BeyondHMI

Developer's Guide: using BhiLibPcl and e!COCKPIT project PrintExample1

The following sections provide detailed instructions for using the BhiLibPcl library.

This document also includes an overview of the PrintExample1 e!COCKPIT project.

PrintExample1 is written entirely in structured text. It is intended to illustrate some features of the BhiLibPcl library.

BhiLibPcl is a library which allows Wago PFC200 PLCs to print directly to a PCL5-compatable laser or inkjet printer.

BhiLibPcl can also be used with certain point of sale thermal printers such as Star TSP800ii (over Ethernet) to print simple text with no PCL commands

Contents

Assembling the components of the solution

Obtain the Example program

Request the PrintExample1 project from info@beyond-hmi.com. This is a complete project developed for a specific configuration of PFC200 (750-8212 / 0040-0010) and I/O modules.

Modify the project to match your PFC200 configuration

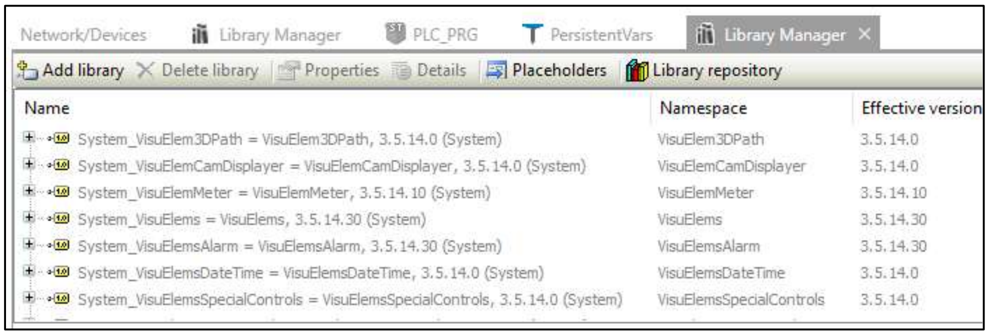
The Device in the project must be changed to match your available hardware configuration. Contact your Wago e!COCKPIT technical resource for assistance with this procedure. Alternatively, you can contact Beyond HMI at info@beyond-hmi.com to arrange assistance with this process.

Obtain the library file

Request the BhiLibPrint compiled-library from info@beyond-hmi.com. There is one version of the library.

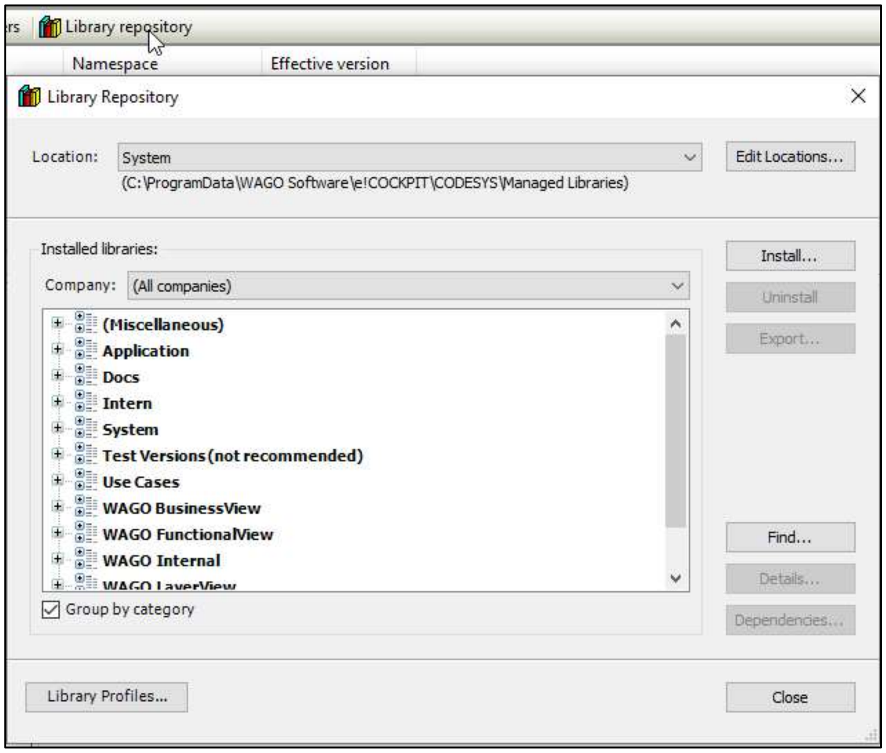
Install the library in your e!COCKPIT installation

