PRINTRONIX AUTO ID

SDK Reference Manual

Thermal Series Printers

Trademark Acknowledgements

Printronix and PSA are registered trademarks of Printronix, Inc.

PXML, T8000, T6000, T4000, and T800 are trademarks of Printronix, Inc. or Printronix Auto ID Technology, Inc.

Table of Contents

1 *Overview*

Introduction

This manual describes an SDK which provides an Application Programming Interface (API) via a Dynamic Linked Library (DLL). The DLL can be easily imported into both C++ and C# applications. The API is structured on the PTX-SETUP and PGL protocols, both of which are proprietary Printronix Auto ID languages. The user application can make function calls provided by the DLL which generates PGL and PTX-SETUP commands to be sent to the printer.

When the user builds forms or generates data to be sent to the printer, the memory space to hold that information is allocated automatically in the DLL. This relieves the user application from memory allocation concerns. Instead, streams of PGL commands are consolidated and optimized inside the DLL.

The SDK API supports the following basic PGL commands:

- Printer reset – PTX_SETUP
- Get printer status – STATUS
- Create Form – CREATE
- Execute Form – EXECUTE
- Paper Setup – PAPER
- Draw Box – BOX
- Draw Lines – DIAG, HORZ, VERT
- Print Ascii Text – ALPHA
- Print Asian Text – TWOBYTE
- Barcode – BARCODE
- RFID Programming – RFWTAG, RFRTAG
- Print Image – LOGO

Additional printer configuration and programming functionality can be made available at a future date if they become necessary. For now, this set of functionalities is provided for the benefit of the developer (easy to understand) and DLL support.

Requirements

The DLL has been compiled and tested with Windows 7 and Windows 8 platforms, both 32 and 64-bit systems. The user must choose the tools, coding environment, and programming language (C# or C++) in which to use the DLL.

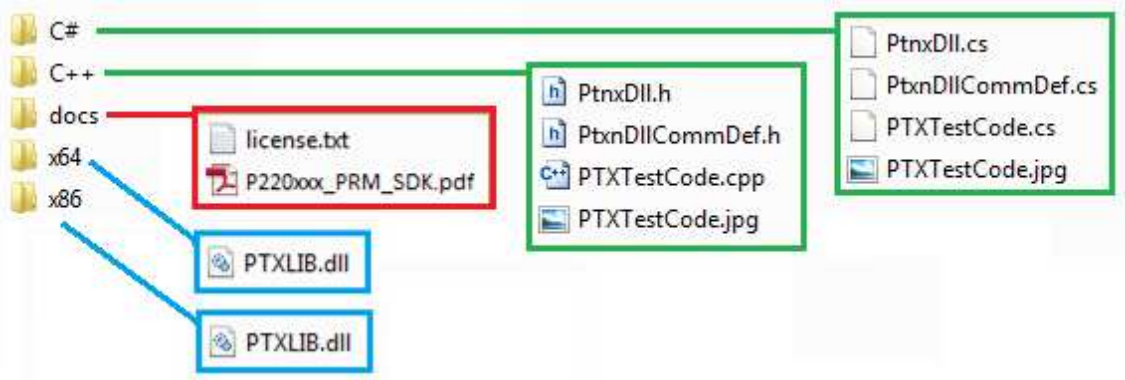
License

This SDK is provided under the MIT open source license which gives the developer freedom in using this code in any other open source or commercial application.

2 SDK Content

Unpacking the SDK

The SDK is a compressed zip file that can be downloaded from the www.PrintronixAutoID.com website which when unzipped is comprised of a few directories and important content:



The full path to the SDK can be found at this URL:

https://printronixautoid.com/support/downloads/printcart-printnet-enterprise-sv-series-download/

C# Files

The C# directory includes the definition files PtnxDll.cs and PtnxDllCommDef.cs as well as a demo version that uses the DLL with PTXTestCode.cs. The demo PTXTestCode.cs can be used to illustrate several functionalities, including creating and printing a label as shown in **PTXTestCode.jpg**.

C++ Files

The C++ directory includes the header files PtnxDll.h and PtnxDllCommDef.h as well as a demo version that uses the DLL with PTXTestCode.c. The demo PTXTestCode.c can be used to illustrate several functionalities, including creating and printing a label as shown in **PTXTestCode.jpg**.

Definition Files

Printronix provides two header or definition files. The header files should be included in the application source files used within any application that references the DLL.

- **`PtnxDll.h (C++) and PtnxDll.cs (C#)`**

  It imports all the API function calls within the DLL under the namespace or class **PTXLIB**. These files provide the prototypes of the functionalities available to your application.

- **`PtnxDllCommDef.h (C++) and PtnxDllCommDef.cs (C#)`**

  These files define various constants used by APIs for input parameters and return status. These are the definition of interface parameters between API and user's application.

Documentation

Documentation in the “docs” directory includes the SDK Programmer’s Manual (this document) plus the MIT license information in **license.txt**.

DLL: 32-bit and 64-bit Systems

The DLL is named PTXLIB.dll and there are two versions. The version in the x86 directory is for 32-bit applications and x64 is for 64-bit applications.

Linking the DLL

The proper DLL file must be linked into the program into order to be accessed. As a default method, you can place the chosen *.dll file in the same path as the executable. However, there are other options as described for the Microsoft Visual Studio environment at Creating and Using a DLL.

