System (DOS)*.

**operation.** A defined action, namely, the act of obtaining a result from one or more operands in accordance with a rule that completely specifies the result for any permissible combination of operands. A program step undertaken or executed by a computer. An action performed on one or more data items, such as adding, multiplying, comparing, or moving.

**operator.** A symbol that represents the action being performed in a mathematical operation. A person who operates a machine.

**operator message.** A message from the operating system or a program telling the operator to perform a specific function or informing the operator of a specific condition within the system, such as an error condition.

**option.** A specification in a statement, a selection from a menu, or a setting of a switch, that may be used to influence the execution of a program. A hardware or software function that may be selected or enabled as part of a configuration process. A piece of hardware (such as a network adapter) that can be installed in a device to modify or enhance device function.

**OS.** Operating system.

**OS/2.** Operating System/2.

**output device.** A device in a data processing system by which data can be received from the system. Synonymous with *output unit*.

**output unit.** Synonym for *output device*.

**overflow exception.** A condition caused by the result of an arithmetic operation having a magnitude that exceeds the largest possible number. See also *underflow exception*.

## P

**parameter.** A name in a procedure that is used to refer to an argument passed to that procedure. A variable that is given a constant value for a specified application and that may denote the application. An item in a menu or for which the user specifies a value or for which the system provides a value when the menu is interpreted. Data passed between programs or procedures.

**parity bit.** A binary digit appended to a group of binary digits to make the sum of all the digits (including the appended binary digit) either always odd (odd parity) or always even (even parity).

**parity (even).** A condition when the sum of all of the digits in an array of binary digits is even.

**parity (odd).** A condition when the sum of all of the digits in an array of binary digits is odd.

**personal computer (PC).** A desk-top, free-standing, or portable microcomputer that usually consists of a system unit, a display, a keyboard, one or more diskette drives, internal fixed-disk storage, and an optional printer. PCs are designed primarily to give independent computing power to a single user and are inexpensively priced for purchase by individuals or small businesses. Examples include the various models of the IBM Personal Computers, and the IBM Personal System/2 computer.

**physical connection.** The ability of two connectors to mate and make electrical contact. In a network, devices that are physically connected can communicate only if they share the same protocol. See also *logical connection*.

**PLD.** Power line disturbance.

**PLU.** Price Look Up.

**plug.** A connector for attaching wires from a device to a cable, such as a store loop. A plug is inserted into a receptacle or plug. To insert a connector into a receptacle or socket.

**pointer.** An identifier that indicates the location of an item of data in memory. A data element that indicates the location of another data element. A physical or symbolic identifier of a unique target.

**point-of-sale terminal.** A unit that provides point-of-sale transaction, data collection, credit authorization, price look-up, and other inquiry and data entry functions.

**polling.** Interrogation of devices for purposes such as to avoid contention, to determine operational status, or to determine readiness to send or receive data. In data communication, the process of inviting data stations to

created on October 2, 2001

transmit, one at a time. The polling process usually involves the sequential interrogation of several data stations.

**polling characters (address).** A set of characters specific to a terminal and the polling operation; response to these characters indicates to the computer whether the terminal has a message to enter.

**port.** An access point for data entry or exit. A connector on a device to which cables for other devices such as display stations and printers are attached. Synonymous with *socket*.

**post.** To affix to a usual place. To provide items such as return code at the end of a command or function. To define an appendage routine. To note the occurrence of an event.

**POST.** Power-On Self Test.

**power line disturbance (PLD).** Interruption or reduction of electrical power.

**Power-On Self Test (POST).** A series of diagnostic tests that are run automatically each time the computer's power is switched on.

**presentation facility.** The visual component of the operating system that presents, in windows, a graphical interface. In OS/2, the presentation facility is Presentation Manager. In AIX, it is Xwindows.

**problem determination.** The process of determining the source of a problem as being a program component, a machine failure, a change in the environment, a common-carrier link, a user-supplied device, or a user error.

**procedure.** A set of related control statements that cause one or more programs to be performed. In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a procedure call. A set of instructions that gives a service representative a step-by-step procedure for tracing a symptom to the cause of failure.

**process.** An instance of an executing application and the resources it is using.

**processor.** In a computer, a functional unit that interprets and executes instructions.

**prompt.** A character or word displayed by the operating system to indicate that it is ready to accept input.

## Q

**queue.** A line or list formed by items in a system waiting for service; for example, tasks to be performed or messages to be transmitted in a message routing system.

## R

**RAM.** Random access memory.

**random access.** An access mode in which specific logical records are obtained from or placed into a mass storage file in a nonsequential manner.

**random access memory (RAM).** A computer's or adapter's volatile storage area into which data may be entered and retrieved in a nonsequential manner.

**read.** To acquire or to interpret data from a storage device, from a data medium, or from another source.

**read-only memory (ROM).** A computer's or adapter's storage area whose contents cannot be modified by the user except under special circumstances.

**receive.** To obtain and store information transmitted from a device.

**record.** A collection of related items of data, treated as a unit; for example, in stock control, each invoice could constitute one record. A complete set of such records may form a file.

**register.** A storage area in a computer's memory where specific data is stored. Registers are used in the actual manipulation of data values during the execution of a program. A storage device having a specified storage capacity such as bit, byte, or computer word, and usually intended for a special purpose. In the IBM Store System, a term that refers to the point-of-sale terminal.

**remove.** To take an attaching device off a network. To stop an adapter from participating in data passing on a network.

**resource.** An element that affects the way devices behave.

**resource set.** The set of resources associated with a device.

**response.** The information the network control program sends to the access method, usually in answer to a request received from the access method. (Some responses, however, result from conditions occurring within the network control program, such as accumulation of error statistics.)

**retry.** In data communication, sending the current block of data a prescribed number of times or until it is entered correctly and accepted.

**return code.** A value (usually hexadecimal) provided by an adapter or a program to indicate the result of an action, command, or operation. A code used to influence the execution of succeeding instructions. A value established by the programmer to be used to

influence subsequent program action. This value can be printed as output or loaded in a register.

**ROM.** Read-only memory.

**routine.** Part of a program, or a sequence of instructions called by a program, that may have some general or frequent use.

## S

**satellite.** A computer that is under the control of another computer and performs subsidiary operations. An offline auxiliary computer.

**SBCS.** Single-byte character set.

**scan.** To pass an item over or through the scanner so that the encoded information is read. See also *wanding*.

**scanner.** A device that examines the bar code on merchandise tickets, credit cards, and employee badges and generates analog or digital signals corresponding to the bar code.

**scroll.** To move all or part of the display image vertically or horizontally to display data that cannot be observed within a single display image. See also *page (2)*.

**segment.** See *cable segment, LAN segment, ring segment*.

**sequential file.** A disk file in which records are read from or placed into the file according to the order they are processed.

**serial input/output (SIO).** Pertaining to the sequential or consecutive occurrence of two or more input, output or both, activities in a single device or channel.

**session.** A connection between two application programs that allows them to communicate. In SNA, a logical connection between two network addressable units that can be activated, tailored to provide various protocols, and deactivated as requested. The data transport connection resulting from a call or link between two devices. The period of time during which a user of a node can communicate with an interactive system, usually the elapsed time between log on and log off. In network architecture, an association of facilities necessary for establishing, maintaining, and releasing connections for communication between stations.

**signal.** A time-dependent value attached to a physical phenomenon for conveying data. A variation of a physical quantity, used to convey data.

**sign-on.** A procedure to be followed at a terminal or workstation to establish a link to a computer. To begin a session at a workstation.

**single-byte character set (SBCS).** A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set*.

**SIO.** See *serial input/output*.

**socket.** Synonym for *port (2)*.

**state transition.** The act of moving from one conversation state to another.

**station.** A point-of-sale terminal that consists of a processing unit, a keyboard, and a display. It can also have input/output devices, such as a printer, a magnetic stripe reader or cash drawers. A communication device attached to a network. The term used most often in LANs is an *attaching device* or *workstation*. An input or output point of a system that uses telecommunication facilities; for example, one or more systems, computers, terminals, devices, and associated programs at a particular location that can send or receive data over a telecommunication line. See also *attaching device, workstation*.

**subdirectory.** Any level of file directory lower than the root directory within a hierarchical file system.

**subroutine.** Section of code that performs a specific task and is logically separate from the rest of the program.

**subsystem.** A secondary or subordinate system, or programming support, usually capable of operating independently of or asynchronously with a controlling system.

**summary journal.** A record of the terminal operational activity that is printed at the terminal.

**switch.** On an adapter, a mechanism used to select a value for, enable, or disable a configurable option or feature.

**system.** In data processing, a collection of people, machines, and methods organized to accomplish a set of specific functions. See also *data processing system* and *operating system*.