- Open any project in e!COCKPIT
- Navigate to a Library Manager Node

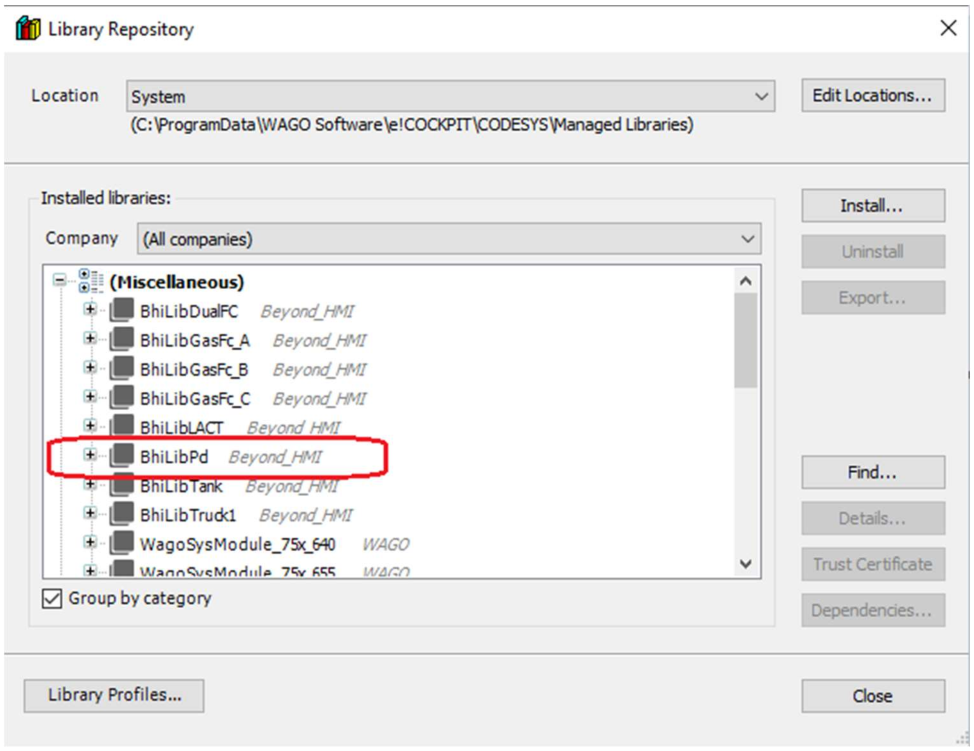


Name	Namespace	Effective version
System_VisuElem3DPath = VisuElem3DPath, 3.5.14.0 (System)	VisuElem3DPath	3.5.14.0
System_VisuElemCamDisplayer = VisuElemCamDisplayer, 3.5.14.0 (System)	VisuElemCamDisplayer	3.5.14.0
System_VisuElemMeter = VisuElemMeter, 3.5.14.10 (System)	VisuElemMeter	3.5.14.10
System_VisuElems = VisuElems, 3.5.14.30 (System)	VisuElems	3.5.14.30
System_VisuElemsAlarm = VisuElemsAlarm, 3.5.14.30 (System)	VisuElemsAlarm	3.5.14.30
System_VisuElemsDateTime = VisuElemsDateTime, 3.5.14.0 (System)	VisuElemsDateTime	3.5.14.0
System_VisuElemsSpecialControls = VisuElemsSpecialControls, 3.5.14.0 (System)	VisuElemsSpecialControls	3.5.14.0

- Select **Library Repository**

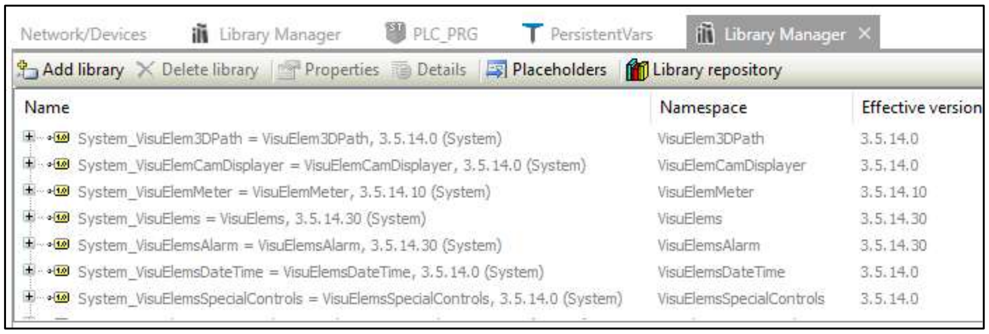


- Select **Install..**
- Navigate to the downloaded library file and click on **Open**.
- Verify that the library was installed in the **Miscellaneous** section