For C++, the header file **`PtnxDll.h`** imports all the API function calls within the DLL under the namespace **PTXLIB** and must be included within your source application. See the example in PTXTestCode.cpp in C++ directory.

Similarly for C#, the **`PtnxDll.cs`** imports all the API function calls within the DLL under the class **PTX_DLL** and must be included within your source application. See the example in PTXTestCode.cs in C# directory.

3 *Printer Setup and Design*

Printer Setup

PTX-SETUP

The Printronix Auto ID thermal printers are setup to process PTX-SETUP by default. This set of printer commands is parsed and executed regardless of active IGP emulation (PGL, ZGL, IGL, etc). In order to use PTX-SETUP, the following operator panel menus must be set:

- **Application > Control > PTX Setup Parse** = 21h
- **Application > Control > PTX Setup SFCC** = Enable

PGL for Graphics

The Printronix Auto ID thermal printers are setup to process PGL thermal emulation for the graphics commands. The following operator panel menus must be set as follows:

- **Application > Control > Active IGP Emul** = PGL
- **Application > PGL Setup > Select SFCC** = 126

PGL Printer Status

To receive status information from the printer from the function `PTXA_GetStatus()`, the operator menus should be set as follows:

- **Application > PGL Setup > Preparser** = Enable
- **System > Printer Mgmt > Ret. Status Port** = Automatic*

For the menu Ret. Status Port, the setting “Automatic” is the default and means that return status will be sent from the same port in which data is received. The developer can choose other settings (e.g., “Serial” would always return data from the serial port even if USB was chosen to send data).

Alternatives for Graphics

The DLL graphic functions require PGL to be the active IGP emulation. However, in the case the user wants to send data from another thermal printing language, this can be accomplished by changing the operator panel menu **Application > Control > Active IGP Emul** to the desired language and then using the function `sendcommand()` to send the raw data.

Alternatives for Printer Status

If the user wants a different thermal printing language to be active, they can request status from the printer using the sendcommand() function commands for the desired language or use PTX-SETUP as follows:

```
sendCommand( "!PTX_SETUP\r\nCONFIG-INFO;ETH\r\nPTX_END\r\n" );
```

This command is documented in the product Administrator’s Manual. The parameter “ETH” is used to tell the printer to respond over ethernet. The data response from the printer would look something like this:

```
!PTX_SETUP
CONFIG-INFO
```

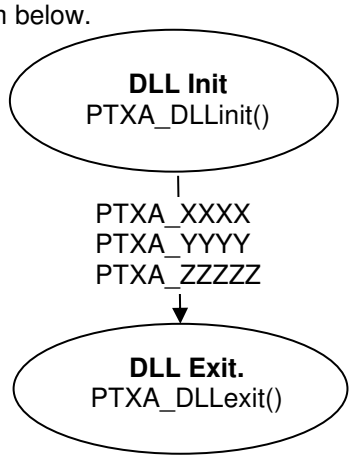
```
MODEL;T6204
PROGRAM VERSION;V2.15B
PROGRAM FILE;P360987
SERIAL NUMBER;SN000843231
DPI;203
RFID;1
ODV;0
GPIO;0
PTX_END
```

After issuing this PTX_SETUP command to the printer, there is a function `PTXA_Read()` from the DLL that can be used to receive the data returned from the printer.

Design Theory and Flow

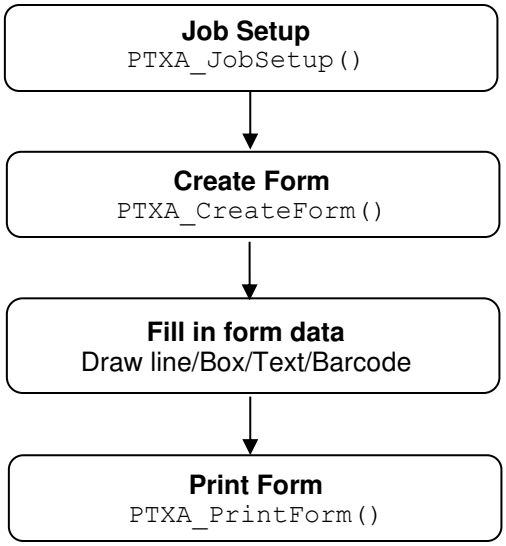
Initialize and Exit DLL

Before using the DLL, it must be initialized with memory allocated. This is accomplished with the PTXA_DLLinit() function. Likewise, when the DLL is no longer needed, memory should be freed with the PTXA_DLLexit() function. Commands and functions to create and print labels would go in between these function calls as shown in the diagram below.



Print Job Flowchart

The below diagram describes the DLL flow to create a form, there is no form data returned to the caller during all those steps until the last function `PTXA_PrintForm()` is called. All the form data is managed inside DLL.



Access PGL Data Stream

When the PGL form is generated and printed by `PTXA_PrintForm()`, the whole PGL form is stored inside DLL. The caller can choose to access the data sent to the printer using the return value of `PTXA_PrintForm()` as shown in the example below. Perhaps the user wants to archive the data or change a single string value and feels this is the easiest way to generate another form.