**system configuration.** A process that specifies the devices and programs that form a particular data processing system.

**system unit.** A part of a computer that contains the processing unit, and may contain devices such as disk and diskette drives. In an IBM Personal Computer, the unit that contains the processor circuitry, read-only memory (ROM), random access memory (RAM), and the I/O channel. It may have one or more disk or diskette drives. In an IBM 4683/4684 terminal, the part of the terminal that contains the processing unit, ROM, RAM, disk and diskette drives, and the I/O channel.

## T

**terminal.** In data communication, a device, usually equipped with a keyboard and a display, capable of sending and receiving information over a communication channel.

**thread.** A unit of execution within a process. It uses the resources of the process.

**throughput.** A measure of the amount of work performed by a computer system over a given period of time, for example, number of jobs per day. A measure of the amount of information transmitted over a network in a given period of time. For example, a network's data transfer rate is usually measured in bits per second.

**till.** A tray in the cash drawer of the point-of-sale terminal, used to keep the different denominations of bills and coins separated and easily accessible.

**touch-down.** When contact is made with a touch-sensitive surface.

**trace.** A record of the execution of a computer program. It exhibits the sequences in which the instructions were executed. A record of the frames and bytes transmitted on a network.

**transaction.** The process of recording item sales, processing refunds, recording coupons, handling voids, verifying checks before tendering, and arriving at the amount to be paid by or to a customer. The receiving of payment for merchandise or service is also included in a transaction. In an SNA network, an exchange between two programs that usually involves a specific set of initial input data that causes the execution of a specific task or job. Examples of transactions include the entry of a customer's deposit that results in the updating of the customer's balance, and the transfer of a message to one or more destination points.

**transition.** See *state transition*.

**transmission.** The sending of data from one place for reception elsewhere.

**transmit.** To send information from one place for reception elsewhere.

**typematic.** A keyboard button that will continue to enter characters or repeat its function as long as the button is held down.

## U

**underflow exception.** A condition caused by the result of an arithmetic operation having a magnitude less than the smallest possible nonzero number. See also *overflow exception*.

**unlock.** To enable a device, such as a scanner or MSR, so that it can read data. See also *lock*.

**universal product code (UPC).** An encoded number that can be assigned to and printed on or attached to an article of merchandise for scanning.

**universal serial bus (USB).** A serial interface standard for telephony and multimedia connections to personal computers.

**UDC.** User defined character.

**UPC.** Universal product code.

**user.** Category of identification defined for file access protection. A person using a program or system.

**user defined character (UDC).** User defined character.

**user interface.** Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device.

## V

**variable.** A named entity that is used to refer to data and to which values can be assigned. Its attributes remain constant, but it can refer to different values at different times. In computer programming, a character or group of characters that refers to a value and, in the execution of a computer program, corresponds to an address. A quantity that can assume any of a given set of values.

**version.** A separate IBM-licensed program, based on an existing IBM-licensed program, that usually has significant new code or new function.

## W

**wandering.** Passing the tip of the wand reader over information encoded on a merchandise ticket, credit card, or employee badge.

**workstation.** An I/O device that allows either transmission of data or the reception of data (or both) from a host system, as needed to perform a job: for example, a display station or printer. A configuration of I/O equipment at which an operator works. A terminal or microcomputer, usually one connected to a mainframe or network, at which a user can perform tasks.



# Index

## Numerics

- 1520 Hand-Held Scanner Model A02
  - default resource values 16-10
  - description 16-2
- 20 CPI escape character sequence 12-23
- 2x20 Character VFD Customer Display,
  - characteristics 8-3
- 3F Fiscal Printer 12-4
- 40-Character Liquid Crystal Display, characteristics 8-3
- 40-Character Vacuum Fluorescent Display II,
  - characteristics 8-3
- 4500 Hand-Held Bar Code Reader
  - description 16-1
- 4500 HHBCR default resource values 16-10
- 4501 Hand-Held Bar Code Reader
  - description 16-1
- 4501 HHBCR default resource values 16-10
- 4585 HHBCR default resource values 16-10
- 4610 microcode, updating G-1
- 4610 Model TI1, TI2, TI3, TI4, printer
  - characteristics 12-8
- 4685 Hand-Held Bar Code Reader
  - description 16-1
- 4685 Point of Sale Keyboard Model K01
  - characteristics 9-10
- 4686 Retail Point of Sale Scanner
  - default resource values 16-10
  - description 16-2
- 4687 Point of Sale Scanner
  - 4687-2 16-2
  - default resource values 16-11
  - default resource values, Model 2 15-5
  - description 16-2
  - Scanner Scale Model 2 15-1
- 4689-001 printer characteristics
  - 4689-002 printer characteristics 12-6
- 4689-301 printer characteristics 12-7
- 4693, controlling the power 13-2
- 4695 Point of Sale Touch Terminal
  - description 17-1
- 4696 Point of Sale Scanner Scale
  - default resource values 15-5, 16-11
  - description 15-1, 16-2
- 4697 Point of Sale Scanner
  - default resource values 16-11
  - description 16-3
- 4698 Point of Sale Scanner
  - default resource values 16-12
- 4698 Point of Sale Scanner Model 2
  - default resource values 16-12
  - description 15-2, 16-3
- 4820 SurePoint Solution Touch Screen Calibration J-1
- 50-Key Modifiable Layout Keyboard and Operator Display characteristics 9-3
- 50-Key Modifiable Layout Keyboard characteristics 9-3

- 7497 Attachment Adapter
  - NVRAM programming read and write operations 11-2

## A

- abnormal end of PosWrite(), message 20-17
- acquiring devices
  - exception 19-1
  - from non-presentation facility application 5-8
  - from presentation facility application 5-7
  - general programming information 5-6
  - PosIOCtl() request 19-41
- addressable print positions maximum, Model 4A printer 12-5
- addressable print positions per character, maximum, for Model 4A printer 12-5
- addressable print positions per line, maximum, for Model 4A printer 12-5
- advance paper to tear bar escape character sequence 12-15, 12-23
- alarm
  - characteristics 6-1
  - device handler 6-1
  - programming 6-1
  - related information 6-2
  - silencing 6-1
  - sounding 6-1, 19-4
  - status, getting 6-2
  - subroutines used with 6-2
  - turning off 19-3
- alarm for programmable power, clearing 13-2
- alphanumeric display, characteristics 8-1
- alphanumeric point-of-sale keyboards
  - ANPOS monitor 3-4
  - configuring 3-4
  - definition of 9-2
- ANPOS Keyboard characteristics 9-5
- ANPOS monitor 3-4
- API (application programming interface)
  - call sequence 18-1
  - overview 18-1
  - PosClose subroutine 18-3
  - PosInitialize subroutine 18-5
  - PosIOCtl subroutine 18-8
  - PosOpen subroutine 18-10
  - PosRead subroutine 18-16
  - PosWrite subroutine 18-19
  - restriction 18-1
  - shutdown sequence 18-1
  - subroutines 18-1
- application
  - address space, NVRAM 11-2
  - configuring 3-1
  - improving maintainability 5-19
  - initializing 5-1
  - Microsoft Windows considerations 5-17
  - multi-threaded design 5-18

- application (*continued*)
  - optimizing performance 5-16
  - polling considerations 5-17
  - Presentation Manager considerations 5-16
  - priority 5-16
  - programming, point-of-sale device 5-1
  - terminating 5-11
- argument list values
  - assigning the value directly 5-13
  - setting 5-12
  - statically initializing the argument list 5-12
  - using the macro PosSetArg 5-13
- automatic or manual document insertion 12-37, 12-38, 12-41
- availability of devices, determining 5-3
  - device 5-4
- avoirdupois weight units 15-1

## B

- barcode printing 12-10
- bitmap data format for Character and Graphics Display 8-6
- bitmaps, writing to the Character and Graphics Display 8-5
- block size, maximum for NVRAM read and write 11-1
- break signal
  - receiving from RS-232C 19-37
  - RS-232C message 20-15
  - sending 19-37
- brightness, controlling LCD 17-4
- buffer format
  - MSR data 10-4
  - scale data 15-2
  - scanner data 16-5
- buffering print data 12-17

## C

- C language header files 5-14
- calibration, 4820 SurePoint Solution Touch J-1
- calibration tool, touch screen J-1
- calibration tool, using the touch screen J-2
- call sequence, API 18-1
- carriage return control character 12-15, 12-22
- cash drawer
  - characteristics 7-1
  - device handler 7-1
  - getting status 7-2
  - opening till 7-1, 19-48
  - programming 7-1
  - related information 7-2
  - setting pulse width 7-2
  - till closed message 20-25
  - till open message 20-26
- changing print characteristics 12-15
- Character and Graphics Display characteristics 8-2
- character line lengths, DI station 12-39
- characters, writing to the display 8-4
- characters per line, maximum
  - 4610 Model TI1, TI2, TI3, TI4 printers 12-9

