

TPG OPOS Driver

USER GUIDE

189-9106172C



Copyright 2008 TPG IPB, Inc., a subsidiary of ATSI Holdings, Inc. All rights reserved.



This page intentionally left blank



Table of Contents

1	Overview.....	1
2	Installation.....	2
3	Registry Usage.....	2
3.1	User-Set Registry Variables.....	2
3.1.1	POSPrinter Communications.....	2
3.1.2	POSPrinter Code 128 barcodes.....	3
3.1.3	POSPrinter Slip Positioning.....	3
3.1.4	POSPrinter Code Page Selection.....	3
3.1.5	MICR Error Codes.....	3
3.1.6	CashDrawer Timing, Status and Signal variables.....	4
3.1.7	Registry Trace Variables.....	4
4	Implementation.....	4
4.1	Version 1.8 specific.....	4
5	Check Scanner.....	4
6	Interpretations.....	6
6.1	Slip AutoPosition.....	6
6.2	Slip Status Event Firing.....	6
6.3	Hardware Detection.....	6
6.4	Receipt Paper Cuts.....	7
6.5	Barcode Printing.....	7
6.5.1	Barcode Data Errors.....	7
6.5.2	Barcode Heights.....	8
6.5.3	Barcode Widths.....	8
6.5.4	Code 128 Printing.....	10
6.5.5	UPC-E Printing.....	10
6.5.6	PDF 417 Barcode Printing.....	11
6.6	Limitations on Bitmaps.....	11
6.7	Character Set Selection.....	11
6.8	RecLineSpacing/RecLineSpacing.....	11
6.9	DotsPerPrinthead/RecLineHeight.....	12
6.10	Rotated Printing.....	12
6.11	CapRec2Color.....	13
6.12	Print Data Alignment.....	13
6.13	Changing the SlpLineSpacing Property.....	13
6.14	Receipt Printing with a Slip Inserted.....	13
6.15	ValidateData Implementation.....	13
6.16	CashDrawer Open.....	14
6.17	StatusUpdateEvents.....	14
6.18	Setting the DataEventEnabled Property.....	14
7	TPG Additions.....	14
7.1	Multiple Downloaded Bitmaps Using SetBitmap.....	14
7.2	Check Flip.....	14
7.3	Extended Error Codes for Improved Error Handling.....	14
7.4	DirectIO.....	15



7.5	MICR Exception Table Support	21
7.6	Exception Table Entry	21
7.7	Examples	22
7.8	Application Header File	22
8	Appendices	23
8.1	Appendix A -- Setting a default code page	23
8.2	Usage and behavior	23
8.3	Example Registry Values	23
8.4	A794 A715 CP850 Emulation	24
8.5	Setting DefaultCodePage in an A7xx.reg file	24
8.6	Appendix B -- Code 128 Barcode Parsing	25
8.7	Format of data presented to the OPOS driver (method 1)	25
8.8	Full ASCII method (Method 2)	26
8.9	The DEL Character	26
8.10	Migration to UPOS version 1.8 and Beyond-Use of Registry Entries	28
8.11	Code128Parse	28
8.12	Code128App18	28
8.13	Code128Parse18	28
8.14	Appendix C -- Changes to the CashDrawer Service Object	30
8.15	PollInterval	30
8.16	StatusDelay	30
8.17	DisableImmediatePoll	30
8.18	CashDrawer_Open	30
8.19	Appendix D -- Print Commands for Printing 90° Rotated	31
8.20	Appendix E --Registry Reference	32
8.21	CashDrawer	32
8.22	MICR (Does not apply to the A794 or A795)	32
8.23	POSPrinter	33



1 Overview.

This driver is an implementation of the OLE for Retail POS driver described in the Application Programmer's Guide, (APG) Release 1.8 and the Control Programmer's Guide, (CPG) Release 1.8. Both documents are available at:

<http://www.monroecs.com/>

TPG provides a single dynamically linked library file (*tpg.dll*) containing Service Objects (SO) for UPOS devices POSPrinter, CashDrawer, MICR and CheckScanner.

The TPG SO has been tested with Control Objects provided by Monroe Consulting Services (address previously given). Follow the News link, then the OPOS CCO Current Version link to whichever CO is required. It is then possible to download and install the CO. The only stipulation is that to obtain the full functionality of the driver, the CO must be equal to or higher than the SO in major version.

This driver is supported using the TPG native USB interface, Windows Printer Class USB and the RS-232 interface. The TPG native USB driver can work in the Windows 2000/XP/Vista operating systems, and can be obtained from

www.tpgprinters.com.

The RS-232 configuration of this driver can communicate with an RS-232 printer through a PC COM port, and it communicates with a USB printer through a virtual COM port, which is created through the use of the TPG RS-232 Emulation USB driver. This driver is available from www.tpgprinters.com. The printer firmware must be configured to operate with the selected communications method. Consult the *Printer's User Guide* (USB Appendix) for details.

This driver is functional in the Windows 2000/XP/Vista operating systems. The SO and CO must be installed running with Administrative privilege, but can operate at a lesser privilege level.

NOTE: *An OPOS CheckScanner device is now part of this distribution and is documented separately. All devices are version 1.8.*

Notes:

- The A794 or A795 does not have slip-handling capabilities; any references to slip handling, or MICR, in the following do not apply.
- Single station printers are printers that have only thermal receipt capability such as the A794 and A795.
- Dual station or *hybrid* printers are printers with thermal receipt and impact slip optional MICR and or Flip such as the A758, A760 and A776.
- Any references to color printing apply only to A795, A760 and A776 printers.



2 Installation.

Microsoft installer databases contain the registry settings and the *tpg.dll* Service Object and are used to install/uninstall the SO. Control objects (if required) are installed separately, as is a sample application and this documentation.

The SO is copyrighted in the *Description* key in the system registry.

Each CO is copyrighted in the About Box. The Common Control Objects are copyright by Research Computer Services, Inc.

The test application is also copyrighted in the About Box.

A general usage license is provided in the file LICENSE.PDF

3 Registry Usage.

The Windows Installer objects contain the device specific information that is entered into the Windows registry when installing the SO. This information is used for the following purposes.

1. Provide a known place in the registry for the CO to locate the SO that supports the device the application wants to open.
2. Provide device specific information that is used by the SO for customization.

The registry key used for OPOS devices is:

HKLM\Software\OLEforRetail\ServiceOPOS\<DeviceCategory>\<DeviceName>
<DeviceCategory> is the OPOS name of a device type, e.g. POSPrinter, MICR
<DeviceName> is a specified device, such as A776, or A795.

See Appendix E for information concerning registry values.

3.1 User-Set Registry Variables

The TPG OPOS Service Objects (SO) contain a number of variables set in the Windows registry. Most of these are placed there by the SO and should not be altered. In some cases, however, it is proper that the end user change certain registry settings. This documents those settings. *All other variables should be left unchanged.*

3.1.1 POSPrinter Communications

Port	Either of “USB” or “COMx”, where ‘x’ is the number of the RS-232 communications port to which the printer is connected. Default is “USB”.
Protocol	If Port is “USB”, then Protocol must likewise be set to “USB”. If port is set to “COMx”, the protocol must be either “DTR” or “XON”, depending upon the printer configuration. Default is “USB”.
Baudrate	Applies only to serial connections. Default is “115200”.
Parity	Applies only to serial connections. Default is “NONE”.



UsbType For USB connections, this value specifies either TPG native USB or Printer class USB. A value of “0” specifies TPG native USB; a value of “1” specifies Printer class. The printer configuration and the registry setting must agree. Default is “1” (Printer class.)

3.1.2 POSPrinter Code 128 barcodes

All variables beginning with “**Code128...**” are user configurable. These are described in Appendix B.

Defaults are (OPOS Version 1.8)

- **Code128** “1”
- **Code128DefaultStart** “103”
- **Code128Parse** “1”
- **Code128App18** “0”
- **Code128Parse18** “0”
- **Code128CheckIncl** “0”

3.1.3 POSPrinter Slip Positioning

TopOfSlipPosition

- “0” No paper movement when the print begins
- “1” Printing starts 0.7” (17.8 mm) from the top edge of the paper
- “2” Prints starts at the top edge of the paper

Default is “1”. Applies only to the A776/B780 POSPrinter

3.1.4 POSPrinter Code Page Selection

DefaultCodePage Described in [Appendix A](#). Default is “999”.

3.1.5 MICR Error Codes

Previous to Version 1.7, several MICR extended status codes were provided by TPG. UPOS version 1.7 introduced new status codes. The registry variable **UseTpgMICRStatusCodes** is provided for backward application compatibility.

If **UseTpgMICRStatusCodes** is nonzero, then the MICR SO returns the pre-V1.7 codes; otherwise the new codes are returned. Default value is “0”

```
const LONG OPOS_EMICR_BADDATA      = 203;
const LONG OPOS_EMICR_NODATA      = 204;
const LONG OPOS_EMICR_BADSIZE     = 205;
const LONG OPOS_EMICR_JAM         = 206;
const LONG OPOS_EMICR_CHECKDIGIT  = 207;
const LONG OPOS_EMICR_COVEROPEN   = 208;
```



Equivalent values are

TPG_BADREAD_DATA can be replaced by OPOS_EMICR_BADDATA

TPG_NOCHECK_NODATA can be replaced by OPOS_EMICR_NODATA

TPG_PAPERJAM_NODATA can be replaced by OPOS_EMICR_JAM

TPG_NOCHARS_NODATA can be replaced by OPOS_EMICR_NODATA

3.1.6 CashDrawer Timing, Status and Signal variables

See Appendix C. The variables in question are

PollInterval	Default is “ 50 ”
StatusDelay	Default is “ 150 ”
DisableImmediatePoll	Default is “ 0 ”
PulseOnTime	Default is “ 65 ”
PulseOffTime	Default is “ 65 ”
CashDrawer_Open	Default is “ HIGH ”

3.1.7 Registry Trace Variables

Each SO contains a sub key, **Trace**, containing data for tracing. Enabling tracing can slow the SO considerably.

Level If nonzero, the driver outputs trace data to a specified file. Default is “**0**”.