Example in C++:

```
char *pBuffer, *p;

/* open printer port controlled by user's application */
.....................

/* Create form */
PTXA_XXXX( );
PTXA_YYYY( );
PTXA_ZZZZ( );

/* General PGL label form */
p = PTXA_PrintForm(lForm);

/* Optional, copy the PGL form to caller's own buffer memory */
pBuffer = malloc( strlen(p) );
memcpy( pBuffer, p, strlen(p) );
```

Alternatively, the user can choose to retrieve the form data stored in the DLL using the API `PTXA_GetFormData()` which provides the data block by block. It copies the desired length of form data to user's memory buffer for each call. For example, the length is set to 1K bytes, the first call of this API will copy the first 1K bytes data to user's buffer; the second call will copy the next 1K bytes data to user's buffer; this is continued until the form data is empty.

Restoring after Reset

To reset the printer, user can call `PTXA_PrinterReset()`, this API will return the previous configuration PGL commands to the caller, caller can then send the data to printer or re-store the data in own memory buffer in user's application.

Example in C++:

```
char buffer[100], *p;

/* open printer port controlled by user's application */
.....................

/* Do the reset and save the form data in progress */
p = PTXA_PrinterReset();

/* optional: restore to caller's memory buffer*/
memcpy( buffer, p, strlen(p) );
```

Important

The function `PTXA_PrinterReset()` reboots the printer. The user's application would have to pause several minutes before issuing any commands to printer while the printer is rebooting.

4 *Detailed Interface*

This section will provide all the detailed interface functions, parameters, and descriptions.

Job setup and form creation is described in Section “Print Job Flowchart”. All printer data is stored inside the DLL until requested by the application. When the application is ready to generate print data, it can use the function `PTXA_PrintForm` (Section ”Form Printing – PTXA_PrintForm”) to retrieve the PGL data. Upon request, then DLL will supply buffers of data to the application which can then be sent to the printer through whatever host the application chooses. When the DLL finally returns an empty buffer, there is no more data to send.

The DLL will perform error checking on parameters according to this document. In many functions, a return value will signify the function parameters were correct. The DLL will not validate the PGL parameters when creating the form, this will be done when execute the PGL form on printer side.

Connect / Disconnect with Printer

Open Port – openport

Function	(int) openport(char * interface)	
Purpose	This API will start the printer spool.	
Input Parameter	**Range / Type**	**Description**
char * interface	For local printer, please specify the printer driver name. Like: “**TTP-244 Plus**”	
	For network printer, please specify the UNC path and printer name. Like: “**\\server\TTP243**”	
	For centronics interface directly, please specify LPT1 to LPT4. Like: “**LPT1**”	
	For USB interface directly, please specify USB. Like: “**USB**”	
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Open Ethernet – openethernet

Function	(int) openethernet(char * ipaddress, int portnumber)	
Purpose	This API will open an ethernet connection.	
Input Parameter	**Range / Type**	**Description**
char * ipaddress	Use IP address format such as “192.168.2.33”	
int portnumber	Port number for connection	
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Close Port – closeport

Function	(int) closeport(void)	
Purpose	This API will close the printer spool.	
Input Parameter	**Range / Type**	**Description**
None		
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

DLL Control

DLL Initialization – PTXA_DLLinit

Function	(void) PTXA_DLLinit(void)	
Purpose	This API will initialize DLL. This is the first API to be called by user’s application before use any other APIs.	
Input Parameter	**Range / Type**	**Description**
None		
Output Parameter	**Range / Type**	**Description**
Return Value	None	

DLL Exit – PTXA_DLLexit

Function	(void) PTXA_DLLexit(void)	
Purpose	This API will free all the allocated memory inside DLL. This is the last API to be called by user’s application before end of DLL calls.	
Input Parameter	**Range / Type**	**Description**
None		
Output Parameter	**Range / Type**	**Description**
Return Value	None	

DLL version - PTXA_GetDLLVersion

Function	(char *) PTXA_ GetDLLVersion(void)	
Purpose	This API returns the PTNX DLL version.	
Input Parameter	**Range / Type**	**Description**
None		
Output Parameter	**Range / Type**	**Description**
Return Value	String	Pointer of the buffer which holds DLL version number. The following message stored in the buffer: *Ptnx DLL version x.x* *Date: mm/dd/yyyy*

Printer Setting

Printer Reset - PTXA_PrinterReset

Function	(char *) PTXA_PrinterReset(void)	
Purpose	This API will reset the printer, use PTX_SETUP command.	
Input Parameter	**Range / Type**	**Description**
None		
Output Parameter	**Range / Type**	**Description**
Return Value	String	This API will provide a pointer into memory where the reset command is stored. The reset command is terminated with a NULL ('\0') character.

Generated Code

```
!PTX_SETUP
CONFIG-RESET
PTX_END
```

Printer Status - PTXA_SetupStatus

Function	(char *) PTXA_SetupStatus (int mode)	
Purpose	This API will enable or disable the snooper.	
Input Parameter	**Range / Type**	**Description**
int mode	PTXA_SnoopSer	Enable the snooper for serial ports
	PTXA_SnoopPar	Enable the snooper for parallel ports
	PTXA_SnoopEth	Enable the snooper for ethernet ports
	PTXA_SnoopOff	Disable the snooper
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Printer Status - PTXA_GetStatus

Function	(char *) PTXA_GetStatus(void)	
Purpose	This API returns PGL STATUS command which user can send the result to the printer to inform the printer send back the printer status to user's application via the port they have opened.	
Input Parameter	**Range / Type**	**Description**
None		
Output Parameter	**Range / Type**	**Description**
Return Value	String	Pointer of the buffer which holds the PGL commands. The following PGL commands stored in the buffer: *~STATUS*

Generated Code

~STATUS

NOTE

- Menus as described in Section "PGL Printer Status" must be set properly.
- Refer to PGL Programmer's Reference Manual for returned data from the ~STATUS command.