- characters per line, maximum (*continued*)
  - 4689-001 printer 12-6
  - 4689-002 printer 12-6
  - 4689-3x1 and TD5 printers 12-7
  - Model 2 12-2
  - Model 3, Model 4, and Model 4A printers 12-3
- chase escape character sequence 12-15, 12-23, 20-7
- check printing 12-43
- checkout keyboards 9-1
- clearing
  - alarm setting for programmable power 13-2
  - display 19-5, 19-6
  - display screen 8-6, 19-38
  - scale display 15-4
- click, controlling the keyboard 9-14
- closing
  - device connections 5-10
  - devices 5-10, 18-3
- code page support
  - displays 8-4
  - printer 12-11
- codes
  - error D-3
  - trace B-1
- completion of printing, determining 12-17
- configuring
  - alphanumeric point-of-sale keyboards 3-4
  - applications 3-1
  - IBM Point of Sale Subsystem 3-1
  - scale 15-4
  - scanner 16-8
- connection, device 5-4, 18-10
- console messages, viewing 4-1
- contrast, controlling LCD 17-4
- control characters 12-21
  - carriage return 12-15
  - escape control character 12-15
  - line feed 12-15
- control functions, performing 18-8
- control lines 14-2
- controlling
  - devices 5-5
  - devices, exception 19-1
  - keyboard click 9-14
  - keyboard typematic function 9-15
  - Num Lock key 9-14
  - point-of-sale-unique keys 9-14
  - printer 12-48
  - RS-232C port 14-2
  - Scroll Lock key 9-14
  - system hot keys 9-15
  - touch screen 17-5
- CR (customer receipt) station
  - advance paper to tear bar 12-22
  - cut paper 12-22
  - cut paper and stamp 12-22
  - fonts 12-2, 12-3, 12-6, 12-7, 12-9
  - function 12-1
  - interleaved printing, normal mode 12-15
  - line spacing 12-2, 12-3, 12-6, 12-8, 12-10
  - logo printing 12-18

created on October 2, 2001

CR (customer receipt) station (*continued*)  
maximum addressable print positions per character 12-5  
maximum addressable print positions per line 12-5  
maximum characters per line 12-2, 12-3, 12-6, 12-7, 12-9  
print barcodes 12-22  
printer errors, download logo mode 12-20  
printer errors, logo mode 12-19  
printer errors, normal mode 12-15  
queues 12-35  
speed for printing 12-2, 12-4, 12-6, 12-8, 12-11  
CTS 14-2  
cursor, NVRAM 11-2  
customized dumps, viewing 4-1  
customizing the IBM Point of Sale Subsystem 3-1  
cut paper and stamp escape character sequence 12-15, 12-23  
cut paper escape character sequence 12-23  
cutter, paper 12-42

## D

data, reading keyboard 9-12  
data, reading touch screen 17-3  
data available message  
MSR 20-6  
printer 20-8  
RS-232C 20-16  
scanner 20-6  
data buffer format  
MSR 10-4  
scale 15-2  
scanner 16-5  
data types A-1  
default modes  
scale 15-4  
scanner 16-9  
default state, keyboard 9-5, 9-7  
defining keys 9-11  
designing multi-threaded applications 5-18  
determining  
availability of devices 5-3  
printer status 12-35  
printing completion 12-17  
device  
acquiring 5-6, 19-41  
acquiring, exception 19-1  
acquiring from non-presentation facility 5-8  
acquiring from presentation facility 5-7  
availability, determining 5-3  
closing 5-10, 18-3  
connection, establish 5-4, 18-10  
controlling 5-5  
defining user-defined characters 5-8  
locking 19-43  
offline message 20-20  
online message 20-22  
opening 5-4, 18-10  
programming 5-1  
reading from 5-9, 18-16

device (*continued*)  
released, event message 20-24  
releasing 5-6, 19-44  
releasing from non-presentation facility 5-8  
releasing from presentation facility 5-7  
setting resource values 19-45  
unlock 19-47  
writing to 5-10, 18-19  
device handler  
alarm 6-1  
cash drawer 7-1  
display 8-1  
keyboard 9-1  
MSR 10-1  
NVRAM 11-1  
printer 12-1  
programmable power 13-1  
RS-232C 14-1  
scale 15-1  
scanner 16-1  
touch 17-1  
DI (document insert) station  
character line lengths 12-39  
controlling 12-36  
defining characters 19-24  
DI station line lengths 12-42  
disable printing 12-33, 19-25  
enable printing 12-33, 19-28  
enabling and disabling 12-15  
flip checks 12-22  
flipping a check 12-42  
fonts 12-2, 12-3, 12-6, 12-9  
front or side loaded documents 12-37, 12-41  
function 12-1  
insertion of narrow documents 12-40  
insertion of wide documents 12-40  
line spacing 12-2, 12-3, 12-6, 12-10  
logo printing 12-18  
manual or automatic insertion of documents 12-37, 12-38, 12-41  
maximum addressable print positions per character 12-5  
maximum addressable print positions per line 12-5  
maximum characters per line 12-2, 12-3, 12-6, 12-9  
portrait or landscape orientation 12-42  
printer errors, download logo mode 12-20  
printer errors, logo mode 12-19  
printer errors, normal mode 12-15  
reading MICR information 12-22  
registering documents 12-22  
reinserting documents 12-37, 12-39, 12-41  
releasing documents 12-22, 12-37, 12-39, 12-41, 12-42  
top or front loaded documents 12-38, 12-40  
DI station line lengths 12-42  
direct mode  
opening NVRAM in 11-3  
reading NVRAM data in 11-3  
writing NVRAM data in 11-3  
disable  
defining characters 19-24

- disable (*continued*)
    - DI station printing 12-33, 19-25
    - fiscal printing 12-33, 19-26
    - hot keys 19-7
    - MSR data 10-5
    - Num Lock key 19-8
    - Scroll Lock key 19-9
    - silence tone 19-35
  - discarding data
    - hold printing 19-31
    - printer 12-33, 12-34
    - queued 19-27
    - reset printer 19-30
    - resume printing 19-32
    - scanner 16-8, 19-40
    - user-defined characters 8-6
  - disk space requirements 1-8
  - disk space requirements for Microsoft Windows 1-8
  - disk space requirements for OS/2 1-8
  - display
    - 2x20 Character VFD Customer Display 8-3
    - 40-Character Liquid Crystal Display 8-3
    - 40-Character Vacuum Fluorescent Display II 8-3
    - alphanumeric 8-1
    - bitmap data format for Character and Graphics 8-6
    - Character and Graphics, characteristics 8-2
    - characteristics 8-1
    - clearing 8-6, 19-5, 19-38
    - code page support 8-4
    - defining characters 19-6
    - device handler 8-1
    - functions 8-3
    - Operator Display 8-2
    - programming 8-1
    - related information 8-7
    - scale, clearing 15-4
    - setting guidance lights 8-6, 21-20
    - Shopper Display 8-2
    - Two-Sided Vacuum Fluorescent Display II 8-3
    - writing bitmaps to Character and Graphics 8-5
    - writing characters to 8-4
  - documents
    - flipping a check 12-42
    - line lengths 12-42
    - loading in printer 12-36
    - manual or automatic insertion 12-37, 12-38, 12-41
    - print orientation 12-42
    - reinserting for printing 12-37, 12-39, 12-41
    - releasing 12-37, 12-39, 12-41, 12-42
    - reversing motor 12-39
    - side or front loaded 12-37, 12-41
    - top or front loaded 12-38, 12-40
  - double-click sensitivity adjustment for touch screen 17-2
  - double density escape character sequence 12-24
  - double-density escape character sequence 12-15
  - double-high font escape character sequence 12-15, 12-24
  - double key 21-23
  - double-strike printing 12-2, 12-4, 12-6, 12-8, 12-11
  - double-wide font escape character sequence 12-15, 12-24
  - DSR 14-2
  - DTR 14-2
  - Dual-Track MSR 10-2
  - dumps, viewing customized 4-1
- ## E
- emphasized printing 12-2, 12-4, 12-6, 12-8
  - enable
    - DI station printing 12-33, 19-28
    - fiscal printing 12-34, 19-29
    - hot keys 19-10
    - Num Lock key 19-11
    - Scroll Lock key 19-12
  - error codes D-3
  - error definitions, RS-232C 14-3
  - error messages
    - IBM Point of Sale Subsystem C-1
    - MSR data available 10-4
    - viewing 4-1
  - escape character sequences 12-22
    - 20 CPI 12-23
    - advance paper to tear bar 12-15, 12-23
    - chase 12-15, 12-23
    - cut paper 12-15, 12-23
    - cut paper and stamp 12-15, 12-23
    - double density 12-24
    - double-high font 12-15, 12-24
    - double-density 12-15
    - double-wide font 12-15, 12-24
    - fix font width 12-24
    - Flip Check 12-24
    - inline logo 12-25
    - move to tab 12-26
    - normal density 12-15, 12-26
    - normal font 12-15, 12-26
    - Print Barcode 12-27
    - Print pre-defined logo 12-28
    - Print pre-defined message 12-28
    - Read MICR 12-15, 12-28
    - register document 12-15, 12-29
    - release document 12-15, 12-29
    - right column align 12-29
    - scalable fonts 12-29
    - select color printing 12-30
    - Small Font 12-31
    - spread font 12-15, 12-31
    - Text Attributes 12-32
    - user-defined fonts 12-31
  - escape control character 12-15, 12-22
  - establishing device connections 5-4, 18-10
  - event messages 20-1
  - event messages, printer 12-35
  - events, viewing 4-2
  - examples
    - calibrate window J-3
    - calibration reminder K-1
    - hardware window J-5
    - initial installation window J-2