FileName Specifies the file to which trace data is written. Default is “**c:\Temp\so_xxx.txt**”. Each SO has a separate file, but they have been shared.

FileLenMax max size of the trace file in kilobytes.

4 Implementation.

The TPG OPOS SO adheres to the OPOS specification, which describes how the driver works in most cases. There are two situations that merit detailed description of the operation of the OPOS driver:

1. The OPOS specification does not define the operation of the driver in a particular situation that is within normal device operation.
2. TPG has implemented something outside the specification.

4.1 Version 1.8 specific

Version 1.8 of the UPOS specification introduces the **CapStatisticsReporting** and **CapUpdateStatistics** capabilities for all devices. Both of these capabilities are **FALSE** in the TPG OPOS Service Objects. This implies that the methods **ResetStatistics**, **RetrieveStatistics** and **UpdateStatistics** return OPOS_E_ILLEGAL for all devices.

5 Check Scanner

The TPG OPOS CheckScanner Service Object (SO) version 1.7 generally follows the UPOS specification for the CheckScanner. Future releases will extend the functionality by means of further refinement and extension of the existing interface and by means of functionality implemented by the DirectIO request.

Registry usage. (*HKLM\Software\OleforRetail\ServiceOPOS\CheckScanner\A776*)



The values under *Trace* are used the same as the other service objects; $0 < Level$ forces tracing to be enabled and trace messages are added to the file named by *fileName*. Most of the data/value pairs under `~\CheckScanner\A776` should not be changed. Ones that may be:

1. To obtain MICR data using the **Check Scanner** (OPOS device) the following programming sequence can be used:
 - **Open** the **MICR** service object
 - **Open, Claim and Enable** the **Check Scanner** service object
 - Set the **ConcurrentMICR** property to **TRUE**
 - Scan an image using **BeginInsertion** and then **EndInsertion**
 - After the **Check Scanner** service delivers the **StatusUpdateEvent** at scan completion, call **RetrieveImage**, and enable **DataEvent** on both the **CheckScanner** and **MICR** objects. MICR data should then be delivered to the appropriate **MICR** properties.
2. Imaged document size is automatically determined and returned in **DocumentHeight** and **DocumentWidth** properties, in term **MapMode** units, default **English** (inches).
3. Crop areas are not supported: the **CapDefineCropArea** is always set to **FALSE**.
4. **StoreImageFiles** is false; however a large circular buffer is used so that images that have not been cleared or overwritten can be retrieved, even if different formats (the grayscale version remains as scanned in the printer buffer).
5. The **Quality** and **QualityList** properties are fixed at 200 dpi.
6. Internal **CheckHealth** is supported at the same level as **MICR**; the driver sends a buffered status sequence and the test passes if the scanner responds. Other levels of check health are unsupported.

The following table shows the values of selected **Check Scanner** properties:

Capability Name	Slip Entry was used	Top Entry was used
CapAutoGenerateFileID	False	False
CapAutoGenerateImageTagData	False	False
CapAutoSize	True	True
CapColor	CHK_CCL_MONO CHK_CCL_GRAYSCALE	CHK_CCL_MONO CHK_CCL_GRAYSCALE
CapConcurrentMICR	True	True – will be False next release
CapDefineCropArea	False	False
CapImageFormat – choice is set by firmware command in prelim release	CHK_CIF_TIFF CHK_CIF_NATIVE (= TIFF grayscale)	CHK_CIF_TIFF CHK_CIF_NATIVE (= TIFF grayscale)
CapImageTagData	False in prelim release	False in prelim release
CapMICRDevice	True	True – will be False next release
CapStoreImageFiles	False	False
CapValidationDevice	True	True



Direct I/O methods available:

1. Direct I/O: **SCN_ENTRY_POINT** selects from slip, or top (which sets false MICR and validation properties) or both (=operator's choice).
Command = **300**; in/out data values:
 - 1 = bottom slip entry
 - 2 = front top entry
 - 3 = either bottom slip or front top (default)
2. Direct I/O: **SCN_IMAGE_TRANSMIT_SIDE** selects from top image, bottom image, or both sides (dual pages in the same TIFF) to transmit.
Command = **301**; in/out data values:
 - 0 = both sides (default)
 - 1 = bottom only
 - 2 = top only

The definitions for **TPG_CHK_ENTRY_POINT** and **TPG_CHK_IMAGE_TRANSMIT_SIDE** are contained in the file **tpg.h** in the include files. Other DirectIO definitions are not yet implemented and should not be used.

6 Interpretations

This section contains descriptions of all areas where interpretation of the specification was necessary to implement an aspect of the OPOS standard.

6.1 Slip AutoPosition

If the registry value SlipAutoPosition = 1, the slip will be reverse fed as part of an EndInsertion method call. This will guarantee the slip starts at the same position each time, independent of front or side insertion.

6.2 Slip Status Event Firing

If *NCR_compatible* = 0 (registry value entry under the printer device unit sub-key), then the printer control will only fire a StatusUpdateEvent for a change in the slip status following the completion of a successful BeginInsertion method. The printer control will stop firing StatusUpdateEvent for slip status changes following the successful completion of an EndRemoval method. (The BeginInsertion and/or EndRemoval methods can be performed through the Printer and/or MICR control. The events are always fired by the printer control.)

If *NCR_compatible* = 1, then the printer control will fire a StatusUpdateEvent for a change in the slip status any time following the completion of a successful Claim method. The printer control will stop firing StatusUpdateEvent for slip status changes following the successful completion of a Release or Close method. The slip status changes can be the result of printer or MICR operation, but the printer control always fires the event.

6.3 Hardware Detection

The SO attempts to detect if hardware is present when setting the DeviceEnabled property to TRUE. Upon receiving a request to enable the device, the SO sends a status



request to the printer, and if no response is received in one second, the SO returns an error (OPOS_E_NOHARDWARE) to the application.

Because the MICR device and CashDrawer are integrated devices of the printers, hardware detection for these device classes uses the same procedure as the printer device. Upon receiving a request to enable the device, the SO sends a status request to the printer, and if no response is received in one second, the SO returns an error (OPOS_E_NOHARDWARE).

6.4 Receipt Paper Cuts

The A794 and A795 supports full cuts and fixed partial cuts. If the percentage specified in the *Percentage* parameter in the CutPaper method is 100%, the knife performs a full cut and returns OPOS_SUCCESS. Any value below 100% in the *Percentage* parameter of the CutPaper method will result in a successful partial cut. The numeric value in the one-shot cut commands that request a paper cut performs in a like fashion.

A call to ValidateData with any percentage value in a one-shot cut command returns success.

6.5 Barcode Printing

The slip station can print barcodes upright and upside down. If the user has set RotateSpecial = {PTR_RP_RIGHT90, PTR_RP_LEFT90}, and then inserted a slip via BeginInsertion/EndInsertion, any barcodes will be printed upright, and RotateSpecial will be reset by the SO to PTR_RP_NORMAL.

If the user has inserted a slip via BeginInsertion/EndInsertion, and then tries to set RotateSpecial to {PTR_RP_RIGHT90, PTR_RP_LEFT90}, RotateSpecial will not change and ResultCode will be set to OPOS_E_ILLEGAL.

6.5.1 Barcode Data Errors

The printer control will return an error (OPOS_E_ILLEGAL) if the user specifies a barcode that cannot be printed for any of the following reasons:

1. The barcode contains an invalid character for the given Symbology.
2. The barcode is missing a necessary character for the given Symbology.

The SO checks for barcode data errors. Legal characters for each symbology can be found in the Uniform Symbology Specification for the desired symbology and are briefly listed below. The SO returns an error (OPOS_E_ILLEGAL) to any data error in a PrintBarCode method call.

Symbology	Legal Characters
UPC-A	0-9 (ASCII)
UPC-E	0-9 (ASCII)
JAN 8	0-9 (ASCII)
JAN 13	0-9 (ASCII)



Symbology	Legal Characters
Codabar	0-9, -, \$, :, /, ., +, (plus start/stop characters A, B, C, D) (ASCII)
Interleaved 2 of 5	0-9 (ASCII)
Code 39	0-9, A-Z, , space, -, ., \$, /, +, %, (plus start/stop character *) (ASCII)
Code 128	Values 00-102 (plus start characters 103, 104, 105) Start code must be sent with data. Stop character will be added by printer.
PDF 417	All 256 ASCII and extended ASCII characters

UPC-A barcodes can be 11 or 12 digits long. The printer firmware adds the check digit if only 11 digits are sent. If the barcode is 12 digits long, the SO does not check the validity of the check digit.

UPC-E barcodes can be 6, 7 or 8 digits long compressed or 11 or 12 digits uncompressed. If 6 or 11 characters are sent, the printer firmware adds the check digit. In the other cases, it is assumed that the last number is the desired check digit; the SO does not check the validity of the check digit.

JAN8 barcodes can be 7 or 8 digits long. The printer firmware adds the check digit if only 7 characters are sent. If the barcode is 8 digits long, the SO does not check the validity of the check digit.

JAN13 barcodes can be 12 or 13 digits long. The printer firmware adds the check digit if only 12 characters are sent. If the barcode is 12 digits long, the SO does not check the validity of the check digit.

Codabar barcodes must start and stop with {A, B, C, or D}, and these characters cannot appear elsewhere in the barcode.

Interleaved 2 of 5 barcodes do not need an even number of characters. The SO will left pad a barcode with character 0 if there is an odd number of characters.

Code 39 barcodes do not need to specify the start and/or stop character '*'. It may be included, but if not, the firmware will add it/them automatically.

Code 128 barcodes must begin with a proper start code. If the last digit is not a valid check digit, the firmware will add it.

6.5.2 Barcode Heights

The printer firmware accepts a command with a byte parameter to set the barcode height. The units for the parameter are 1/152". The printer has a vertical resolution of 1/203", however. Therefore, if MapMode = PTR_MM_DOTS, the maximum value for the *Height* parameter of the PrintBarCode method is 341.