Creating and Executing Forms

This section lists down the APIs which used to creating form and set the corresponding printer parameters.

Form Setup

Printer Setup - PTXA_JobSetup

Function	int PTXA_JobSetup (int mode, int value)	
Purpose	This API will set the printer attributes one by one, using PGL PAPER command in NORMAL mode. The value definition follows PGL PAPER command standard.	
Input Parameter	**Range / Type**	**Description**
int mode	Constants from PtnxDllCommDef.h described in Table 1 below.	
int value	Constants from PtnxDllCommDef.h described in Table 1 below.	
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Table 1 Mode and Value for `PTXA_JobSetup` Function

mode	**units**	***value***	**Comments**
PTXA_JobModeReset	n/a	n/a	Reset mode parameter will not link any mode parameter for the form
PTXA_JobModeWidth	0.0001”	100 - 8500	Maximum width based on model
PTXA_JobModeLength	0.0001”	100 - 99000	Range is 0.1” – 99”
PTXA_JobModeSpeed	IPS	1 - 14	Maximum IPS varies by model
PTXA_JobModeIntensity	n/a	-15 to 15	
PTXA_JobModeLabel	PTXA_JobLabelNoSensor PTXA_JobLabelMark PTXA_JobLabelGap PTXA_JobLabelAdvGap PTXA_JobLabelAdvNotch		Not all modes are available, see PAPER command from PGL Programmer’s Reference Manual
PTXA_JobModeMedia	PTXA_JobMediaCon PTXA_JobMediaTearoffStrip PTXA_JobMediaTearoff PTXA_JobMediaPeeloff PTXA_JobMediaCut		Not all modes are available, see PAPER command from PGL Programmer’s Reference Manual
PTXA_JobModeRotate	Degrees	0, 90, 180, 270	

Generated Code Samples

API Call	PGL Code Generated
PTXA_JobSetup(PTXA_JobModeReset, *value*)	**n/a**
PTXA_JobSetup(PTXA_JobModeWidth, *value*)	~PAPER;SIZE *value*,0
PTXA_JobSetup(PTXA_JobModeLength, *value*)	~PAPER;SIZE 0,*value*
PTXA_JobSetup(PTXA_JobModeSpeed, *value*)	~PAPER;SPEED IPS *value*
PTXA_JobSetup(PTXA_JobModeIntensity, *value*)	~PAPER;INTENSITY *value*
PTXA_JobSetup(PTXA_JobModeLabel, *value*)	~PAPER;LABELS *value*
PTXA_JobSetup(PTXA_JobModeMedia, *value*)	~PAPER;MEDIA *value*
PTXA_JobSetup(PTXA_JobModeRotate, *value*)	~PAPER;ROTATE *value*

Create Form - PTXA_CreateForm

Function	int PTXA_ CreateForm (char * formname)	
Purpose	Request to create form, use PGL CREATE command.	
Input Parameter	**Range / Type**	**Description**
char *formname	String	Name of the form created
Output Parameter	**Range / Type**	**Description**
Return Value	0-1000	Form that is created. Save this value and use when creating graphics within the form.

Generated Code

~CREATE;formname

Form Setup - PTXA_FormSetup

Function	int PTXA_ FormSetup (int mode, int value)	
Purpose	This API sets up the parameter for the form.	
Input Parameter	**Range / Type**	**Description**
int mode	PTXA_FormSetupLen: Length of form in IGP dots or 1/72” PTXA_FormSetupCount: Number of times to execute form	
int value	PTXA_FormSetupLen: 0 – 65536 PTXA_FormSetupCount: 1 – 999	
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Generated Code

When mode= PTXA_FormSetupLen: *~CREATE;formname;formlength*

When mode= PTXA_FormSetupCount: *~EXECUTE;formname;count*

Position Scale – PTXA_Scale

Function	int PTXA_Scale (int hdpi, int vdpi)	
Purpose	Define the vertical line spacing and the horizontal pitch of the form for data positioning specified by dot row and column, use PGL SCALE;DOT command. The default scale is character row/column, when this API is set, the form scale will be changed to dot row/column.	
Input Parameter	**Range / Type**	**Description**
int hdpi		Horizontal resolution, default is 60dpi
int vdpi		Vertical resolution, default is 72dpi
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Generated Code

```
SCALE;DOT;hdpi;vdpi
```