created on October 2, 2001

exit lists 5-11

## F

fat-finger condition 21-25

Feature E expansion card ports 14-2

file, resource 3-1, 3-3

fiscal data, reading from the IBM Model 3F Fiscal Printer 12-13

fiscal printer

disable printing 12-33, 19-26

discarding data and fiscal commands 12-33

enable printing 12-34, 19-29

enabling and disabling printing 12-15

event message 20-9

hold printing 12-34

interleaved printing 12-15

power on errors 12-21

printer errors 12-21

printing 12-44

printing a line of text at the printer 12-21

release printing 12-34

reset printer 12-34

status message 20-10

writing data in download logo mode 12-20

writing data in download message mode 12-19

writing data in fiscal mode 12-20

fix font width escape character sequence 12-24

flip check escape character sequence 12-24

flipping a check 12-42

font

4610 SureMark model TI1, TI2, TI3, TI4 printer

fonts 12-9

4689-001 printer fonts 12-6

4689-3x1 and TD5 printer fonts 12-7

Amikake 12-8

contrast 12-8

interactions 12-17

keisen 12-8

line characters 12-8

Model 2 printer fonts 12-2

Model 3 and Model 4 printer fonts 12-3

print direction 12-8

specification 12-4, 12-6, 12-10

switching 12-17

fonts E-1

front or side loaded documents 12-37, 12-41

front or top loaded documents 12-38, 12-40

functions

alarm 6-1

cash drawer 7-1

control, performing 18-8

display 8-3

keyboard 9-12

MSR 10-3

NVRAM 11-2

overview 1-1

printer 12-11

programmable power 13-1

RS-232C 14-2

scale 15-2

functions (*continued*)

scanner 16-4

touch 17-3

## G

getting

cash drawer status 7-2

input 5-2

resource values 19-42

resource values for printer 12-35

guidance lights, setting 8-6, 21-20

## H

Hand-Held Bar Code Reader (HHBCR)

4685 bar code reader 16-1

description 16-1

HHBCR-1 16-1

HHBCR-2 16-1

hardware requirements 1-2

header files, C language 5-14

hold printing 19-31

hot keys

disabling 19-7

enabling 19-10

system 9-15

## I

IBM Point of Sale Subsystem OS/2 and Windows

customizing 3-1

initializing 18-5

improving maintainability of applications 5-19

initializing

applications 5-1

IBM Point of Sale Subsystem 18-5

inline logo escape character sequence 12-25

input, getting 5-2

input/output control requests for the printer 12-33

input,touch 17-1

interleaved printing 12-15

internal keyboard speaker

turning on 19-36

internal printer speaker

turning off 19-19, 19-49

turning on 19-20, 19-50

introduction to the IBM Point of Sale Subsystem 1-1

## K

key

definitions 9-11

Num Lock, controlling 9-14

point-of-sale-unique, controlling 9-14

Scroll Lock, controlling 9-14

system hot keys 9-15

typematic function, controlling 9-15

## keyboard

- 4685 Point of Sale Keyboard Model K01
  - characteristics 9-10
- 50-Key Modifiable Layout Keyboard and Operator Display characteristics 9-3
- 50-Key Modifiable Layout Keyboard
  - characteristics 9-3
- alphanumeric point-of-sale keyboards, configuring 3-4
- alphanumeric point-of-sale keyboards, description 9-2
- ANPOS Keyboard characteristics 9-5
- character received, non-system 20-4
- characteristics 9-1
- checkout 9-1
- click 21-25
- click, controlling 9-14
- default state 9-5, 9-7, 9-8
- device handler 9-1
- double key 21-23
- fat-finger 21-24
- functions 9-12
- internal speaker, turning off 19-19, 19-49
- internal speaker, turning on 19-20, 19-50
- key definitions 9-11
- lights 9-13
- lock, manager 9-13
- microcode updates 9-2
- Modifiable Layout Keyboard with Card Reader
  - characteristics 9-4
- numeric keypad location, specifying 9-15
- numeric keypad style, specifying 9-15
- numeric pad format 21-27
- PC Point of Sale Keyboard characteristics 9-7
- PLU Keyboard and Display-III characteristics 9-9
- point-of-sale considerations 9-13
- Point of Sale Keyboard V characteristics 9-8
- programming 9-1
- reading data 9-12
- related information 9-15
- restrictions 9-12
- Retail Alphanumeric Point of Sale Keyboard Card Reader characteristics 9-6
- retail point-of-sale keyboard, characteristics 9-3
- retail point-of-sale keyboard, description 9-2
- status change message 20-3
- system considerations 9-12
- tone, speaker 9-13
- typematic delay 21-29
- typematic frequency 21-29
- typematic function, controlling 9-15

**L**

- label data, unexpected 16-9
- lights, keyboard 9-13
- line buffering 12-17
- line feed control character 12-15, 12-21
- line length maximum
  - 4610 Model TI1, TI2, TI3, TI4 printers 12-9
  - 4689-001 printer 12-6

- line length maximum (*continued*)
  - 4689-002 printer 12-6
  - 4689-3x1 ant TD5 printers 12-7
  - Model 3, Model 4, and Model 4A printers 12-3
- line spacing for
  - 4689-001 printer 12-6
  - 4689-002 printer 12-6
  - 4689-3x1 and TD5 printers 12-8
  - Model 2 printers 12-2
  - Model 3 and Model 4 printers 12-3, 12-10
- list, exit processing 5-11
- lists, argument 5-12
- loading documents in printer 12-36
- lock, manager keyboard 9-13
- locked state, scanner 16-4
- locking
  - devices 19-43
  - MSR 10-5
  - scanner 16-4
- logged messages, viewing 4-1
- logo mode
  - writing data 12-18

**M**

- macros A-1
- maintainability of applications, improving 5-19
- manager, keyboard lock 9-13
- manual or automatic document insertion 12-37, 12-38, 12-41
- maximum block size for NVRAM read and write 11-1
- memory requirements 1-8
- memory requirements for Microsoft Windows 1-8
- memory requirements for OS/2 1-8
- messages
  - break signal, RS-232C 20-15
  - data available, RS-232C 20-16
  - error C-1
  - error codes D-3
  - event 20-1
  - event, printer 12-35
  - offline device 20-20
  - online device 20-22
  - paper sensor 20-13
  - POS keyboard character 20-28
  - printer cover position 20-13
  - printer error 20-11
  - scanner data available 20-19
  - viewing console 4-1
  - viewing customized dumps 4-1
  - viewing logged 4-1
  - viewing point-of-sale error messages 4-1
- metric weight units 15-1
- MICR Reader 12-43
- MICR recognition and check flipping 12-11
- microcode updates for keyboards 9-2
- microcode updates for touch screen 17-2
- Microsoft Windows considerations for applications 5-17
- mixing different fonts 12-17
- Model 2 printer characteristics 12-1
- Model 3 and Model 4 printer characteristics 12-2

created on October 2, 2001

- Model 3F Fiscal Printer 12-4
- Model 3F printer characteristics 12-4
- Model 3R printer characteristics 12-4
- Model 4A font file format E-1
- Model 4A printer characteristics 12-5
- Model 4R printer characteristics 12-4
- Modifiable Layout Keyboard with Card Reader characteristics 9-4
- modifying resources 5-13
- move to tab escape character sequence 12-26