6.5.3 Barcode Widths

The printer supports four barcode module widths: 2, 3, 4 or 5 dots per module. The SO will interpret the *Width* parameter of the PrintBarCode method in the following fashion:



Width	Action
≤ 0	Error (OPOS_E_ILLEGAL) returned
1,2	Module width set to 2
3	Module width set to 3
4	Module width set to 4
5	Module width set to 5
> 5	Module width based on closest fit barcode

If the user specifies *Width* > 5 , then the SO picks the module width that results in a barcode that is as close as possible to the desired *Width*. This barcode may be bigger or smaller than *Width*.

If the user specifies a *Width* that is wider than the print zone, but the barcode fits on the page, then the barcode is printed and no error is returned to the user.

If the user chooses *Width* > 5 , then the printer control will return an error (OPOS_E_ILLEGAL) if the barcode cannot be printed because it is too wide to fit on the page at any module width.

If the user specifies a specific module width (*Width* = 2, 3, 4, 5) that causes the barcode to be wider than the print zone, then the control will return an error (OPOS_E_ILLEGAL).

The SO checks the width of the resultant barcode and returns an error (OPOS_E_ILLEGAL) if the barcode is wider than the print zone. The maximum number of barcode characters at each module width are listed below. For the variable width barcode symbologies {Codabar, Interleaved 2 of 5, Code 39, Code 128, PDF 417}, the values below are valid for the receipt station only. (No slip maxima are given here at this time.) For the Fixed length barcode symbologies {UPC-A, UPC-E, JAN8, JAN13}, the values below are valid for both the slip and receipt stations.

Symbology	Module Width	Maximum Number of Characters Printed
UPC-A	2, 3, 4, 5	11 + check digit – fixed number
UPC-E	2, 3, 4, 5	6 + check digit – fixed number
JAN8	2, 3, 4, 5	7 + check digit – fixed number
JAN13	2, 3, 4, 5	12 + check digit – fixed number
Codabar	2, 3, 4, 5	Determined by $M(10C + W + 1) \geq 448$ M = module width in dots (2, 3, 4) C = number of characters, inc. start and stop W = number of wide characters



Symbology	Module Width	Maximum Number of Characters Printed
Interleaved 2 of 5	2	40 (must be even number of digits)
	3	26 (must be even number of digits)
	4	18 (must be even number of digits)
	5	14 (must be even number of digits)
Code 39	2	20 + start + stop
	3	12 + start + stop
	4	9 + start + stop
	5	6 + start + stop
Code 128	2	23 + start + stop + check digit
	3	14 + start + stop + check digit
	4	9 + start + stop + check digit
	5	7 + start + stop + check digit
PDF 417	2, 3, 4	Seven characters print per line, with 58 rows maximum.

6.5.4 Code 128 Printing

All printers support Code 128 bar code printing. However, if the registry entries are changed as noted below, the functionality of the driver will be affected.

If *Code128* = 0 (registry value entry under the printer device unit sub-key), then the printer control will return an error (OPOS_E_ILLEGAL) when asked to print a Code 128 barcode.

If *Code128* = 1 (registry value entry under the printer device unit sub-key), then the printer control will attempt to print a Code 128 barcode.

The application must send the **value** of the character desired as designated by the Uniform Symbology Specification for Code 128 to the SO. The SO sends the data directly to the printer as it is received. For example, sending the string “g12345” to the printer will result in the Code 128A barcode “QRSTU” being printed. Likewise, the data “0x69 0x10 0x11 0x12 0x13 0x14” will result in the Code 128C barcode “1617181920”.

6.5.5 UPC-E Printing

The SO will accept uncompressed or compressed UPC-E barcode formats. The number of characters sent in the data field is based on the following schematic. **Important Note:** deviation from this schematic will result in invalid data.

Number of Digits	Description of Data
6	6 compressed UPC-E digits, NO leading 0, NO check digit
7	7 compressed UPC-E digits, NO leading 0, check digit included
8	8 compressed UPC-E digits, leading 0 included, check digit included
11	11 uncompressed UPC-E digits, leading 0 included, NO check digit
12	12 uncompressed UPC-E digits, leading 0 included, check digit included



6.5.6 PDF 417 Barcode Printing

The SO will adjust the dots per module setting to give the closest fit to the specified *Width*, but the *Height* parameter will be ignored. The user cannot vary the number of characters per row or any other parameter of the PDF 417 barcode.

6.6 Limitations on Bitmaps

The SetBitmap method is used to download a logo into printer memory. This logo is sent to the printer upon calling SetBitmap and stored in the printer's flash memory. The registry entry LoadBitmapsToFlash, which was available in previous versions of this driver, must always be 1.

The printer supports multiple downloaded bitmaps (up to 255) stored in flash memory. To provide access to this additional functionality, the SO has been extended so that the user application can set up to 255 bitmaps. To use this extension, the application should call SetBitmap with BitmapNumber = { 1 – 255 }. This will load the bitmap into the printer. To print the Bitmap, use the one-shot “ESC|#B” where # = { 1-255 } as previously set with SetBitmap.

The SO supports the downloading/printing of color bitmaps. Any *.BMP file that is monochrome (1 bit/pixel, or black and white), or is 4 or 8 bits/pixel can be printed on the printer. (The printer must be configured to print color bitmaps, which can be accomplished using the printer's Level 1 configuration menu.) The SO cannot handle 24 bit/pixel bitmaps, and will return OPOS_E_ILLEGAL to any method call to download/print them. To print color bitmaps, the user must set *CapRec2Color* to 1 in the registry.

The SO will convert any non-white and non-black pixel to the second color supported by the paper. To improve the printed appearance of color bitmaps you may use the LogoEZ™ utility to convert color bitmaps to black, white, and one other color. This utility can be obtained from www.tpgprinters.com or contact support@tpgprinters.com.

6.7 Character Set Selection

The SO defaults to the ANSI character set. No default is specified in the OPOS specification.

The SO supports character set 101. It is used as the user-defined character set, although there is no method for downloading such a set to the printer at the present time. This character set prints as Code Page 437.

If the ASCII character set is currently selected, then characters 0x80-0xFF print as null bytes, not space characters.

6.8 RecLineSpacing/RecLineSpacing

The OPOS property RecLineSpacing is configurable through the registry entry DotsPerLinespace under the Receipt sub-key. When the SO is started, it checks the registry for the value Receipt\DotsPerLinespace. If this value is present, it is used. If this value is not present, the SO checks for the registry entry CapRec2Color. If this value is not present, or is present and is not zero, the SO sets RecLineSpacing = 30. If CapRec2Color = 0 in the registry, the SO sets RecLineSpacing = 26. When



CapRec2Color = 0, the minimum value for RecLineSpacing is 24. When CapRec2Color = 1, the minimum value for RecLineSpacing is 27.

If WriteAllVariables = 1 in the registry, the SO will write any missing value/data pairs to the registry when DeviceEnabled is set TRUE. The SO will write CapRec2Color = 1 and DotsPerLinespace = 30 as defaults.

6.9 DotsPerPrinthead/RecLineHeight

The OPOS property RecLineHeight is configurable through the registry entry DotsPerPrinthead under the Receipt sub-key. When the SO is started, it checks the registry for the value Receipt\DotsPerPrinthead. If this value is present, it is used. If this value is not present, the SO checks for the registry entry CapRec2Color. If this value is not present, or is present and is not zero, the SO sets RecLineHeight = 27. If CapRec2Color = 0 in the registry, the SO sets RecLineHeight = 24.

If WriteAllVariables = 1 in the registry, the SO will write any missing value/data pairs to the registry when DeviceEnabled is set TRUE. The SO will write CapRec2Color = 1 and DotsPerPrinthead = 27 as defaults.

6.10 Rotated Printing

The printer supports the use of font attributes such as italics, bold, underline, reverse video, etc. while in rotated mode. As a result, the OPOS service object also supports the one-shots while in rotated print mode.

Blank lines may be fed by sending a Carriage Return/Line Feed or an empty PrintNormal statement to the printer.

For 80 mm paper, there is a maximum of 21 lines when printing in rotated mode to the receipt station of a color printer that has CapRec2Color = 1 in the registry, or does not have an entry for CapRec2Color in the registry. For a color printer that has CapRec2Color = 0 in the registry, there is a maximum of 23 lines when printing in rotated mode to the receipt station. Sending more than 21/23 lines to the receipt in rotated mode will result in the first 21/23 lines being printed, and OPOS_E_FAILURE returned for all remaining lines. OPOS_E_ILLEGAL will not be returned subsequently upon calling RotatePrint with *Rotation* = PTR_RP_NORMAL.

The maximum number of characters per line has been arbitrarily set to 88 for the receipt. This value is double the capacity of the given station in regular printing mode. The value can be set to any number by changing the registry entry SidewaysMaxChars under the receipt station sub-key to the desired value and restarting the SO.

There is a maximum of 33 lines when printing in rotated mode to the slip. Sending more than 33 lines to the slip in rotated mode will result in the first 33 lines being printed, and OPOS_E_FAILURE returned for all remaining lines. OPOS_E_ILLEGAL will not be returned subsequently upon calling RotatePrint with *Rotation* = PTR_RP_NORMAL.

The maximum number of characters per line has been arbitrarily set to 132 for the slip station. This value is double the capacity of the given station in regular printing mode. The value can be set to any number by changing the registry entry SidewaysMaxChars under the slip station sub-key to the desired value and restarting the SO.



When PrintImmediate is called after the application enters rotated print mode, the characters are currently printed in rotated **character** mode, not rotated line mode.

6.11 CapRec2Color

Unless CapRec2Color = 0 in the registry, the SO assumes the color printers are configured for two-color printing. See the sections on RecLineSpacing, RecLineHeight, and SidewaysMaxLines for default values of these OPOS properties in this case. To configure a printer SO for monochrome printing only, put CapRec2Color = 0 in the registry, and delete any existing value/data pairs for {DotsPerPrinthead, DotsPerLinespace, SidewaysMaxLines} under the Receipt sub-key.

6.12 Print Data Alignment

The printer SO does not support alignments other than left (PTR_BM_LEFT), center (PTR_BM_CENTER), and right (PTR_BM_RIGHT) for the PrintBarcode method. Any attempt to specify a different alignment in this method will cause the method to return an error (OPOS_E_ILLEGAL). Other methods that support alignments may be called with either the predefined alignments or an arbitrary alignment. The methods that support alignment specification are: PrintBarcode, PrintBitmap, and SetBitmap.