Text, Barcodes, Graphics

Draw Box - PTXA_DrawBox

Function	int PTXA_DrawBox (int lForm, int thickness, int SR, int SC, int ER, int EC)	
Purpose	Draw box, use PGL BOX command	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to be printed
int thickness	1 – form length	Line thickness, enter a value of 1 or greater.
int SR	0 – form length	Starting row of the box. This parameter can be dot row or character row based on scale setting. Does not support the CP.DP format. The default is character row.
int SC	0 – form width	Starting column of the box. This parameter can be dot column or character column based on scale setting. Does not support the CP.DP format. The default is character column.
int ER	0 – form length	Ending row of the box. This parameter can be dot row or character row based on scale setting. Does not support the CP.DP format. The default is character row.
int EC	0 – form width	Ending column of the box. This parameter can be dot column or character column based on scale setting. Does not support the CP.DP format. The default is character column.
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Generated Code

BOX
thickness;SR;SC;ER;EC
STOP

Draw Line - PTXA_DrawLine

Function	int PTXA_DrawLine (int lForm, int thickness, int SR, int SC, int ER, int EC)	
Purpose	Draw line, based on the parameters SR,SC,ER,EC, it will use PGL DIAG/HORZ/VERT command accordingly determined inside DLL.	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to be printed
int thickness	1 – form length	Line thickness, enter a value of 1 or greater.
int SR	0 – form length	Starting row of the line. This parameter can be dot row or character row based on scale setting. Does not support the CP.DP format. The default is character row.
int SC	0 – form width	Starting column of the line. This parameter can be dot column or character column based on scale setting. Does not support the CP.DP format. The default is character column.
int ER	0 – form length	Ending row of the line. This parameter can be dot row or character row based on scale setting. Does not support the CP.DP format. The default is character row.
int EC	0 – form width	Ending column of the line. This parameter can be dot column or character column based on scale setting. Does not support the CP.DP format. The default is character column.
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed
NOTE		
1. When SR<>ER and SC<>EC, it will use DIAG 2. When SR==ER, it will use HORZ 3. When SC==EC, it will use VERT		

Generated Code

DIAG/HORZ/VERT
thickness;SR;SC;ER;EC
STOP

Font Setup - PTXA_FontSetup

Function	int PTXA_FontSetup(int mode, char* value)	
Purpose	Set font of text.	
Input Parameter	**Range / Type**	**Description**
int mode	See Table 2 below.	
char* value	See Table 2 below.	
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Table 2 Font Setup Parameters

mode	***value***	**Comments**
PTXA_FontName	String	Identifies the specific typeface.
PTXA_FontBold	PTXA_BoldOn PTXA_BoldOff	Selects a bold attribute.
PTXA_FontSlant	PTXA_SlantRight PTXA_SlantLeft	Selects a slanting factor
PTXA_FontEncoding	PTXA_CPAscii PTXA_CPGB PTXA_CPBig5 PTXA_CPKSC PTXA_CPSJIS PTXA_CPUTF8	Selects a font symbol set.

ASCII Text - PTXA_PrintTextEng

Function	int PTXA_PrintTextEng(int lForm, int SR, int SC, int VE, int HE, char * data)	
Purpose	Define and positions ASCII text on static data field	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to be printed
int SR	0 – length of form	This parameter can be dot row or character row based on scale setting. Does not support the CP.DP format. The default is character row.
int SC	0 – width of form	This parameter can be dot column or character column based on scale setting. Does not support the CP.DP format. The default is character column.
int VE	0 – 63	Vertical expansion is a multiple by default height.
int HE	0 – 63	Horizontal expansion is a multiple by default width.
char *data	Text String Data	String data to be printed. This function will use the "*" character as the default delimiter for the data. If "*" is used as part the data, it will use another printable character for the delimiter (determined by DLL internally).
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Generated Code

```
ALPHA
SR;SC;VE;HE;*data*
STOP
```

Asian Text - PTXA_PrintTextCH

Function	int PTXA_ PrintTextCH (int lForm, int SR, int SC, int VE, int HE, char * data)	
Purpose	Define and positions ASCII or Hanzi text on static data field	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to be printed
int SR	0 – length of form	This parameter can be dot row or character row based on scale setting. Does not support the CP.DP format. The default is character row.
int SC	0 – width of form	This parameter can be dot column or character column based on scale setting. Does not support the CP.DP format. The default is character column.
int VE	0 – 63	Vertical expansion is a multiple by default height.
int HE	0 – 63	Horizontal expansion is a multiple by default width.
char *data	Text String Data	String data to be printed. This function will use the "*" character as the default delimiter for the data. If "*" is used as part the data, it will use another printable character for the delimiter (determined by DLL internally).
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Generated Code

```
TWOBYTE
SR;SC;VE;HE;*data*
STOP
```

Print Image - PTXA_PrintImage

Function	int PTXA_PrintImage (int lForm, int SR, int SC, char *logoname)	
Purpose	Prints logo	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to be printed
int SR	0 – length of form	This parameter can be dot row or character row based on scale setting. Does not support the CP.DP format. The default is character row.
int SC	0 – width of form	This parameter can be dot column or character column based on scale setting. Does not support the CP.DP format. The default is character column.
char *logoname	Name of Logo	The logo must be predefined via ~LOGO in PGL NORMAL mode, or LOGODEF in PGL CREATE mode.
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Generated Code