## MSR

- characteristics 10-1
- data available message 20-6
- device handler 10-1
- Dual-Track, low profile 10-2
- Dual-Track, standard profile 10-2
- functions 10-3
- locking 10-5
- One-Track 10-1
- programming 10-1
- read buffer format 10-4
- reading data 10-4
- related information 10-5
- restriction 10-3
- Three-Track 10-2
- Two-Head 10-3
- Two-Sided 10-3
- unlocking 10-3
- multi-threaded application design 5-18
- multiple thread restriction 18-1

## N

- narrow document insertion 12-40
- non-presentation facility
  - acquiring from 5-8
  - releasing from 5-8
- normal density escape character sequence 12-26
- normal-density escape character sequence 12-15
- normal font escape character sequence 12-15, 12-26
- normal mode
  - printing a line of text at the printer 12-15
  - writing data to the printer 12-14
- notification messages 20-1
- null modem cable used with RS-232C port 14-3
- Num Lock key
  - controlling 9-14
  - disabling 19-8
  - enabling 19-11
  - set off 19-13
  - set on 19-14
- numeric keypad format 21-27
- numeric keypad location, specifying 9-15
- numeric keypad style, specifying 9-15

## NVRAM

- application address space 11-2
- characteristics 11-1
- cursor 11-2
- device handler 11-1
- direct mode, opening in 11-3
- functions 11-2

## NVRAM (continued)

- programming 11-1
- reading data in direct mode 11-3
- reading data in sequential mode 11-3
- related information 11-4
- sequential mode, opening in 11-3
- writing data in direct mode 11-3
- writing data in sequential mode 11-3

## O

- One-Track MSR 10-1
- opening
  - cash drawer till 7-1, 19-48
  - device 18-1
  - NVRAM in direct mode 11-3
  - NVRAM in sequential mode 11-3
- Operator Display, characteristics 8-2
- optimizing application performance 5-16
- OS/2 system hot keys 9-15
- overview of functions 1-1

## P

- paper cutter 12-42
- PC Point of Sale Keyboard characteristics 9-7
- performance considerations, printer 12-47
- performance of applications, optimizing 5-16
- performing
  - control functions 18-8
  - problem determination 4-1
  - problem determination on OS/2 4-1
  - problem determination on the Microsoft Windows operating system 4-2
- PLU Keyboard and Display-III characteristics 9-9
- point-of-sale
  - 4610 SureMark Model characteristics 12-8
  - unique keys, controlling 9-14
- point-of-sale keyboard considerations 9-13
- Point of Sale Keyboard V characteristics 9-8
- polling considerations for applications 5-17
- portrait or landscape orientation 12-42
- POS\_ALARM\_SILENCE\_ALARM 19-3
- POS\_ALARM\_SOUND\_ALARM 19-4
- POS\_DSP\_CLEAR\_SCREEN 19-5
- POS\_DSP\_DEFINE\_CHARACTERS 19-6
- POS\_KBD\_DISABLE\_HOT\_KEYS 19-7
- POS\_KBD\_DISABLE\_NUM\_LOCK 19-8
- POS\_KBD\_DISABLE\_SCROLL\_LOCK 19-9
- POS\_KBD\_ENABLE\_HOT\_KEYS 19-10
- POS\_KBD\_ENABLE\_NUM\_LOCK 19-11
- POS\_KBD\_ENABLE\_SCROLL\_LOCK 19-12
- POS\_KBD\_SET\_NUM\_LOCK\_OFF 19-13
- POS\_KBD\_SET\_NUM\_LOCK\_ON 19-14
- POS\_KBD\_SET\_SCROLL\_LOCK\_OFF 19-15
- POS\_KBD\_SET\_SCROLL\_LOCK\_ON 19-16
- POS\_KBD\_SET\_TYPMATIC\_OFF 19-17
- POS\_KBD\_SET\_TYPMATIC\_ON 19-18
- POS\_KBD\_SILENCE\_TONE 19-19
- POS\_KBD\_SOUND\_TONE 19-20
- POS\_POWER\_OFF 19-21

- POS\_POWER\_ON 19-22
- POS\_POWER\_SET\_ALARM 19-23
- POS\_PRN\_DEFINE\_CHARACTERS 19-24
- POS\_PRN\_DISABLE\_DI\_PRINTING 19-25
- POS\_PRN\_DISABLE\_FISCAL\_PRINTING 19-28
- POS\_PRN\_DISCARD\_DATA 19-27
- POS\_PRN\_ENABLE\_DI\_PRINTING 19-28
- POS\_PRN\_ENABLE\_FISCAL\_PRINTING 19-29
- POS\_PRN\_HOLD\_PRINTING 19-31
- POS\_PRN\_RESET\_PRINTER 19-30
- POS\_PRN\_RESUME\_PRINTING 19-32, 19-33
- POS\_PRN\_RETRY\_PRINTING 19-34
- POS\_PRN\_SILENCE\_TONE 19-35
- POS\_PRN\_SOUND\_TONE 19-36
- POS\_RS232\_SEND\_BREAK 19-37
- POS\_SCALE\_CLEAR\_SCREEN 19-38
- POS\_SCALE\_ZERO\_SCALE 19-39
- POS\_SCAN\_DISCARD\_DATA 19-40
- POS\_SYS\_ACQUIRE\_DEVICE 19-41
- POS\_SYS\_GET\_VALUES 19-42
- POS\_SYS\_LOCK\_DEVICE 19-43
- POS\_SYS\_RELEASE\_DEVICE 19-44
- POS\_SYS\_SET\_VALUES 19-45
- POS\_SYS\_UNLOCK\_DEVICE 19-47
- POS\_TILL\_OPEN\_TILL 19-48
- POS touch 17-5
  - related information 17-5
- POS\_TOUCH\_SILENCE\_TONE 19-49
- POS\_TOUCH\_SOUND\_TONE 19-50
- PosAlarm resource set
  - overview 21-16
  - PosNalarmStatus 21-16
- PosClose subroutine 18-3
- PosDevice resource set
  - overview 21-7
  - PosNdeviceNumber 21-8
  - PosNportNumber 21-14
  - PosNqueueHandle 21-14
  - PosNslotNumber 21-15
- PosDisplay resource set
  - overview 21-16
  - PosNcharSize 21-17
  - PosNdisplayCodePage 21-17
  - PosNdisplayCursor 21-19
  - PosNdisplayLightsOn 21-20
  - PosNdisplayMode 21-19
  - PosNpixelX 21-21
  - PosNpixelY 21-21
- PosDrawer resource set
  - overview 21-21
  - PosNpulseWidth 21-21
  - PosNtillStatus 21-22
- PosInitialize subroutine 18-5
- PosIOCtl requests 19-1
- PosIOCtl subroutine 18-8
- positioning the print head 12-39
- PosKeyboard resource set
  - overview 21-22
  - PosNdoubleKey01 - PosNdoubleKey60 21-23
  - PosNfatFingerTimeout 21-24
  - PosNkeyboardClick 21-25
- PosKeyboard resource set (*continued*)
  - PosNkeyboardLightsOn 21-26
  - PosNkeyLock 21-25
  - PosNnumpadLocation 21-26
  - PosNnumpadStyle 21-27
  - PosNnumpadZero 21-27
  - PosNtoneDuration 21-28
  - PosNtoneFreq 21-28
  - PosNtoneVolume 21-28
  - PosNtypematicDelay 21-29
  - PosNtypematicFreq 21-29
- POSM\_KBD\_STATUS\_CHANGE 20-3
- POSM\_KBD\_WM\_CHAR 20-4
- POSM\_MSR\_DATA\_AVAIL 20-6
- POSM\_PRN\_CHASE\_COMPLETE 20-7
- POSM\_PRN\_DATA\_AVAIL 20-8
- POSM\_PRN\_FISCAL\_ERROR 20-9
- POSM\_PRN\_FISCAL\_STATUS 20-10
- POSM\_PRN\_PRINTER\_ERROR 20-11
- POSM\_PRN\_STATUS\_CHANGE 20-13
- POSM\_RS232\_BREAK\_DETECTED 20-15
- POSM\_RS232\_DATA\_AVAIL 20-16
- POSM\_RS232\_XMIT\_ABORT 20-17
- POSM\_RS232\_XMIT\_COMPLETE 20-18
- POSM\_SCAN\_DATA\_AVAIL 20-19
- POSM\_SYS\_DEVICE\_OFFLINE 20-20
- POSM\_SYS\_DEVICE\_ONLINE 20-22
- POSM\_SYS\_DEVICE\_RELEASED 20-24
- POSM\_TILL\_CLOSED 20-25
- POSM\_TILL\_OPENED 20-26
- POSM\_TOUCH\_DATA 20-27
- PosMsr Resource Set 21-30
  - overview 21-30
- PosNalarmStatus 21-16
- PosNbarCodeProgramming 21-67
- PosNbarCodes1 21-62
- PosNbarCodes2 21-64
- PosNbarCodes3 21-65
- PosNbarCodes4 21-66
- PosNbaudRate 21-53
- PosNbeepFreq 21-67
- PosNbeepLength 21-67
- PosNbeepState 21-68
- PosNbeepVolume 21-68
- PosNblinkLength 21-69
- PosNblock1Type 21-70
- PosNblock2Type 21-71
- PosNblock3Type 21-71
- PosNblockReadMode 21-69
- PosNbVolSwitchState 21-72
- PosNcharSize 21-17
- PosNcheckModulo 21-72
- PosNcode128ScansPerRead 21-73
- PosNcode39ScansPerRead 21-73
- PosNCRWidth 21-35
- PosNdataBits 21-54
- PosNdecodeAlgorithm 21-73
- PosNdeviceNumber 21-8
- PosNdiOrientation 21-35
- PosNdisplayCodePage 21-17
- PosNdisplayCursor 21-19