6.13 Changing the SlpLineSpacing Property

The property SlpLineSpacing has a threshold such that $0 \leq (\text{SlpLineSpacing} - \text{SlpLineHeight}) \leq 12$. Therefore, SlpLineSpacing has a valid range of {7..19} dots. If a value less than 7 dots is requested, the driver will default to 7 dots (which adds zero blank dot rows to the printed line). If a value greater than 19 dots is entered, the driver will default to 19. This property is affected by the value of MapMode. If MapMode is any value other than PTR_MM_DOTS, the correct values will vary according to the appropriate conversion. Because of rounding error, some adjustments may be required to achieve desired SlpLineSpacing.

The property SlpLineHeight is fixed to 7. It cannot be changed. Attempting to change it will result in ResultCode = OPOS_E_ILLEGAL, and the SlpLineSpacing and SlpLinesNearEndToEnd will not be updated.

6.14 Receipt Printing with a Slip Inserted

If the application has performed a BeginInsertion on the printer slip station or MICR device, then any attempt to print to the receipt will return an error (OPOS_E_ILLEGAL) and no data will be printed. Once a BeginInsertion has been sent, data can be sent to the receipt only after a successful EndRemoval method is sent through the printer or MICR device.

6.15 ValidateData Implementation

There is a "one shot" command that specifies the amount to feed the receipt/slip. There is a numeric parameter in this command that can be specified in the currently selected MapMode units. The ValidateData method is supposed to return an error (OPOS_E_ILLEGAL) if the value specified is not exactly attainable, but a close fit exists. TPG always returns success to this command.



6.16 CashDrawer Open

Some Cash Drawers return a logical high signal for open and a logical low signal for closed while others return the opposite. To make it easy to correct for these differences, there is a registry entry (CashDrawer_Open) for CashDrawer status. The SO must be restarted after changing this parameter.

6.17 StatusUpdateEvents

The StatusUpdateEvent PTR_SUE_REC_NEAREMPTY is fired when the StatusUpdateEvent PTR_SUE_REC_EMPTY is fired. The “near empty” event can be fired on its own, too, when the printer senses the receipt roll is almost exhausted, but if it has not been fired and the SO detects that paper is exhausted, it fires both events.

6.18 Setting the DataEventEnabled Property

The DataEventEnabled property in the MICR control may be set to TRUE at any time, even if the MICR device is not enabled. This has no adverse effect on the drivers and is actually beneficial, since some automatic responses (such as the ErrorEvent Event) aren’t returned until DataEventEnabled is set to TRUE.

7 TPG Additions

This section contains all additions TPG has made to the driver outside the letter of the OPOS standard.

7.1 Multiple Downloaded Bitmaps Using SetBitmap

As described earlier, TPG has extended the SetBitmap method to make available up to 255 individual downloaded bitmaps.

7.2 Check Flip

Some models of the hybrid 2 station printers are capable of flipping a check following a MICR read. This lets the user perform a MICR read, then print the validation, and then flip the check and print its face before returning it to the customer for signature. Following a flip operation, the user must reverse feed the check so the top edge is at a desired spot to start rotated printing on the face.

7.3 Extended Error Codes for Improved Error Handling

According to the OPOS specification, when the ResultCode property is not OPOS_E_EXTENDED, the ResultCodeExtended property may be set with proprietary values for more descriptive error reporting. The following ResultCodeExtended values may be returned when the Result Code for a particular operation is OPOS_E_FAILURE or OPOS_E_NOSERVICE.

Command Human Readable Interpretation	ResultCode	Definition of ResultCodeExtended
Returned in ErrorEvents		
TPG_BADREAD_DATA	OPOS_E_FAILURE	MICR read with data missing



Command Human Readable Interpretation	ResultCode	Definition of ResultCodeExtended
TPG_NOCHECK_NODATA	OPOS_E_FAILURE	No document in slip station, no data read
TPG_PAPERJAM_NODATA	OPOS_E_FAILURE	Check jammed, do data preserved
TPG_NOCHARS_NODATA	OPOS_E_FAILURE	No MICR encoded characters on form
TPG_KNIFE_ERROR	OPOS_E_FAILURE	An error has occurred with the knife
TPG_KNIFE_SLIP_JAM	OPOS_E_FAILURE	Knife or Slip jam has occurred
TPG_SLIP_MOTOR_JAM	OPOS_E_FAILURE	Slip motor has experienced a jam
TPG_FATAL_ERROR	OPOS_E_FAILURE	A fatal error has occurred
TPG_UNKNOWN_ERROR	OPOS_E_FAILURE	An unknown error has occurred
Returned in Open Method		
TPG_REGQUERY_COM_ERROR	OPOS_E_FAILURE	Error with COM port while checking settings
TPG_REG_QUERY_ERROR	OPOS_E_FAILURE	An error has occurred while retrieving values from the registry
TPG_INVALID_PORT	OPOS_E_NOSERVICE	COM PORT specified is invalid
TPG_INVALID_BAUD_RATE	OPOS_E_NOSERVICE	Baudrate requested is not supported
TPG_INVALID_PARITY	OPOS_E_NOSERVICE	Parity requested is not supported
TPG_INVALID_PROTOCOL	OPOS_E_NOSERVICE	Requested Handshaking protocol is not supported
TPG_RECEIVEBUFFER_ERROR	OPOS_E_NOSERVICE	An error has occurred with the receive buffer
Returned for Flash or RAM Writes		
TPG_FLASH_LOGOAREA_FULL	OPOS_E_FAILURE	Flash memory is full, erase and retry
TPG_FLASH_CHARAREA_FULL	OPOS_E_FAILURE	Flash memory is full, erase and retry
TPG_RAM_LOGO_AREA_FULL	OPOS_E_FAILURE	RAM logo area is full

7.4 DirectIO

TPG has added a number of DirectIO implementations to increase the versatility and functionality of the TPG OPOS driver. All additions are enumerated in the TPG.H file (including Human Readable Interpretations) for ease of use in host applications.



The DirectIO additions fall into four categories:

- Functional Commands
- Parameter Commands
- Flash memory Management Commands
- Remote Diagnostics Commands

Command Human Readable Interpretation	Command Description
Functional Commands	
TPG_EXECUTE_CHECK_FLIP <i>pData</i> is not used. <i>pString</i> is not used.	Send Check Flip command to Printer. This command must be preceded by successful calls to BeginInsertion and EndInsertion.
Parameter Commands	
TPG_DOWNLOADED_BITIMAGE_DENSITY <i>pData</i> must be: 0 – As is (normal dimensions) 1 – Double wide 2 – Double high 3 – Double high and wide (quadruple size) <i>pString</i> is not used.	Change DPI of the Print Downloaded Bit Image command (used in conjunction with SetBitmap and the “ESC B” one-shot. This command overrides the registry value of DownLoadedBitImageDensity
CODE128_CHECK_INCLUDED <i>pData</i> parameter must be 0 or 1. <i>pString</i> is not used.	Change the default behavior of the SO to calculate check digit for Code 128 barcodes (if the last character is the same value of the check digit) or not. This command overrides the registry value of Code128CheckIncl
TPG_MICR_EXCEPTION <i>pData</i> is not used. <i>pString</i> encoded string defining an exception table entry	Load exception tables for check field parsing
Flash Management Commands	
TPG_ERASE_CHARLOGO_FLASH <i>PData</i> and <i>pString</i> are not used.	Erase 64k flash memory page used for user logos and character sets
TPG_ERASE_USERDATA_FLASH <i>PData</i> and <i>pString</i> are not used.	Erase 64k flash memory page used for user data storage



Command Human Readable Interpretation	Command Description
TPG_LOAD_CHARS_TO_RAM <i>pString</i> is the path and name of the file that contains the user-defined character set command sequences. <i>pData</i> is not used.	Download a user-defined character set definition file to printer RAM
TPG_LOAD_CHARS_TO_FLASH <i>pString</i> is the path and name of the file that contains the user-defined character set command sequences. <i>pData</i> is not used.	Download a user-defined character set definition file to printer flash memory
TPG_WRITE_TO_USERDATA The lowest byte of <i>pData</i> is the number of bytes to write and the high 3 bytes are the beginning address—presently only 2 bytes used. <i>pString</i> is the data to write.	This command corresponds to the TPG command Write to User Data Storage – { 0x1b 0x27 m addr data }
TPG_READ_FROM_USERDATA The lowest byte of <i>pData</i> is the number of bytes to write and the high 3 bytes are for the beginning address—presently only 2 bytes used. <i>pString</i> is the data to write.	This command corresponds to the TPG command Read From User Data Storage – { 0x1b 0x34 m addr data }
TPG_DEFINE_WATERMARK <i>pString</i> is used to specify standard MS BMP full file name, watermark shading percentage, input logo index, output logo index, and desired watermark width in current MapMode units. <i>pString</i> must be formatted as: “full logo file name, _shading percentage, _input index, _output index, _desired water mark width”, where '_' is a space character. <i>pData</i> is not used, and must be set to 0. For instance, <i>pString</i> ="C:\My Pictures\white.bmp, 60, 1, 6, 450" <i>pData</i> =0	This command works with TPG_TOGGLE_WATERMARK command to provide a watermark.