Add the Library to Your Project

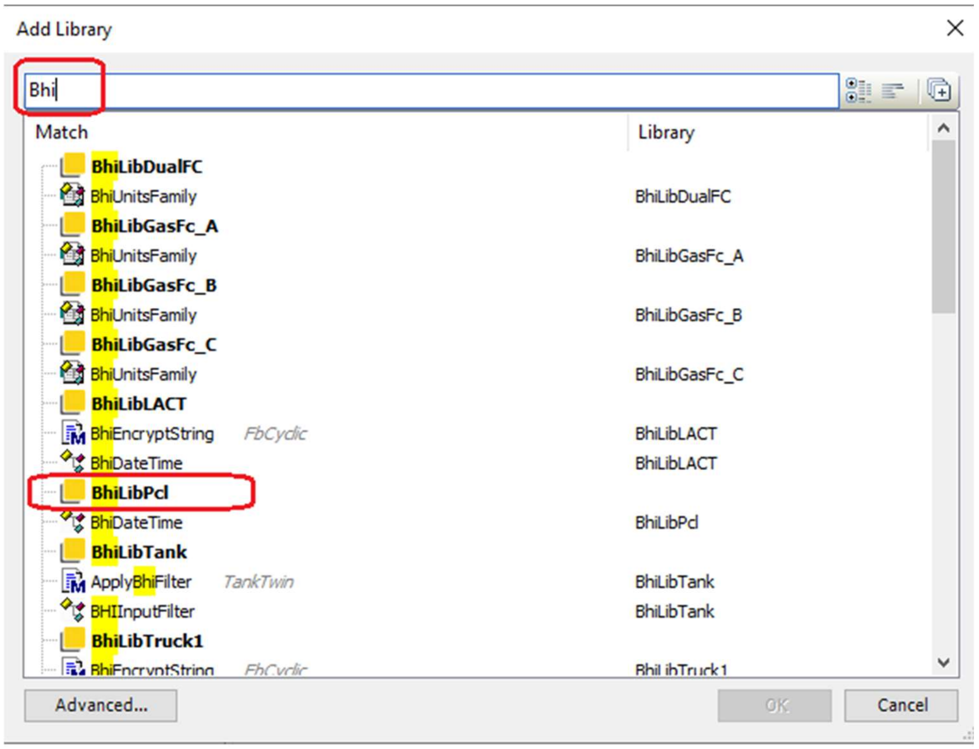
- Open the LACTExample1 project in e!COCKPIT.
- Navigate to a Library Manager



Name	Namespace	Effective version
System_VisuElem3DPath = VisuElem3DPath, 3.5.14.0 (System)	VisuElem3DPath	3.5.14.0
System_VisuElemCamDisplayer = VisuElemCamDisplayer, 3.5.14.0 (System)	VisuElemCamDisplayer	3.5.14.0
System_VisuElemMeter = VisuElemMeter, 3.5.14.10 (System)	VisuElemMeter	3.5.14.10
System_VisuElems = VisuElems, 3.5.14.30 (System)	VisuElems	3.5.14.30
System_VisuElemsAlarm = VisuElemsAlarm, 3.5.14.30 (System)	VisuElemsAlarm	3.5.14.30
System_VisuElemsDateTime = VisuElemsDateTime, 3.5.14.0 (System)	VisuElemsDateTime	3.5.14.0
System_VisuElemsSpecialControls = VisuElemsSpecialControls, 3.5.14.0 (System)	VisuElemsSpecialControls	3.5.14.0

Select **Add Library**

Start typing the library name until the library appears in bold text



Match		Library
BhiLibDualFC		
BhiUnitsFamily		BhiLibDualFC
BhiLibGasFc_A		
BhiUnitsFamily		BhiLibGasFc_A
BhiLibGasFc_B		
BhiUnitsFamily		BhiLibGasFc_B
BhiLibGasFc_C		
BhiUnitsFamily		BhiLibGasFc_C
BhiLibLACT		
BhiEncryptString	*FbCyclic*	BhiLibLACT
BhiDateTime		BhiLibLACT
BhiLibPcl		
BhiDateTime		BhiLibPcl
BhiLibTank		
ApplyBhiFilter	*TankTwin*	BhiLibTank
BHIInputFilter		BhiLibTank
BhiLibTruck1		
BhiEncryptString	*FbCyclic*	BhiLibTruck1

Select the Library and select **OK**.