created on October 2, 2001

PosNdisplayLightsOn 21-20  
PosNdisplayMode 21-19  
PosNdisplayRequired 21-57  
PosNDIWidth 21-35  
PosNdoublekey 21-23  
PosNdReadTimeout 21-74  
PosNdTouchMode 21-74  
PosNeAN13ScansPerRead 21-75  
PosNeAN8ScansPerRead 21-75  
PosNfatFingerTimeOut 21-24  
PosNfeedDirection 21-36  
PosNfiscalCountry 21-37  
PosNfiscalNotify 21-37  
PosNfiscalPLDStatus 21-37  
PosNfiscalVersion 21-38  
PosNheadParkedPosition 21-38  
PosNinterleaved 21-39  
PosNiTFLength1 21-76  
PosNiTFLength2 21-77  
PosNiTFLengthType 21-77  
PosNiTFScansPerRead 21-78  
PosNjANTwoLabelDecode 21-78  
PosNkeyboardClick 21-25  
PosNkeyboardLightsOn 21-26  
PosNkeyLock 21-25  
PosNlabelsQueued 21-78  
PosNlaserSwitchState 21-79  
PosNlaserTimeout 21-79  
PosNleftMarginCR 21-39  
PosNlineFeedCR 21-40  
PosNlineFeedDI 21-40  
PosNlineFeedSJ 21-41  
PosNlineMode 21-54  
PosNmotorTimeout 21-80  
PosNnumpadLocation 21-26  
PosNnumpadStyle 21-27  
PosNnumpadZero 9-4, 9-6, 21-27  
PosNnumWeightDigits 21-57  
PosNnvramCursor 21-30  
PosNnvramMode 21-31  
PosNnvramSize 21-31  
PosNoperMode 21-58  
PosNparity 21-55  
PosNpixelX 21-21  
PosNpixelY 21-21  
PosNportNumber 21-14  
PosNpowerAlarm 21-31  
PosNprintAlignment 21-42  
PosNprintColorMode 21-42  
PosNprintCRCharSetx 21-42  
PosNprintDICharSetx 21-43  
PosNprintFeatures 21-43  
PosNprintMode 21-44  
PosNprintQualityMode 21-44  
PosNprintStation 21-45  
PosNprintStatus 21-46  
PosNprintStatus2 21-47  
PosNprintTabStops 21-47  
PosNprintToneDuration 21-48  
PosNprintToneFrequency 21-48  
PosNprintToneNote 21-48  
PosNprintToneOctave 21-49  
PosNprintToneVolume 21-49  
PosNprintUpsideDown 21-50  
PosNpulseWidth 21-21  
PosNqueueAllLabels 21-80  
PosNqueueHandle 21-6, 21-14  
PosNrawPrintStatus 21-50  
PosNreadTimeout 21-6  
PosNresumeString 21-50  
PosNretryString 21-52  
PosNrs232Status 21-55  
PosNscansPerRead 21-81  
PosNslotNumber 21-15  
PosNstopBits 21-55  
PosNstoreScansPerRead 21-81  
PosNsupplementals 21-82  
PosNtillStatus 21-22  
PosNtimeoutChar 21-56  
PosNtoneDuration 21-28  
PosNtoneFreq 21-28  
PosNtoneVolume 21-28  
PosNtouchBackLightOnEvent 21-90  
PosNtouchBrightness 21-90  
PosNtouchClickVolume 21-91  
PosNtouchContrast 21-91  
PosNtouchEntryClick 21-91  
PosNtouchExitClick 21-92  
PosNtouchMaxX 21-92  
PosNtouchMaxY 21-92  
PosNtouchMode 21-92  
PosNtouchScreenSaverTime 21-93  
PosNtouchToneDuration 21-93  
PosNtouchToneFreq 21-93  
PosNtouchToneVolume 21-94  
PosNtwoLabelFlagPair1 21-83  
PosNtwoLabelFlagPair2 21-84  
PosNtwoLabelFlagPair3 21-85  
PosNtwoLabelFlagPair4 21-86  
PosNtypematicDelay 21-29  
PosNtypematicFreq 21-29  
PosNuPCAScansPerRead 21-87  
PosNuPCDScansPerRead 21-87  
PosNuPCEScansPerRead 21-88  
PosNuPCEExpansion 21-88  
PosNverifyPriceChk 21-89  
PosNvibrationFilter 21-58  
PosNvitalProductData 21-7  
PosNvram resource set  
    overview 21-30  
    PosNnvramCursor 21-30  
    PosNnvramMode 21-31  
    PosNnvramSize 21-31  
PosNweightMode 21-59  
PosNzeroIndState 21-59  
PosNzeroRetState 21-60  
PosOpen subroutine 18-10  
PosPower resource set  
    overview 21-31  
    PosNpowerAlarm 21-31  
PosPrinter resource set  
    overview 21-32

PosPrinter resource set *(continued)*

PosNCRWidth 21-35  
 PosNdiOrientation 21-35  
 PosNDIWidth 21-35  
 PosNfeedDirection 21-36  
 PosNfiscalCountry 21-37  
 PosNfiscalNotify 21-37  
 PosNfiscalPLDStatus 21-37  
 PosNfiscalVersion 21-38  
 PosNheadParkedPosition 21-38  
 PosNinterleaved 21-39  
 PosNleftMarginCR 21-39  
 PosNlineFeedCR 21-40  
 PosNlineFeedDI 21-40  
 PosNlineFeedSJ 21-41  
 PosNprintAlignment 21-42  
 PosNprintColorMode 21-42  
 PosNprintCRCharSetx 21-42  
 PosNprintDICharSetx 21-43  
 PosNprintFeatures 21-43  
 PosNprintMode 21-44  
 PosNprintQualityMode 21-44  
 PosNprintStation 21-45  
 PosNprintStatus 21-46  
 PosNprintStatus2 21-47  
 PosNprintTabStops 21-47  
 PosNprintToneDuration 21-48  
 PosNprintToneFrequency 21-48  
 PosNprintToneNote 21-48  
 PosNprintToneOctave 21-49  
 PosNprintToneVolume 21-49  
 PosNprintUpsideDown 21-50  
 PosNrawPrintStatus 21-50  
 PosNresumeString 21-50  
 PosNretryString 21-52

## POSQMSG 20-1

PosRead subroutine 18-16

## PosRs232c resource set

overview 21-53  
 PosNbaudRate 21-53  
 PosNdataBits 21-54  
 PosNlineMode 21-54  
 PosNparity 21-55  
 PosNrs232Status 21-55  
 PosNstopBits 21-55  
 PosNtimeoutChar 21-56

## PosScale resource set

overview 21-56  
 PosNdisplayRequired 21-57  
 PosNnumWeightDigits 21-57  
 PosNoperMode 21-58  
 PosNvibrationFilter 21-58  
 PosNweightMode 21-59  
 PosNzeroIndState 21-59  
 PosNzeroRetState 21-60

## PosScanner resource set

overview 21-60  
 PosNbarCodeProgramming 21-67  
 PosNbarCodes1 21-62  
 PosNbarCodes2 21-64  
 PosNbarCodes3 21-65