Command Human Readable Interpretation	Command Description
TPG_TOGGLE_WATERMARK <i>pString</i> is used to specify the index of the loaded and shaded watermark logo in the printer's flash memory, and the skip distance between two prints in 8x printer dot rows. <i>pString</i> must be formatted as: "output index,_skip number", where '_' is a space character. <i>pData</i> is not used and must be set to 0. User can turn off the watermark by setting the skip number to zero. For instance, <i>pString</i> = "6, 3" <i>pData</i> = 0	This command works with TPG_DEFINE_WATERMARK command to turn on or off a watermark.
TRANSMIT_FLASH_STATUS <i>pData</i> and <i>pString</i> are not used.	Return the result of the last flash memory action from the firmware. Corresponds to the TPG firmware command 0x1d 0x72 0x04.
Remote Diagnostics Commands	
TPG_REMOTE_DIAGNOSTICS The value of <i>pData</i> is taken from the rows that follow in this table. <i>pString</i> will be set according to the requirements of the desired diagnostic as well.	Execute the corresponding Remote Diagnostic command. These commands call the TPG Remote Diagnostics commands.
Remote Diagnostics (<i>pData</i>)	
SERIALNO_WRITETO_NVRAM <i>pString</i> must contain exactly 10 bytes of alphanumerical data	Change the serial number
SERIALNO_TONVRAM_RCPTVER <i>pString</i> must contain exactly 10 bytes of alphanumerical data	Change the serial number and print it to the receipt for verification
SERIALNO_RETURN <i>pString</i> is not used.	Return serial number from the printer
CLASS_MODELNO_WRITETO_NVRAM <i>pString</i> must contain exactly 15 bytes of alphanumerical data. If the Class/Model number being written is less than 15 characters, the end of the field must be padded with spaces.	Change the class/model number



Command Human Readable Interpretation	Command Description
CLASS_MODELNO_TONVRAM_RCPTVER <i>pString</i> must contain exactly 15 bytes of alphanumeric data. If the Class/Model number being written is less than 15 characters, the end of the field must be padded with spaces.	Change the class/model number
CLASS_MODELNO_RETURN <i>pString</i> is not used.	Return class/model number from printer.
BOOT_FIRMWARE_PARTNO_RETURN <i>pString</i> is not used.	Return boot firmware part number from printer.
BOOT_FIRMWARE_CRC_RETURN <i>pString</i> is not used.	Return boot firmware CRC from printer.
FLASH_FIRMWARE_PARTNO_RETURN <i>pString</i> is not used.	Return flash memory firmware part number from printer.
FLASH_FIRMWARE_CRC_RETURN <i>pString</i> is not used.	Return flash memory firmware CRC from printer.
RECLINES_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Change the receipt lines tally
RECLINES_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Change the receipt lines tally and print it to the receipt for verification
RECLINES_TALLY_CLEAR <i>pString</i> is not used.	Clear the receipt lines tally
RECLINES_TALLY_RETURN <i>pString</i> is not used.	Return the receipt lines tally
KNIFECUT_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Change the knife cut tally
KNIFECUT_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Change the knife cut tally and print it to the receipt for verification
KNIFECUT_TALLY_CLEAR <i>pString</i> is not used.	Clear the knife cut tally
KNIFECUT_TALLY_RETURN <i>pString</i> is not used.	Return the knife cut tally
SLIPCHAR_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Command to change the Slip Character Tally
SLIPCHAR_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Command to change the Slip Character Tally and print it to the receipt for verification
SLIPCHAR_TALLY_CLEAR <i>pString</i> is not used.	Clear the Slip Characters Tally
SLIPCHAR_TALLY_RETURN <i>pString</i> is not used.	Return the Slip Characters Tally
MICRREAD_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Command to change the MICR Read Tally



Command Human Readable Interpretation	Command Description
MICRREAD_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Command to change the MICR Read Tally and print it to the receipt for verification
MICRREAD_TALLY_CLEAR <i>pString</i> is not used.	Clear the MICR Read Tally
MICRREAD_TALLY_RETURN <i>pString</i> is not used.	Return the MICR Read Tally
HOURS_ON_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Change the hours on tally
HOURS_ON_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Change the hours on tally and print it to the receipt for verification
HOURS_ON_TALLY_CLEAR <i>pString</i> is not used.	Clear the hours on tally
HOURS_ON_TALLY_RETURN <i>pString</i> is not used.	Return the hours on tally
BOOT_FIRMWARE_VERSION_RETURN <i>pString</i> is not used.	Return the boot firmware version string
FLASH_FIRMWARE_VERSION_RETURN <i>pString</i> is not used.	Return the flash memory firmware version string
FLASHCYCLES_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Change the flash memory cycles tally
FLASHCYCLES_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Change the flash memory cycles tally and print it to the receipt for verification
FLASHCYCLES_TALLY_CLEAR <i>pString</i> is not used.	Clear the flash memory cycles tally
FLASHCYCLES_TALLY_RETURN <i>pString</i> is not used.	Return the flash memory cycles tally
KNIFEJAMS_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Change the knife jams tally
KNIFEJAMS_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Change the knife jams tally and print it to the receipt for verification
KNIFEJAMS_TALLY_CLEAR <i>pString</i> is not used.	Clear the knife jams tally
KNIFEJAMS_TALLY_RETURN <i>pString</i> is not used.	Return the knife jams tally
COVEROPENINGS_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Change the cover openings tally
COVEROPENINGS_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Change the cover openings tally and print it to the receipt for verification



Command Human Readable Interpretation	Command Description
COVEROPENINGS_TALLY_CLEAR <i>pString</i> is not used.	Clear the cover openings tally
COVEROPENINGS_TALLY_RETURN <i>pString</i> is not used.	Return the cover openings tally
MAX_TEMPERATURE_TALLY_RETURN <i>pString</i> is not used.	Return the maximum temperature tally
SLIP_LINES_TALLY_WRITETO_NVRAM <i>pString</i> must contain exactly 8 digits.	Command to change the Slip Lines tally
SLIP_LINES_TALLY_TONVRAM_RCPTVER <i>pString</i> must contain exactly 8 digits.	Command to change the Slip Lines tally and print it to the receipt for verification
SLIP_LINES_TALLY_CLEAR <i>pString</i> is not used.	Clear the Slip Lines Tally
SLIP_LINES_TALLY_RETURN <i>pString</i> is not used.	Return the Slip Lines Tally

Please refer to the Users Guide for complete descriptions of the individual remote diagnostic commands. For the commands that return data to the application the *pString* will contain the data returned, so the calling application must ensure that the memory space pointed to is large enough to hold the maximum possible amount of data.

7.5 MICR Exception Table Support

Some banks print the check serial number in a location which cannot be electronically distinguished without bank-specific exception information. For these cases, the specific bank is picked out by its transit number, which is a predefined MICR field. The SO looks in an exception table for a transit number match and uses a mask associated with the transit number to locate the serial number. If there is no transit number match in the table, the SO attempts to locate the serial number in default locations.

7.6 Exception Table Entry

The MICR control supports a DirectIO method to let the application download an exception table entry into the SO. This method may be used to load several entries into the SO. Once entries are loaded, they cannot be removed without terminating the SO. The entries must be reloaded every time the SO is started.

The DirectIO method takes three parameters:

Command – Command number, which is 1000 for the MICR exceptions. (See below for a human readable mnemonic for this value.)

pData – NULL

pString – A string containing the nine characters of the transit field, followed by a space, followed by a mask string containing the characters { 'a', 's', 'x' }.

The SO saves the transit field string and the mask string to a table of exceptions. The SO compares the transit field of every MICR read to every transit field string stored in the

exception table. If no matches are found, then the SO follows the default parsing rules to locate the serial number and the account number.

If there is a match in the transit field, then the digits following the second transit symbol in the MICR data are parsed according to the mask string associated with the correct transit field string.

a = account number digit
s = serial number digit
x = ignore this digit

Some account fields have spaces or dashes in them. With this exception table the user controls whether or not the space or dash appears in the parsed account number. For instance, if the account number was “245 48967” and you wanted the space to appear in the parsed account number, the mask would be “aaaaaaaa” but if you did not want the space to appear in the parsed account number the mask would be “aaaxaaaaa”.

7.7 Examples

In the examples, the following notation is used:

t = MICR transit symbol
o = MICR on us symbol

Example 1

t1234567890t1234 9876543210o 1234 is the check serial number
9876543210 is the account number

```
pString = "1234567890 ssssxaaaaaaaaaax"
```

Example 2

t22137-632t001 6042202o927540 2754 is the check serial number
6042202 is the account number

```
pString = "22137-632 xxxxxaaaaaaaaxxssssx"
```

7.8 Application Header File

TPG provides a file (TPG.H) that contains definitions of certain constants for DirectIO and ResultCodeExtended as described earlier in this document so that C/C++ applications can refer to certain commands and errors using a human readable mnemonic.

TPG also provides a similar file (TPG.BAS) that contains the same information in the appropriate format for use by Visual BASIC developers.



8 Appendices

8.1 Appendix A -- Setting a default code page

Certain legacy application programs, when started, assume a particular code page setting. However, when a different printer is installed, a different default code page may be used and characters previously available are now unavailable. So that the application need not be rebuilt to add code to select a different code page, a registry variable, **DefaultCodePage**, is introduced. This configures the OPOS driver to use any code page available on the printer and known to the driver as default for a printer.

8.2 Usage and behavior

DefaultCodePage is a variable taking a long value, one of known code pages, such as 437, 858 and others. The value selected must be available in the printer; this can be determined by examining the diagnostics document, produced by resetting the printer while pressing the feed button. The variable must also be available in the OPOS driver for the printer in question. The values for each printer are documented following. It is possible to remove **DefaultCodePage** from the registry; in this case, the printer uses the default that it would normally. If **DefaultCodePage** is set to an incorrect value, likewise, the default setting is used.

If the printer is not printing as expected, be sure that you have entered **DefaultCodePage** in the right location in the registry (see following), that the value is legal for the printer, that the code page is available on the printer and that your spelling is correct.

8.3 Example Registry Values

A7xx can be one of A793, A794 or any other known model number.

HKLM\Software\OLEforRetail\ServiceOPOS\POSPrinter\A7xx\DefaultCodePage REG_SZ 437 (US)
HKLM\Software\OLEforRetail\ServiceOPOS\POSPrinter\A7xx\DefaultCodePage REG_SZ 333 (Illegal; forces use of default)
HKLM\Software\OLEforRetail\ServiceOPOS\POSPrinter\A7xx\DefaultCodePage REG_SZ 999 (Windows, also known as 1252)
HKLM\Software\OLEforRetail\ServiceOPOS\POSPrinter\A7xx\DefaultCodePage REG_SZ 737 (Greek; this forces default on, for instance, an A793)
HKLM\Software\OLEforRetail\ServiceOPOS\POSPrinter\A7xx\<not present> (Default is used)

Code Pages Supported by the OPOS Driver

Bold italic denotes the OPOS driver default code page for the printer.