LOGO
SR;SC;logoname
STOP

Barcode Setting - PTXA_SetBarcode

Function	int PTXA_SetBarcode(int bcdtype, int mode, char *value)	
Purpose	Set the optional parameters for barcode, this API needs to be called before `PTXA_PrintBarcode()`, and the parameter is set one by one.	
Input Parameter	**Range**	**Description**
int bcdtype	PTXA_BCD_C39 PTXA_BCD_C128A PTXA_BCD_C128B PTXA_BCD_C128C PTXA_BCD_EAN13 PTXA_BCD_UPCA PTXA_BCD_MAXICODE PTXA_BCD_QR PTXA_BCD_PDF417	The type of barcode symbology. Constants are defined in `PtnxDllCommDef.h`.
int mode	See Table 3. Note that not all modes can be combined with every barcode type as shown in Table 4.	Determine the barcode characteristic to change.
char *value		The value of mode, in ASCII text format
Output Param	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed

Table 3 Barcode Option Mode and Value

mode	***value***	**Comments**
PTXA_BCDOptParam_RESET	None Required	Changes all values back to default.
PTXA_BCDOptParam_HEIGHT	n.m Format	Overall height of barcode symbol, value format is n.m where n is tenths of inch and m = IGP dots
PTXA_BCDOptParam_ROTATION	PTXA_BcdRotatePort PTXA_BcdRotateLand PTXA_BcdRotateInvPort PTXA_BcdRotateInvLand	Rotation: 0 degrees Rotation: 90 degrees Rotation: 180 degrees Rotation: 270 degrees
PTXA_BCDOptParam_MAGNIFY	X1, X1A, X2	Specifies the built-in barcode ratios to use (e.g., X1, X1A, X2). See PGL Programmers Reference Manual for available options.
PTXA_BCDOptParam_PDF	PTXA_BcdPDFOff PTXA_BcdPDFOn	PDF enables the Printable Data Field.
PTXA_BCDOptParam_RATIO	Narrow bar: Narrow Space: Wide bar : Wide Space	Ratio to supply your own set of narrow and wide bar values according to the PGL Programmer's Reference Manual.

NOTE:

Not all the modes can be combined with all the barcode type, the valid combinations are as follows, it will be checked inside DLL, the details can be referred to PGL Programmer's Reference Manual.

Table 4 Barcode Type versus Mode Allowed

Barcode Type	Modes Available
PTXA_BCD_C39	PTXA_BCDOptParam_HEIGHT PTXA_BCDOptParam_ROTATION PTXA_BCDOptParam_MAGNIFY PTXA_BCDOptParam_PDF
PTXA_BCD_C128A PTXA_BCD_C128B PTXA_BCD_C128C	PTXA_BCDOptParam_HEIGHT PTXA_BCDOptParam_ROTATION PTXA_BCDOptParam_MAGNIFY PTXA_BCDOptParam_PDF
PTXA_BCD_EAN13	PTXA_BCDOptParam_HEIGHT PTXA_BCDOptParam_ROTATION PTXA_BCDOptParam_MAGNIFY PTXA_BCDOptParam_PDF
PTXA_BCD_UPCA	PTXA_BCDOptParam_HEIGHT PTXA_BCDOptParam_ROTATION PTXA_BCDOptParam_MAGNIFY PTXA_BCDOptParam_PDF
PTXA_BCD_MAXICODE	PTXA_BCDOptParam_ROTATION
PTXA_BCD_QR	PTXA_BCDOptParam_ROTATION PTXA_BCDOptParam_RATIO *(only the x dimension in printer dots)*
PTXA_BCD_PDF417	PTXA_BCDOptParam_ROTATION

Generated Code

```
C3/9;CW;X2;H3
```

(barcode CODE39, rotation=90)
(barcode CODE39, magnify=X2)
(barcode CODE39, height=3)

Barcode Printing - PTXA_PrintBarcode

Function	int PTXA_PrintBarcode (int lForm, int bcdtype, int SR, int SC, char *data)	
Purpose	Barcode specified will be included in the form.	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to have the barcode included
int bcdtype	PTXA_BCD_C39 PTXA_BCD_C128A PTXA_BCD_C128B PTXA_BCD_C128C PTXA_BCD_EAN13 PTXA_BCD_UPCA PTXA_BCD_MAXICODE PTXA_BCD_QR PTXA_BCD_PDF417	The type of barcode symbology.
int SR	0 – form length	This parameter can be dot row or character row based on scale setting. Does not support the CP.DP format. The default is character row.
int SC	0 – form width	This parameter can be dot column or character column based on scale setting. Does not support the CP.DP format. The default is character column.
char *data	Symbology Data	Data to be encoded in the barcode. This function will use the "*" character as the default delimiter for the data. If "*" is used as part the data, it will use another printable character for the delimiter (determined by DLL internally).
Output Param	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed, Parameter Invalid

Generated Code

```
BARCODE
C3/9;CW;A2  (barcode CODE39, rotation=90, magnify=A2)
SR;SC
*data*
PDF         (pdf=1)
STOP
```

RFID

RFID Setup - PTXA_SetupRFID

Function	int PTXA_SetupRFID(int length, bool incremental, char* format, int increment_amount, char* data)	
Purpose	This API sets up the parameter for the RFID.	
Input Parameter	**Range / Type**	**Description**
int length	0-1000	A decimal number specifying the bit length of a field within a tag
bool incremental	True/False	This parameter set bit field from non-incremental data format to incremental fixed data format.
char* format	B, D, and H	This parameter specify the format for the passcode data. B for binary, D for decimal, and H for hexadecimal.
int increment_amount	0-1000	If bit field is incremental fixed data format, this parameter specifies the amount to increment each time the form is executed.
char *data	Symbology Data	Data to be set to RFID. This function will use the "*" character as the default delimiter for the data. If "*" is used as part the data, it will use another printable character for the delimiter (determined by DLL internally).
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed, Parameter Invalid