PosScanner resource set *(continued)*

PosNbarCodes4 21-66  
 PosNbeepFreq 21-67  
 PosNbeepLength 21-67  
 PosNbeepState 21-68  
 PosNbeepVolume 21-68  
 PosNblinkLength 21-69  
 PosNblock1Type 21-70  
 PosNblock2Type 21-71  
 PosNblock3Type 21-71  
 PosNblockReadMode 21-69  
 PosNbVolSwitchState 21-72  
 PosNcheckModulo 21-72  
 PosNdecodeAlgorithm 21-73  
 PosNdReadTimeout 21-74  
 PosNdTouchMode 21-74  
 PosNeAN13ScansPerRead 21-75  
 PosNeAN8ScansPerRead 21-75  
 PosNiTFLength1 21-76  
 PosNiTFLength2 21-77  
 PosNiTFLengthType 21-77  
 PosNjANTwoLabelDecode 21-78  
 PosNlabelsQueued 21-78  
 PosNlaserSwitchState 21-79  
 PosNlaserTimeout 21-79  
 PosNmotorTimeout 21-80  
 PosNqueueAllLabels 21-80  
 PosNscansPerRead 21-81  
 PosNstoreScansPerRead 21-81  
 PosNsupplementals 21-82  
 PosNtwoLabelFlagPair1 21-83  
 PosNtwoLabelFlagPair2 21-84  
 PosNtwoLabelFlagPair3 21-85  
 PosNtwoLabelFlagPair4 21-86  
 PosNuPCAScansPerRead 21-87  
 PosNuPCDScansPerRead 21-87  
 PosNuPCEscansPerRead 21-88  
 PosNuPCExpansion 21-88  
 PosNverifyPriceChk 21-89

PosSetArg macro, using 5-13

## PosSystem resource set

example 21-5  
 overview 21-5  
 PosNqueueHandle 21-6  
 PosNreadTimeout 21-6  
 PosNvitalProductData 21-7

## PosTouch resource set

PosNtouchBackLightOnEvent 21-90  
 PosNtouchBrightness 21-90  
 PosNtouchClickInactive 21-92  
 PosNtouchClickVolume 21-91  
 PosNtouchContrast 21-91  
 PosNtouchEntryClick 21-91  
 PosNtouchMaxX 21-92  
 PosNtouchMaxY 21-92  
 PosNtouchMode 21-92  
 PosNtouchScreenSaverTime 21-93  
 PosNtouchToneDuration 21-93  
 PosNtouchToneFreq 21-93  
 PosNtouchToneVolume 21-94

PosTouch Resource Set 21-89

created on October 2, 2001

PosTouch Resource Set (*continued*)

- overview 21-89
- PosWrite subroutine 18-19
- power supply
  - 4693 13-2
  - querying time for power to be turned on 13-3
  - set day and time to restore 19-23
  - turn off 19-21
  - turn on 19-22
- pre-defined messages and logos 12-10
- preparing to read MSR data 10-3
- presentation facility
  - acquiring devices 5-7
  - POS keyboard character message 20-28
  - releasing devices 5-7
- Presentation Manager considerations for applications 5-16
- print barcode escape character sequence 12-27
- print pre-defined logo escape character sequence 12-28
- print pre-defined message escape character sequence 12-28
- printer
  - 3F Fiscal Printer restrictions 12-4
  - 4689-001 printer characteristics 12-6
  - 4689-002 printer characteristics 12-6
  - 4689-00x Printer restrictions 12-6, 12-8
  - 4689-301 printer characteristics 12-7
  - addressable print positions 12-5
  - addressable print positions per character, maximum 12-5
  - addressable print positions per line, maximum 12-5
  - Amikake specification 12-8
  - barcode printing 12-10
  - changing print characteristics 12-15
  - characteristics 12-1
  - characters per line, maximum 12-2, 12-3, 12-6, 12-7, 12-9
  - chase escape character sequence 20-7
  - check printing 12-43
  - code page support 12-11
  - contrast specification 12-8
  - control characters 12-21
  - controlling 12-48, 17-5
  - data alignment 12-22
  - data available message 20-8
  - defining characters 19-24
  - determining when printing is complete 12-17
  - device handler 12-1
  - DI station character line lengths 12-39
  - disable DI station 12-33, 19-25
  - disable fiscal printing 12-33
  - discarding data 12-33
  - document insert station, controlling 12-36
  - double-strike printing 12-2, 12-4, 12-6, 12-8, 12-11
  - emphasized printing 12-2, 12-4, 12-6, 12-8
  - enable DI station printing 12-33
  - enable fiscal printing 12-34
  - enabling and disabling fiscal printing 12-15
  - error message 20-11
  - errors in print stations, download logo mode 12-20

printer (*continued*)

- errors in print stations, logo mode 12-19
- errors in print stations, normal mode 12-15
- escape character sequences 12-22
- event messages 12-35
- fiscal event message 20-9
- fiscal printing 12-44
- fiscal status message 20-10
- font interactions 12-17
- font specification 12-4, 12-6, 12-10
- fonts 12-2, 12-3, 12-6, 12-7, 12-9
- functions 12-11
- getting resource values 12-35
- hold 12-34
- hold printing 12-34
- input/output control requests 12-33
- interleaved printing 12-15
- keisen specification 12-8
- line buffering 12-17
- line characters specification 12-8
- line length 12-2, 12-3, 12-6, 12-7, 12-9
- logo printing 12-18
- logo printing, speed 12-2, 12-4, 12-6, 12-8, 12-11
- Model 2 characteristics 12-1
- Model 3 and Model 4 characteristics 12-2
- Model 3F Fiscal Printer 12-4
- Model 3R and Model 4R characteristics 12-4
- Model 4A characteristics 12-5
- paper sensor message 20-13
- performance considerations 12-47
- power on errors, fiscal mode 12-21
- pre-defined messages and logos 12-10
- print direction specification 12-8
- print head positioning 12-39
- print mechanism 12-1, 12-3, 12-6, 12-7, 12-9
- printer cover position message 20-13
- printer errors, fiscal mode 12-21
- printing a line of text, normal mode 12-15
- printing a line of text at the printer, fiscal mode 12-21
- programming 12-1
- queues 12-35
- reading data from 12-12
- reading fiscal data 12-13
- receipt paper cutter 12-42
- reinserting documents 12-37, 12-39, 12-41
- related information 12-48
- release 12-34
- release printing 12-34
- reset 12-34
- reset printer 12-34
- resources 12-35
- resume printing 19-33
- resume printing after an error 12-34
- retry printing 19-34
- retry printing after an error 12-34
- reversible document station motor 12-39
- setting resource values 12-34
- silence tone 19-35
- speed for printing 12-2, 12-4, 12-6, 12-8, 12-11
- status, determining 12-35

printer (*continued*)

- user-defined characters 12-44
- writing data in download logo mode 12-20
- writing data in download message mode 12-19
- writing data in fiscal mode 12-20
- writing data in normal mode 12-14
- writing data to 12-14

printing checks 12-43

priority of applications 5-16

problem determination 4-1

problem determination on OS/2 4-1

problem determination on the Microsoft Windows operating system 4-2

processing, exit list 5-11

processing unexpected scanner data 16-9

programmable power device

- characteristics 13-1
- clearing alarm setting 13-2
- device handler 13-1
- functions 13-1
- power, turning on and off to 4693 13-2
- programming 13-1
- querying time for power to be turned on 13-3
- related information 13-3

programming, device 5-1

programs, sample

- aiptstr 1-1
- anposkey 1-1
- checkout 1-1
- demo 1-1

pulse width, cash drawer 7-2

## Q

querying time for power to be turned on 13-3

queued data, discarding 19-27

queues, printer 12-35

## R

read MICR escape character sequence 12-28

Read MICR escape character sequence 12-15

read operation suspension 18-1

reading

- data, RS-232C 14-3
- data from MSR 10-4
- data from MSR, preparing to 10-3
- fiscal data 12-13
- fiscal data from the printer 12-13
- from devices 5-9, 18-16
- from MICR reader 12-12
- keyboard data 9-12
- MICR data from the printer 12-12
- MSR buffer format 10-4
- NVRAM data in direct mode 11-3
- NVRAM data in sequential mode 11-3
- scale data 15-2
- scanner data 16-4
- touch screen data 17-3

receipt paper cutter 12-42

receiving

- break signal from RS-232C 19-37

register document escape character sequence 12-15, 12-29

reinserting documents for printing 12-37, 12-39, 12-41

release document escape character sequence 12-15, 12-29

released device event message 20-24

releasing devices 19-44

- from non-presentation facility 5-8
- from presentation facility 5-7
- general information 5-6

releasing documents 12-37, 12-39, 12-41, 12-42

requests, PosIOCtl 19-1

requirements

- disk space 1-8
- hardware 1-2
- Microsoft Windows disk space 1-8
- Microsoft Windows memory 1-8
- OS/2 disk space 1-8
- OS/2 memory 1-8
- software 1-7