Code page 1252 is equivalent to code page 999.

A793, A756-xxxx:	437, 850 , 858, 998, 101
A721:	437, 850 , 852, 865, 998, 101
A794, A758, A760, A776, A795:	437, 737, 850, 852, 858, 860, 862, 863, 865, 866, 998, 999 , 101, 1252



8.4 A794 A715 CP850 Emulation

A794 printers support a feature known as *A715 CP850 Emulation*. This, essentially, interchanges the assignments of CP 850 and CP 858. For the OPOS driver to operate correctly, the *A715 CP850 Emulation* feature must be **disabled**. To do so, use the Printer Config Menu under Emulation/Software Options.

8.5 Setting DefaultCodePage in an A7xx.reg file

Each driver installation package contains a file A7xx.reg, specific to the printer model, which, when used with the installation script, sets registry variables. Each such file contains a setting for **DefaultCodePage** identical to the driver default code page. This variable definition is provided as a convenience; if it is missing the driver will still start with the default code page. Change the variable setting to start the driver with a different code page.



8.6 Appendix B -- Code 128 Barcode Parsing

ASCII parsing for Code 128 barcodes, introduced in UPOS version 1.8, has been added to the TPG OPOS 1.5 driver. See the UPOS specification¹, p703. The implementation follows the UPOS document with a few points clarified. Two methods of parsing data sent to the driver are described, along with the necessary registry settings. If no registry values are set, then each version of driver behaves as documented in the UPOS specification, with PTR_BCS_Code128 being unparsed.

{A, {B, {C are specified as code shift sequences. Each code set (A, B or C) can shift to either of the other two; for example, A can shift to B or C. The specification says nothing about start codes (103-105), but at least one application uses the {C sequence to place a start code. If one of the mentioned sequences occurs at the beginning of data, a start code is inserted, otherwise, an appropriate shift code.

Because of the overloading mentioned above, plus the fact that the UPOS specification doesn't forbid doing so, the usual start code characters (103-105) are also recognized as start characters in this implementation.

If the sentinel character ('{') is followed by a character other than those specified, or occurs as the last character in the input, an ILLEGAL error is returned. This helps detect application errors, since it is unlikely that the application would intend using the sentinel in this manner.

Since this is added to a V1.5 driver, the symbol PTR_BCS_Code128_Parsed is unavailable to applications using the V1.5 symbol definitions. Registry variables set under the "...\\POSPrinter\\Axxx" key control driver parsing behavior. See the description of variable behaviors later in the document. Before installing the driver, change the Axxx.reg file included in the distribution. The registry value type is REG_SZ.

8.7 Format of data presented to the OPOS driver (method 1)

Code 128 bar symbols are mapped to values 0-105. There is also a symbol used to end a barcode, corresponding to no value. In turn, depending on the code set chosen, reading a barcode symbol will return the value, which then is used in one of three ways, depending upon the code set in effect. For Code sets A and B, if the value is from 0-63 (decimal), this is mapped to ASCII values 32-95, displaying the characters starting at <space> and ending with <underscore>. If the value is between 64 and 95, how it is mapped depends on the code set. If Code A is selected, these are treated as nonprintable ASCII characters 0-31 (NUL to US). If Code B, then characters <backquote> through DEL result. This includes the lower case alphabet. Values above 95 have special meanings in Codes A and B.

¹UnifiedPOS Retail Peripheral Architecture Version 1.8 June 30, 2003
This is available from <http://www.monroecs.com>, among other places.



Code C maps barcode values 0-99 decimal to pairs of decimal digits 00-99. This allows, for example, a sequence of 10 digits to be encoded in 5 barcode symbols. By convention, if a sequence has an odd number of digits, a 0 is placed at the beginning; for example 12345 becomes 012345. The Code 128 barcode specification is maintained by ANSI. It is sold by AIM Global (www.aimglobal.org; search using X5-4.) Many other websites have Code 128 information, as well.

For presenting a string of ASCII data to the driver (excluding use of binary conversion) the following convention is followed:

Code A: For bar code values between 0 and 63, printable characters <space> through <underscore> are used. The driver subtracts 32 to obtain the correct encoding. For the unprintable characters 64-95, the Code B printable symbols (lowercase ASCII letters) are used, and again, 32 subtracted. When printed, the HRI characters for values 64-95 are shown as dots, not as characters. When scanned, they perform the correct functions, such as bell or tab.

Code B: The Code B code set is used, <space> through DEL are presented and again 32 is subtracted, yielding values in the correct range.

Code C: Only decimal digits, FNC1 ({1}) and Code shifts ({A}, {B}) are allowed. Each pair of digits is combined into a single value, which is then passed to the barcode printer. For example, in sequence, upper to lower, '4' and '5' produce the value $4 \times 10 + 5 = 45$, which is then passed to the printer. If the sequence of digits contains an odd number of characters, a '0' digit is placed at the beginning of the sequence.

8.8 Full ASCII method (Method 2)

A second configuration is available for ASCII presentation to the driver. The method described earlier in the document, for Code A and Code B, uses only Code B printable characters (<space> through DEL) as input. Any character presented in the range of NUL to US (0-31) is illegal.

The full ASCII method for Code A allows characters in the range 0-95 to be input; for Code B, the range is 32-127. For either code set, any character outside the range is illegal. When a `SHIFT` ({S}) is encountered, the following character must be in the range of the other code set. Code C treats `SHIFT` codes as illegal.

8.9 The DEL Character

In Code B, characters 0-94 have a printable representation and can be entered as ASCII text. Character 95, the DEL character has no such representation. We extend the UPOS standard for use of the sentinel representation and use {D} to enter DEL.

TABLE OF VALUES INSERTED

	Code A	Code B	Code C
{A	*	Code A (101)	Code A (101)
{B	Code B (100)	*	Code B (100)
{C	Code C (99)	Code C (99)	*
{D	DEL (95) ^{2,3}	DEL (95)	*
{1	FNC1 (102)	FNC1 (102)	FNC1 (102)
{2	FNC2 (97)	FNC2 (97)	*
{3	FNC3 (96)	FNC3 (96)	*
{4	FNC4 (101)	FNC4 (100)	*
{S	SHIFT (98)	SHIFT (98)	*
{{	Left Brace (91)	Left Brace (91)	*

1. The first column is the character pair ({x) received.
2. The remaining columns are the values inserted when the specified code set is effective.
3. Each entry consists of the name (e.g. FNC1) and the value inserted (102).
4. An asterisk means that the character pair is illegal in the code set selected. The driver will return status of E_ILLEGAL.
5. If {A, {B or {C occurs as the first sequence in the data, the appropriate value STARTA (103), STARTB (104) or STARTC (105) is inserted.
6. STARTx may occur directly at the start of the data, rather than {x.
7. Anything other than a type of start code sequence ({x or STARTx) at the start of data causes the driver to return E_ILLEGAL.

² DEL is in Code B; when using Method 1, it can be used for data entry when in Code A.

³ When using Method 2, in Code A, this must be preceded by a SHIFT code.



8.10 Migration to UPOS version 1.8 and Beyond-Use of Registry Entries

Although Code 128 parsing is a feature of UPOS Version 1.8 and above, it has been added to the TPG Version 1.5 Opos driver, with some restrictions and capabilities not otherwise specified. In order to make this as smooth an implementation as possible, and to help applications when migrating to use of the driver, some registry entries controlling the driver configuration are added.

All driver values are of type REG_SZ and are interpreted as int32 values. All settings apply only to symbologies **PTR_BCS_Code128** and **PTR_BCS_Code128_Parsed**.

8.11 Code128Parse

For driver versions 1.5-1.7, the lowest two bits of the int32 value are meaningful and apply to **PTR_BCS_Code128**:

Bits 0 and 1 clear	No parsing
Bit 0 set and bit 1 clear	Use method 1 for parsing
Bit 0 clear and bit 1 set	Use method 2 for parsing
Both bits set	No parsing.

If parsing is selected, it is always enabled for **PTR_BCS_Code128**. If **Code128Parse** is not present, then no parsing is performed.

For driver versions 1.8 and above the above settings still apply to **PTR_BCS_Code128**.

8.12 Code128App18

This applies to driver versions 1.5-1.7. If present with data of 1, the driver will recognize **PTR_BCS_Code128_Parsed** although the symbol is undefined in Opos versions under 1.8. This is used by version 1.8 applications that make requests through the 1.5 drivers for this style of barcode. **Code128Parse** must be present and selecting a parsing method, either 1 or 2.

8.13 Code128Parse18

This is meaningless for driver versions below 1.8. For 1.8 and above, this applies only to **PTR_BCS_Code128_Parsed**.

Bit 0 clear	Use method 1
Bit 0 set	Use method 2

If **Code128Parse18** is missing, then assume method 1.



Printing Barcodes from the MS POS application

The OPOS specification for OPOS versions 1.8 and later states that printer Service Objects (SO) versions 1.7 and earlier can accept barcode data in any format, but that, starting with OPOS version 1.8 an ASCII parsing encoding is supported, which, if followed, should allow application programs to use different Service Objects more readily.

The Microsoft Point of Sale application (MSPOS), using another printer, passes to PrintBarCode, a Code 128 data string with no start code and characters encoded as ASCII. This does not adhere to the OPOS parsing convention mentioned above, but assuming an older (pre 1.8) OPOS driver, is permitted. An example of the data seen by the PrintBarCode method (four decimal bytes): 42 49 49 42. Note that there is no Code128 start character (one of 103, 104 or 105, or, properly using the ASCII parsing encoding, 123 65 42 49 49 42.) By checking the barcode printed by the competitive printer, the intent is to print the barcode for “*11*” using Codeset A.

To print the same barcode using the TPG driver the registry must be set thus:

Code128	1	Can print Code128
Code128DefaultStart	103	Codeset A if Start code missing
	104	Codeset B if Start code missing
	105	Codeset C if Start code missing
Code128Parse	1	if an ASCII representation used
	0	if Code128 data is sent

To illustrate setting **Code128Parse** for printing the barcode “*123*”, assuming that **Code128DefaultStart** is 103 (Codeset A.):

If the string sent is “42 49 50 51 42”, then setting **Code128Parse** to 1 prints a Code A barcode as desired. (ASCII data representation.)