RFID Writting - PTXA_WriteRFID

Function	int PTXA_WriteRFID(int lForm, char* LOCK, char* membank)	
Purpose	Program an RFID tag	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to have the RFID included
char* LOCK	1-FFFFFFFF	Optional parameter to lock the data block to prevent it from being overwritten
char * membank	EPC TID USR ACS KIL PC	Specifies which tag logical memory area that this command will be applied
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed, Parameter Invalid

RFID Verifying - PTXA_VerifyRFID

Function	int PTXA_VerifyRFID(int lForm, char* UNLOCK, char* membank, int length, char* format)	
Purpose	This API read the content of an RFID tag into a dynamic field, and request the printer to send to the host the ASCII representation.	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to have the RFID included
char* UNLOCK	1-FFFFFFFF	Optional parameter to unlock the data block so it can be overwritten later
char * membank	EPC TID USR ACS KIL PC	Specifies which tag logical memory area that this command will be applied
int length	0-1000	A decimal number specifying the overall bit length of the RFID tag memory bank
char * format	B, D, H	A letter specifying the representation format of the field data. B for binary, D for decimal, and H for hexadecimal.
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed, Parameter Invalid

Form Printing

Form Printing – PTXA_PrintForm

Function	(char *) PTXA_PrintForm (int lForm)	
Purpose	This API returns and send the form created by PTXA_CreateForm to user's application, DLL will proceed the APIs based on the input from user's application and generate PGL form (including the printer setup and creating form), return the pointer of buffer which hold the entire PGL program to user's application. Caller can send them to the printer or re-store them to the particular memory in user's application.	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to be printed
Output Parameter	**Range / Type**	**Description**
Return Value	Pointer to buffer	Pointer to buffer which holds the entire PGL program, and end with "\0".

Get Form Data – PTXA_GetFormData

Function	char* PTXA_ GetFormData (int lForm)	
Purpose	This API copies the form data created by PTXA_ PrintForm into user's buffer.	
Input Parameter	**Range / Type**	**Description**
int lForm	0-1000	Form to be printed
Output Parameter	**Range / Type**	**Description**
Return Value	Pointer to buffer	Pointer to buffer which holds the entire PGL program, and end with "\0".

Getting Data from Printer

Get Data From Printer – PTXA_Read

Function	char* PTXA_Read(char * EndSymbol)	
Purpose	This API copies the form data created by PTXA_ PrintForm into user's buffer by the requested data length.	
Input Parameter	**Range / Type**	**Description**
char* EndSymbol	String	End symbol of the send back data
Output Parameter	**Range / Type**	**Description**
Return Value	Pointer to buffer	Pointer to the buffer to hold the data sent back

Sending Command to Printer

Send Command to Printer –sendcommand

Function	int sendcommand (char * printercommand)	
Purpose	This API send command to printer..	
Input Parameter	**Range / Type**	**Description**
char* printercommand	String	The printer command to send.
Output Parameter	**Range / Type**	**Description**
Return Value	PTXA_OK PTXA_INVALID	Operation was Successful Operation Failed, Parameter Invalid