requirements for system 1-2

reset printer 19-30

resource

- 4500 HHBCR default values 16-10
- 4501 HHBCR default values 16-10
- 4585 HHBCR default values 16-10
- 4686 Scanner default values 16-10
- 4687 Scanner default values 15-5, 16-11
- 4696 Scanner Scale default values 15-5, 16-11
- 4697 Scanner default values 16-11
- argument lists 5-12
- file 3-1
- file, using 3-3
- getting values for printer 12-35
- modifying 5-13
- PosIOCtl() values, getting 19-42
- PosIOCtl() values, setting 19-45
- printer 12-35
- retrieving 5-13
- setting values for printer 12-34
- USB Scanner Scale default values 15-5
- using 5-12

restoring power supply 19-23

restriction

- keyboard device handler 9-12
- MSR device handler 10-3

resume printing 19-32, 19-33

resume printing after an error 12-34

Retail Alphanumeric Point of Sale Keyboard with Card Reader characteristics 9-6

retail point-of-sale keyboards

- characteristics 9-3
- description 9-2

retrieving resources 5-13

retry printing 19-34

retry printing after an error 12-34

right column align escape character sequence 12-29

RS-232C

- break signal message 20-15

created on October 2, 2001

## RS-232C (continued)

- characteristics 14-2
- controlling the port 14-2
- data available message 20-16
- device handler 14-1
- error definitions 14-3
- functions 14-2
- null modem cable 14-3
- programming 14-1
- reading data 14-3
- receiving break signal 19-37
- related information 14-4
- status 14-4
- subroutines, used with 14-4
- writing data 14-4

RTS 14-2

## S

### sample programs

- aiptstr 1-1
- anposkey 1-1
- checkout 1-1
- demo 1-1

scalable fonts escape character sequence 12-29

### scale

- 4687 Scanner, default resource values 15-5
- 4687 Scanner Scale Model 2 15-1
- 4696 Scanner Scale, default resource values 15-5
- characteristics 15-1
- clearing display 15-4
- configuring 15-4
- data buffer format 15-2
- default modes 15-4
- device handler 15-1
- functions 15-2
- IBM 4696 Scanner Scale Model 1 15-1
- programming 15-1
- reading data 15-2
- related information 15-5
- USB Scanner Scale, default resource values 15-5
- zeroing 15-4, 19-39

Scale, USB interface 15-2

- description 15-2

### scanner

- 1520-A02 16-2
- 4685 Scanner 16-1
- 4686 Scanner 16-2
- 4686 Scanner default values 16-10
- 4687 Scanner 16-2
- 4687 Scanner default resource values 16-11
- 4696 Scanner Scale 16-2
- 4696 Scanner Scale default resource values 16-11
- 4697 Scanner 16-3
- 4697 Scanner default resource values 16-11
- 4698 Scanner 16-3
- characteristics 16-1
- configuring 16-8
- data available, message 20-19
- data buffer format 16-5
- default modes 16-9

### scanner (continued)

- device handler 16-1
- discard data 19-40
- discarding scanner data 16-8
- functions 16-4
- HHBCR-1 16-1
- HHBCR-2 16-1
- HHBCR default resource values 16-10
- locking 16-4
- programming 16-1
- reading scanner data 16-4
- related information 16-13
- supported devices 16-1
- unexpected data, processing 16-9
- unlocking 16-4
- USB Scanner 16-3

### Scroll Lock key

- controlling 9-14
- disabling 19-9
- enabling 19-12
- set off 19-15
- set on 19-16

select color printing escape character sequence 12-30

### sequential mode

- opening NVRAM in 11-3
- reading NVRAM data in 11-3

### setting

- argument list values 5-12
- cash drawer pulse width 7-2
- device resource values 19-45
- guidance lights, display 8-6, 21-20
- resource values for printer 12-34

### Shopper Display

- characteristics 8-2
- setting guidance lights 8-6, 21-20

shutdown sequence, API 18-1

silencing the alarm 6-1, 19-3

### SJ (summary journal) station

- fonts 12-2, 12-3, 12-6, 12-7, 12-9
- function 12-1
- interleaved printing, normal mode 12-15
- line spacing 12-2, 12-3, 12-6, 12-8, 12-10
- maximum addressable print positions per character 12-5
- maximum addressable print positions per line 12-5
- maximum characters per line 12-2, 12-3, 12-6, 12-7, 12-9
- printer errors, download logo mode 12-20
- printer errors, logo mode 12-19
- printer errors, normal mode 12-15
- queues 12-35

small font escape character sequence 12-31

software requirements 1-7

sounding the alarm 6-1, 19-4

speaker tone, keyboard 9-13

### specifying

- numeric keypad location 9-15
- numeric keypad style 9-15

speed for printing on 4610 SureMark Point of Sale Printer models 12-11

speed for printing on 4689-001 printer 12-6

- speed for printing on 4689-002 printer 12-6
- speed for printing on 4689-3x1 ant TD5 printer 12-8
- speed for printing on Model 2 printers 12-2
- speed for printing on Model 3 and Model 4 printers 12-4
- spread font escape character sequence 12-15, 12-31
- status
  - alarm 6-2
  - cash drawer 7-2
  - change message 20-3
  - fiscal message 20-10
  - printer, determining 12-35
  - RS-232C, getting 14-4
  - touch message 20-27
- successful PosWrite(), message 20-18
- switching fonts 12-17
- system keyboard considerations 9-12
- system requirements 1-2

## T

- terminating applications 5-11
- text attributes escape character sequence 12-32
- Three-Track MSR 10-2
- till, cash drawer
  - closed message 20-25
  - open message 20-26
  - opening 7-1
- tone, keyboard
  - silence 19-19, 19-49
  - sound 19-20, 19-36, 19-50
  - touch data message 20-27
- tone, keyboard speaker 9-13
- top or front loaded documents 12-38, 12-40
- touch
  - 4820 SurePoint Solution J-1
  - audible feedback, controlling 17-4
  - backlight on event messages, controlling 17-5
  - characteristics 17-1
  - determining which touch screen is available 17-4
  - device handler 17-1
  - double-click sensitivity adjustment on Windows 17-2
  - error codes 17-6
  - event messages 17-5
  - functions 17-3
  - input 17-1
  - LCD brightness, controlling 17-4
  - LCD contrast, controlling 17-4
  - microcode updates 17-2
  - mouse emulation on OS/2 17-2
  - mouse emulation on Windows 17-2
  - PosIOctl control requests 17-5
  - programming 17-1
  - reading data 17-3
  - resources 17-5
  - restriction of touch screen device handler 17-3
  - screen saver time, controlling 17-5
  - subroutines used 17-5
  - tone output 17-2
  - tone,using 17-3
  - touch mouse emulation 17-2

- touch (*continued*)
  - video 17-1
- touch screen calibration tool
  - RS485 touchsScreen J-1
- trace codes B-1
- trace events, viewing 4-1
- turning off power supply 19-21
- turning on power supply 19-22
- Two-Head MSR 10-3
- Two-Sided MSR 10-3
- Two-Sided Vacuum Fluorescent Display II,
  - characteristics 8-3
- typematic delay 21-29
- typematic frequency 21-29
- typematic function
  - controlling 9-15
  - set off 9-15, 19-17
  - set on 9-15, 19-18

## U

- unexpected scanner data, processing 16-9
- unlock device 19-47
- unlocked state, scanner 16-4
- unlocking the MSR 10-3
- updating microcode G-1
- USB scale interface 15-5
  - default resource values 15-5
- USB Scanner 16-3
  - description 16-3
- USB scanner interface 16-12
  - default resource values 16-12
- user-defined fonts escape character sequence 12-31
- using resources 5-12
- using the resource file 3-3
- using the touch screen calibration tool J-2

## V

- video display, touch screen 17-1
- viewing
  - console messages 4-1
  - customized dumps 4-1
  - events 4-2
  - logged messages 4-1
  - point-of-sale error logs 4-2
  - point-of-sale error messages 4-1
  - trace events 4-1

## W

- wide document insertion 12-40
- WM\_CHAR 20-28
- writing
  - bitmaps to Character and Graphics Display 8-5
  - characters to display 8-4
  - data, RS-232C 14-4
  - data in download logo mode 12-20
  - data in download message mode 12-19
  - data in fiscal mode 12-20
  - data in logo mode to printer 12-18

created on October 2, 2001

writing (*continued*)

NVRAM data in direct mode 11-3

NVRAM data in sequential mode 11-3

printer data 12-14

to devices 5-10, 18-19

writing data to the printer in normal mode 12-14

## **Z**

zeroing the scale 15-4, 19-39

created on October 2, 2001





created on October 2, 2001



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SC30-3560-11