If the string sent is “10 17 18 19 10”, then setting **Code128Parse** to 0 again prints the correct barcode (Code128 data representation.)

Other **Code128xxxx** registry settings may be set as appropriate.



8.14 Appendix C -- Changes to the CashDrawer Service Object

Three data variables have been added to the cash drawer registry for all OPOS CashDrawer service objects. The values, along with those for the pulse on and off times, are dependent on the brand of cash drawer used and should be adjusted accordingly.

8.15 PollInterval

This specifies in milliseconds the time between status polls to the cash drawer. Default is 50 ms. Shortening this interval causes the CPU to spend more time polling. Lengthening the time increases the likelihood that the driver will miss a drawer transition.

8.16 StatusDelay

When an OpenDrawer operation is executed, the driver pauses for StatusDelay milliseconds and then polls the drawer to obtain an initial drawer state. If the drawer is locked or held shut, the signal indicating that the drawer is open is quite short. This interval should be set slightly longer than the length of the longest short drawer open interval. This is determined by testing the signals with an oscilloscope or some other method. For the HP cash drawer, the interval is determined to be 130 ms; the default value for StatusDelay is 150 ms.

8.17 DisableImmediatePoll

If nonzero, this disables the immediate poll following the OpenDrawer operation. Default is 0.

Legacy variables are **PulseOnTime** and **PulseOffTime**. These, again, depend on the specific make/model of cash drawer used. Consult the cash drawer specification. Default values for both are 65, which is doubled; therefore with the default setting, the pulse on is 130 ms and the off pulse is also 130 ms.

8.18 CashDrawer_Open

Indicates the sense of the signal required to open the cash drawer. This varies with the make of cash drawer used. Possible values are LOW and HIGH. Default is HIGH.

Notice of Cash Drawer Registry usage. March 15, 2007

In the registry files (A7xx.reg) provided with the TPG OPOS Service Objects (SO), the values **PollInterval**, **StatusDelay**, **DisableImmediatePoll**, **PulseOnTime**, **PulseOffTime** and **CashDrawer_Open** are furnished to control various polling parameters and the electrical pulse on/off time required to kick the drawer open. The values provided at this time are an empirical choice by TPG engineering based on observed operation under usage on the HP rp5000 POS station, running Microsoft Dynamics Windows POS application version 2.0. The values will differ depending on the cash drawer used, the application and possibly other factors. TPG cannot be responsible for the values, which must follow the specification of the drawer used.

Defaults for **PollInterval**, **PulseOnTime** and **PulseOffTime** are, respectively, 50 milliseconds, 65 ms, and 65 ms. See the document CashDrawerChanges.xxx for more information.



8.19 Appendix D -- Print Commands for Printing 90° Rotated

Excerpt from the UnifiedPOS Retail Peripheral Architecture Version 1.9 January 16, 2005 documentation for printNormal:

Line Feed (10)	Print any data in the line buffer, and feed to the next print line. (A Carriage Return is not required in order to cause the line to be printed.)
Carriage Return (13)	<p>If a Carriage Return immediately precedes a Line Feed, or if the line buffer is empty, then it is ignored.</p> <p>Otherwise, the line buffer is printed and the printer does not feed to the next print line. On some printers, print without feed may be directly supported. On others, a print may always feed to the next line, in which case the Service will print the line buffer and perform a reverse line feed if supported. If the printer does not support either of these features, then Carriage Return acts like a Line Feed.</p> <p>The validateData method may be used to determine whether a Carriage Return without Line Feed is possible, and whether a reverse line feed is required to support it.</p>

Considerations for the TPG Service Object (SO):

For printing 90° rotated, the TPG SO recognizes four sequences: Carriage Return (CR), Line Feed (LF), CRLF and LFCR. ***Each of these has the same effect: the sideways line, if any, is terminated and the print position moves to the next sideways print line.*** Note that while printing sideways, none of these controls affects the physical print mechanism; the lines are buffered until the sideways print is terminated.

The CR semantics for sideways print is to act like an LF, advancing the line, as mentioned above.

The LFCR sequence is not mentioned by the UPOS documentation except by default; that is the LF semantics followed by the CR semantics. Conceivably, this could result in an advance of 2 lines rather than 1. The TPG SO treats this as a single line advance.



8.20 Appendix E –Registry Reference

This section explains the registry entries under each device unit sub-key. The reference is incomplete; other variables are explained in sections preceding this. In particular, we recommend against changing the registry unless explicitly permitted/recommended to do so (see section 3.1 and appendix C.)

8.21 CashDrawer

@ – Default device driver registry reference.

Description – Text description of the OPOS SO.

DeviceDescription – Text description of this device.

Position – CashDrawer position. Allowed values are {1, 2}.

Uses – The POSPrinter SO device unit sub-key which describes the printer to which the CashDrawer is attached.

ConnectedPrinterType – Identifies the printer. See the description of PrinterType below for allowed values.

CashDrawer_Open – Change return values so that DrawerOpened property returns the correct value for the status of the drawer. Allowed values are {LOW, HIGH}. Default = "HIGH".

PulseOnTime – Time in ms that the CashDrawer solenoid is energized to open the drawer. Default = 65 ms.

PulseOffTime – Time in ms that the CashDrawer solenoid is turned off after energizing to prevent overheating of the solenoid. Default = 65 ms.

Trace – Sub-key that contains the following driver debug parameters.

Level – 0 = Do not print trace log information to a file, 1 = Print trace log information to a file. Default = 0.

FileName – File name to save trace log information.

FileLenMax – Maximum size in KB of the trace log file. If this value is removed, tracing is disabled.

8.22 MICR (Does not apply to the A794 or A795)

@ – Default device driver registry reference.

Service – Text description of the path and filename of the driver executable.

Description – Text description of the OPOS SO.

DeviceDescription – Text description of this device.

CmdInitiateMicr – Binary key containing the hexadecimal command sequence to initiate a MICR read at the printer. Default = 1b 77 01.

CmdOpenSlipJaws – Binary key containing the hexadecimal command sequence to open the slip jaws. Default = 18.

SlpPaperStatusBits – Decimal value that is used as a mask with the status request (1b 76) response to determine when a sip is present. Default value = 96.

AllowPartialData – 0 = Do not return partial data when there is an unsuccessful MICR read which might contain some valid data; 1 = Return partial data when there is an unsuccessful MICR read which might contain some valid data. Default = 0.
[Character read errors will be indicated by a '?' in the RawData property.]

MICRType – Identifies the printer that contains this MICR device. See the description of PrinterType in the POSPrinter registry section below. Default = 12.



Uses –The POSPrinter SO device unit sub-key which describes the printer to which the MICR reader is attached.

Trace – Sub-key that contains the following driver debug parameters.

Level – 0 = Do not print trace log information to a file, 1 = Print trace log information to a file. Default = 0.

FileName – File name to save trace log information.

FileLenMax – Maximum size in KB of the trace log file. If this value is removed, tracing is disabled.

8.23 POSPrinter

The following registry entries are REQUIRED to successfully connect to the SO.

@ – Default device driver registry reference.

Port – COM port. Allowed values are {COM1, ..., COM8}. Default = COM1. For use with printers supporting the TPG USB interface, the allowed value is {USB} Note that the Protocol entry must also be set to USB in this case.

Baudrate – Allowed values are {9600, 19200, and 115200}. Default = 1152000.

Parity – Allowed values are {NONE, ODD, EVEN}. Default = NONE.

Protocol – For RS-232. Allowed values are {DTR, XON}. Default = XON. For use with printers supporting the TPG USB interface, the allowed value is {USB}. Note that the Port entry must also be set to USB in this case.

The following registry entries are REQUIRED by the SO to configure properly. If the values do not exist in the registry, the SO will write defaults. These defaults must be changed to properly reflect your printer's physical capabilities and firmware configuration in order to operate correctly for the connected device.

ReceiptLineChars – Default receipt width in characters. Allowed value is {44}. Default = 44.

SlipLineChars – Default slip width in characters. Allowed value is {66}. Default = 66.

ReceiveBufferLength – An internal parameter used by the SO. Allowed value is {128}. Default = 128.

PrinterType – Stores specific type of printer connected. Default = 9.

PrinterType	Printer Model
0	TPG A793
1	TPG A756-4xxx
4	TPG A756-8xxx
5	TPG A756-3xxx
6	TPG A794
7	TPG A721
9	TPG A758
11	TPG A795
12	TPG A760
43	TPG A776

The following registry entries are written by the SO to their default value. They may be changed to customize the functionality of the SO.



WaitDoc – This parameter is not used. Default = 1.

WaitDocStart – This parameter is not used. Default = 1.

WatchTime – Timer in ms for asynchronous printer status check. Allowed values are {0...2000000}. If the value is 0, there is no asynchronous printer check. Default = 2000 ms.

SlipAutoPosition – 0 = Do not position the slip at all after successful BeginInsertion/EndInsertion calls; 1 = Move the slip to a position where the top of the first row of characters will be 1.0 mm from the top of the slip after a successful BeginInsertion/EndInsertion calls. Default = 0

SlipBothSides – 0 = Printer will not print on both sides of an inserted form. 1 = Printer supports printing on both sides of an inserted form. Default = 0.

NCR_compatible – 0 = Do not fire slip StatusUpdateEvents before calling BeginInsertion; 1 = Fire slip StatusUpdateEvents at any time, to emulate the NCR OPOS driver.

CommTimeout – Maximum time in ms to wait for a response from the printer for general communication. Default = 10000 ms.

RetryTimeout – Maximum time in ms to wait for a response from the printer before ceasing attempts to resend data to printer. Default = 1000 ms.

flowOffTimeout – The maximum time to wait to receive a FlowOn after a FlowOff. This timeout is used to differentiate handshaking delays from error conditions. Default = 4000 ms.

flowBlocksize – Size of data blocks to send to printer. Default = 64 bytes.