5 *Application Examples*

Print a Label

This example is also in the demo\PTXTestCode.cpp and shows how a given function can use the DLL to create a PGL label with the printer connected through USB. This was created using Visual Studio 10 and creating a simple GUI form with buttons to execute the function `RunDll()`.

```
#include "stdafx.h"              // Standard Microsoft Visual C++ Header
#include "MyForm.h"              // Default Microsoft Visual C++ GUI class
#include <string>
#include <iostream>
#include "PtnxDll.h"             // for DLL functions
#include "PtnxDllCommDef.h"      // for DLL constants, parameters

#using <mscorlib.dll>
using namespace std;
using namespace PTXLIB;          // Import for DLL functions

void RunDll(void)
{
   /* OPEN THE CONNECTION TO USB */
   PTXLIB::openport("USB");

   /* INIT THE DLL */
   PTXLIB::PTXA_DLLinit();

   // Coordinates are in units of 10 cpi, 6 lpi
   PTXLIB::PTXA_Scale(10, 6);

   // No sensor will be used for this job (e.g., continuous mode)
   PTXLIB::PTXA_JobSetup(PTXA_JobModeLabel, PTXA_JobLabelNoSensor);

   // Rotation of label is portrait
   PTXLIB::PTXA_JobSetup(PTXA_JobModeRotate, PTXA_JobRotatePort);

   // Create the label object
   int lForm = PTXLIB::PTXA_CreateForm("TestForm");
   cout << "Form Number:" << lForm <<endl;

   /* BOX */
   PTXLIB::PTXA_DrawBox(lForm, 2, 3, 4, 30, 30);

   /* HORZ LINES */
   PTXLIB::PTXA_DrawLine(lForm, 1, 14, 4, 14, 30);
   PTXLIB::PTXA_DrawLine(lForm, 1, 19, 4, 19, 30);
   PTXLIB::PTXA_DrawLine(lForm, 1, 24, 4, 24, 30);

   /* ALPHA - Default ASCII font */
   PTXLIB::PTXA_PrintTextEng(lForm, 11, 10, 1, 1, "CORPORATION");
   PTXLIB::PTXA_PrintTextEng(lForm, 12, 10, 0, 0, "STREET 1");
```

```
    PTXLIB::PTXA_PrintTextEng(lForm, 13, 10, 0, 0, "IRVINE, CA 92714");
    PTXLIB::PTXA_PrintTextEng(lForm, 4,  8,  0, 0, "FROM:");
    PTXLIB::PTXA_PrintTextEng(lForm, 10, 8,  0, 0, "TO:");
    PTXLIB::PTXA_PrintTextEng(lForm, 16, 8,  0, 0, "S.O.  S05995");
    PTXLIB::PTXA_PrintTextEng(lForm, 21, 8,  0, 0, "S/N: 456789");
    PTXLIB::PTXA_PrintTextEng(lForm, 26, 8,  0, 0, "P/N: 102245");

    /* ASIAN FONT - KAIU.TTF was downloaded separately to printer */
    PTXLIB::PTXA_FontSetup(PTXA_FontName, "KAIU.TTF");
    /* SELECT UTF-8 encoding with Asian Font */
    PTXLIB::PTXA_FontSetup(PTXA_FontEncoding, "UTF8");
    PTXLIB::PTXA_PrintTextCH(lForm, 6, 10, 1, 1, "一二三");
    PTXLIB::PTXA_PrintTextCH(lForm, 7, 10, 0, 0, "四五六");
    PTXLIB::PTXA_PrintTextCH(lForm, 8, 10, 1, 1, "123456");

    /* BARCODES (5/10 inches high) */
    PTXLIB::PTXA_SetBarcode(PTXA_BCD_C39, PTXA_BCDOptParam_HEIGHT, "5");
    PTXLIB::PTXA_PrintBarcode(lForm, PTXA_BCD_C39, 16, 8, "S05995");
    PTXLIB::PTXA_PrintBarcode(lForm, PTXA_BCD_C39, 21, 8, "456789");
    PTXLIB::PTXA_PrintBarcode(lForm, PTXA_BCD_C39, 26, 8, "102245");

    /* PRINT THE LABEL */
    PTXLIB::PTXA_PrintForm(lForm);

    /* CLOSE THE CONNECTION */
    PTXLIB::closeport();

    /* CLOSE THE DLL */
    PTXLIB::PTXA_DLLexit();
}
```

Get Status from Printer

This example is also in the demo\PTXTestCode.cpp and shows how to retrieve status from the printer when the `PTXA_GetStatus` command is issued. It assumes the same header and includes were present as in the prior example `RunDll()`.

```
void GetStatus(void)
{
    PTXLIB::PTXA_SetupStatus(PTXA_SnoopSer);
    std::cout << PTXLIB::PTXA_GetStatus();
}
```

Get ZGL Data to Printer

This example is also in the demo\PTXTestCode.cpp and shows how to send another emulation format such as ZGL to the printer in a raw format. This assumes the operator panel of the printer has been setup to receive ZGL using the **Application > Control > Active IGP Emul** menu. It assumes the same header and includes were present as in the prior example `RunDll()`.

```
void SendData(void)
{
    PTXLIB::sendcommand("^JR\n");
}
```

Printronix Auto ID Customer Support Center

IMPORTANT **Please have the following information available prior to calling the Printronix Customer Support Center:**

- Model number
- Serial number (located on the back of the printer)
- Installed options (i.e., interface and host type if applicable to the problem)
- Configuration printout: Refer to the *Administrator's Manual.*
- Is the problem with a new install or an existing printer?
- Description of the problem (be specific)
- Good and bad pictures that clearly show the problem (faxing or emailing of these pictures may be required)

Americas	(844) 307-7120 Service@PrintronixAutoID.com
Europe, Middle East, and Africa	+31 (0) 24 3030 340 EMEA_support@PrintronixAutoID.com
Asia Pacific	+886 3 990 6155 APAC_support@PrintronixAutoID.com
China	+86 755 2398 0479 CHINA_support@PrintronixAutoID.com

Corporate Offices

Printronix Auto ID
3040 Saturn Street, Suite
200, Brea, CA 92821
U.S.A.

Phone: (844) 307-7120
Fax: (657) 258-0817

Printronix Auto ID, EMEA Head Office
Georg-Wimmer-Ring
8b D-85604 Zorneding, Germany

Phone: +49 (0) 8106 37979-000
Email: EMEA_Sales@PrintronixAutoID.com

Printronix Auto ID, Asia Pacific Head Office
Taiwan
9F, No. 95, Minquan Rd.
Xindian Dist., New Taipei City
231 Taiwan (R.O.C)

Phone: +886 3 990 6155
Fax: +886 3 990 6215

Printronix Auto ID, China Head Office
Shenzhen
New World Center 2510 room
No. 6009, Yitian road
Futian District, Shenzhen
518000
China

Phone: +86 755 2398 0479
Fax: +86 755 2398 0773

Visit the Printronix web site at www.PrintronixAutoID.com