CommReadTotalTimeoutConstant – The maximum amount of time in ms to wait to receive a read response from the printer. Default = 1000 ms.

CommWriteTotalTimeoutConstant – The maximum amount of time in ms to wait to receive a response from a write to the printer. Default = 1000 ms.

WriteAllVariables – 0 = Do not write ALL possible registry entries to the registry for diagnostic/configuration purposes; 1 = Write ALL possible registry entries to the registry with the value they are set to as default. This operation will not replace values that are already in the registry (with the exception of the descriptive entries). Default = 0.

CapPowerReporting – 0 = Printer does not have the ability to report power state changes; 1 = Printer does have the ability to report power state changes. Default = 1.

CapRecNearEndSensor – 0 = Printer does not have a receipt paper low sensing capability; 1 = Printer has receipt paper low sensing capability. Default = 1.

FeedBeforeCut – 0 = Printer does not feed the last line of the receipt past the knife before the cut as part of the CutPaper method; 1 = Printer feeds the last line of the receipt past the knife before the cut as part of the CutPaper method. Default = 1.

CapRec2Color – 0 = Printer cannot print in any color but black; 1 = Printer can print in a color other than black. Default = 1.

TPGPaperCategoryM – This is the *m* parameter of the printer command to select a paper type. This type is used to configure the printer's algorithms to optimize print output for the selected paper. A table is provided below, but refer to the *printer's User Guide* for all allowed selections of this parameter. *This parameter is not currently used.* Default = 0.



TPGPaperFormVersionN – This is the *n* parameter of the printer command to select a paper type. This type is used to configure the printer's algorithms to optimize print output for the selected paper. A table is provided below, but refer to the *printer's User Guide* for all allowed selections of this parameter. *This parameter is not currently used.* Default = 0.

Paper Type	TPGPaperCategoryM	TPGPaperFormVersionN
Monochrome Paper	0	0
Kanzaki Red/Black (P-310RB)	1	0
Kanzaki Red/Black (P-320RB)	5	0
Mitsubishi Blue/Black (PB770)	5	0

The following registry entries are descriptive in nature and are written by the SO.

Service – Text description of the path and filename of the driver executable.

Description – Text description of the OPOS SO.

Version – Text description of the version number of the OPOS SO.

DeviceDescription – Text description of this device.

The following registry entries are not written unless the value of WriteAllVariables is 1. These values may be entered into the registry to further customize the SO for the specific device attached.

Code128 – 0 = Printer cannot print Code 128 barcodes; 1 = Printer can print Code 128 barcodes. Default = 1.

Code93 – 0 = Printer cannot print Code 93 barcodes; 1 = Printer can print Code 93 barcodes. Default = 0.

PDF417 – 0 = Printer cannot print PDF 417 barcodes; 1 = Printer can print PDF 417 barcodes. Default = 1.

EAN128 – 0 = Printer cannot print EAN 128 barcodes; 1 = Printer can print EAN 128 barcodes. Default = 1.

CapCoverSensor – 0 = Printer does not have a cover sensor; 1 = Printer does have a cover sensor. Default = 1.

CapRecPapercut – 0 = Printer does not have a knife; 1 = Printer has a knife. Default = 1.

RealTimeCommands – Determines how the SO performs error recovery during asynchronous printing. Default = 3. This is a bit-wise operator.

Bit	ON
0	GS EOT 2, GS EOT 4, GS EOT 5 sequences are used to retrieve printer status instead of ESC v
1	GS EXT 2 sequence is used in ClearOutput method to clear printer buffer
2	Send buffered command 0x10 to clear stubborn data from print buffers



- SlipPaperStatusBits** – A mask used to determine the slip status from the printer status response. Default = 96.
- CharacterSetList** – The list of available character sets in the printer. Corresponds to the CharacterSetList property of POSPrinter.
- RecLineCharsList** – The list of available receipt station character pitches in characters per line. This is determined by the value of ReceiptLineChars in the registry.
- SlpLineCharsList** – The list of available slip station character pitches in characters per line. This is determined by the value of SlipLineChars in the registry.
- CmdOpenSlipJaws** – Specifies the command to open the slip jaws. The value(s) should be entered in hexadecimal format. Default = 0x12.
- DownloadedBitImageDensity** – The value n , needed to specify the density of the Print Downloaded Bit Image command GS / n . Default for is 0. Allowed values are {0, 1, 2, 3} 0 = as is; 1 = double wide; 2 = double high; 3 = double high & double wide.
- CmdSelectSlipPaper** – Specifies the command to select the slip station for printing. The value(s) should be entered in hexadecimal format. Default = 0x11, 0x1c”.
- Trace** – Sub-key that contains the following driver debug parameters.
- Level** – 0 = Do not print trace log information to a file; 1 = Print trace log information to a file. Default = 0.
 - FileName** – File name to save trace log information.
 - FileLenMax** – Maximum size, in KB of the trace log file. If this value is removed, tracing is disabled.

The following parameters will be written to the registry when the value of WriteAllVariables is 1 except as noted.

- Receipt** – Sub-key that contains parameters needed to perform special tasks on the receipt station. The SO may create separate entries for each station even if the device does not have a particular station.
- DotsHoriz** – Number of dots in one horizontal dot row for given station. Allowed values are {576, 640}. Default = 576.
- DownloadByteLimit** – Maximum size for a downloaded bit image. Default = 64000.
- DownloadVerticalDotLimit** – Maximum vertical height in dots of a downloaded bit image. Default = 1000.
- SidewaysMaxChars** – The maximum number of characters allowed on one line while printing in Rotated 90° left or right. Default = 88.
- SidewaysMaxLines** – The maximum number of lines printable while printing in Rotate 90° left or right. If CapRec2Color = 1, the allowed values are {1-21}. If CapRec2Color = 0, the allowed values are {1-23}. Default = 21.
- DotsPerPrinthead** – The number of vertical dots in the physical print mechanism. If CapRec2Color = 1, the allowed value is {27}. If CapRec2Color = 0, the allowed value is {24}. Default = 27.
- BitImageMode** – The value for m (density), of an image printed using the command $ESC * m n1 n2 d1...dn$. Allowed values are {0, 1, 32, 33}. Default = 33.
- DotHeight_Mm** – The height of 1 dot in millimeters. Default = 0.125 mm.
- DotWidth_Mm** – The width of 1 dot in millimeters. Default = 0.125 mm.



DotHeight1000th_inch – The height of 1 dot in thousandths of an inch. Default is 4.9213 thousandths of an inch.

DotWidth1000th_inch – The width of 1 dot in thousandths of an inch. Default = 4.9213 thousandths of an inch.

DotsPerLineSpace – The height of one print line in dots, including extra dot rows between print lines. Allowed values are {27-127}. Default = 30.

DotsPerCharHeight – The height of the character cell. Allowed value is {27}. Default = 27.

BasicPrinterUnitsPerDot – A parameter used to translate different MapMode values into units expected by the printer firmware. Allowed value is {1.99607}. Default = 1.99607.

The following registry entries are written by the SO to their default value. They may be changed to customize the functionality of the SO.

PrintTimeChar – The amount of time in milliseconds it takes the station to print one character. Used to help determine a timeout before checking for error at printer. Default = 0.

PrintTimeNewline – The amount of time in milliseconds it takes the station to print one line feed. Used to help determine a timeout before checking for error at printer. Default = 0.

The following registry entries are not written unless the value of WriteAllVariables is 1. These values may be entered into the registry to further customize the SO for the specific device attached.

Compressed – Sub-key that contains parameters for printing in compressed mode.

SidewaysLineFeedUnits – Number of dot rows to feed to simulate a line feed when printing in a 90° rotated state. Default = 10.

Normal – Sub-key that contains parameters for printing in normal mode.

SidewaysLineFeedUnits – Number of dot rows to feed to simulate a line feed when printing in a 90° rotated state. Default = 13.

Slip – Sub-key that contains parameters needed to perform special tasks on the slip station. The SO may create separate entries for each station even if the device does not have a particular station.

DotsHoriz – Maximum number of dots in one horizontal dot row for given station. Allowed values are {330}. Default = 330.

SidewaysMaxChars – The maximum number of characters allowed on one line while printing in Rotated 90° left or right. Default = 132.

SidewaysMaxLines – The maximum number of lines printable while printing in Rotate 90° left or right. Allowed values are {1-33}. Default = 33

DotsPerPrinthead – The number of vertical dots in the physical print mechanism. Allowed value is {8}. Default = 8.

BitImageMode – The value for *m* (density), of an image printed using the command ESC * *m* n1 n2 d1...dn. Allowed values are {0, 1, 32, 33}. Default = 0.

DotHeight_Mm – The height of 1 dot in millimeters. Default = 0.35278 mm.

DotWidth_Mm – The width of 1 dot in millimeters. Default = 0.36576 mm.



DotHeight1000th_inch – The height of 1 dot in thousandths of an inch. Default = 13.9 thousandths of an inch.

DotWidth1000th_inch – The width of 1 dot in thousandths of an inch. Default = 14.4 thousandths of an inch.

DotsPerLineSpace – The height in dots, of one print line including extra dot rows. Default = 10.

DotsPerCharHeight – The height of the character cell in dots. Default = 7.

BasicPrinterUnitsPerDot – A parameter used to translate different MapMode values into units expected by the printer firmware. Allowed value is {2}. Default = 2.

The following registry entries are written by the SO to their default value. They may be changed to customize the functionality of the SO.

PrintTimeChar – The amount of time in milliseconds it takes the station to print one character. Used to help determine a timeout before checking for error at printer. Default = 0.

PrintTimeNewline – The amount of time in milliseconds it takes the station to print one line feed. Used to help determine a timeout before checking for error at printer. Default = 0.

The following registry entries are not written unless the value of WriteAllVariables is 1. These values may be entered into the registry to further customize the SO for the specific device attached.

Compressed – Sub-key that contains parameters for printing in compressed mode.

No entries are used for this station.

Normal – Sub-key that contains parameters for printing in normal mode.

No entries are used for this station.

Note: the SO may write other keys/values/data that are not listed here to the registry. These items should not be changed, as they may affect operation of the software.