Build and the project

You should now be able to successfully rebuild the project in e!COCKPIT.

Licensing

The BhiLibPrint library utilizes runtime licensing. Each PLC upon which it runs must have a license. Licenses are obtained from Beyond HMI, Inc.

Trial Mode

Upon startup, the library will run in trial mode for 100 printing operations. While in trial mode, the library is fully functional. After 100 print operations have been executed – and if no license is installed - the library will stop printing.

If the PLC program is stopped and restarted, the 100-print trail period begins again. Therefore, PLC program developers should be able to develop and test programs without needing a license for their development PLCs.

Steps to Obtain a Runtime License

To fully license the BhiLibPrint library on a PLC, the following steps must be executed:

- Include library features in a PLC program (*reference other instructions for PLC program developers within in this document*)
- Install the PLC program on the target PLC specimen
- Start the program running on the PLC
- Open the library's main screen and capture the Site Code
- Transmit the site code to Beyond HMI, Inc. and provide payment information
    - Please use info@beyond-hmi.com to initiate contact with us.
- Wait for Beyond HMI, Inc. to return a license file
- Install the license file in the PLC's /home/user/ directory. This can be accomplished using SSH/FTP tools or with Beyond HMI's **BHI License Tool (BLT)** Windows program. BLT can be downloaded at https://beyond-hmi.com/software-downloads
- Open the library's Main screen and confirm that the license check result is green

Licenses are perpetual. No maintenance fee is required. Licenses are keyed to a site code and are not portable between PLCs. Please contact Beyond HMI if you need to move a license to another PLC.

How your program can interact with the Library

Overview

The Fundamental procedure for using the library to print text consists of the following steps:

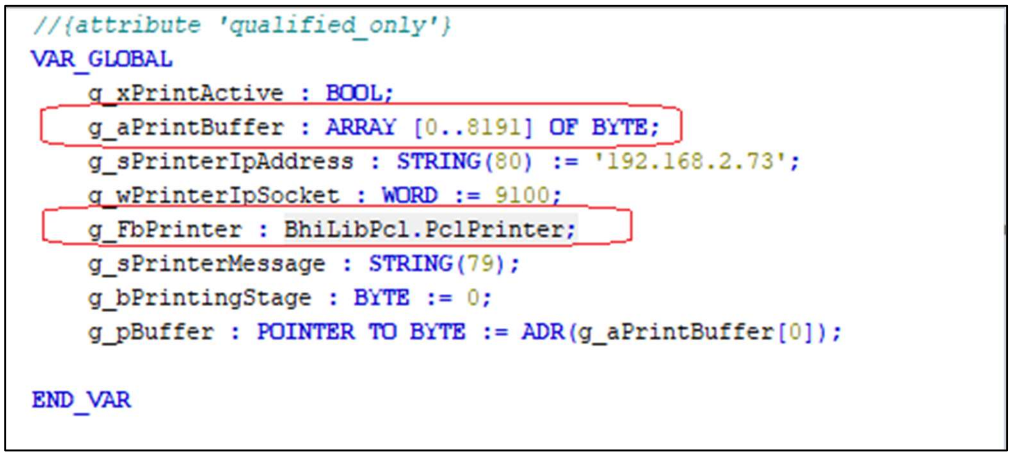
- Declare (in your program) an instance of the library's *PclPrinter* Function Block
- Declare (in your program) an array of bytes to use as a buffer
- Have your program call the *PclPrinter* function block on each scan
- Use the library's *HostAssignWorkingBuffer* function to give the library the allocated buffer for assembling text and commands to be printed
- Use the library's various functions to append text and commands to the buffer
- Call the *PclPrinter* function block's *PrintTextBuffer* method – passing the buffer.
- Call the *PclPrinter* function block's *xIsIdle* method repeatedly – until the method returns TRUE
- Use the *PclPrinter* function block's methods to get feedback about status of the print job

These steps are demonstrated in the PrintExample1 project.

Declaration of global variables

You will need to use the library function block called *PclPrinter*, so you will need to declare an instance of that function block.

The library can build a collection of text and commands to send to the printer, but it needs a buffer in which to store this data as it is being built. You must declare a buffer in your program and pass it to the library for use. The following excerpt from the PrintExample1 program illustrates this.

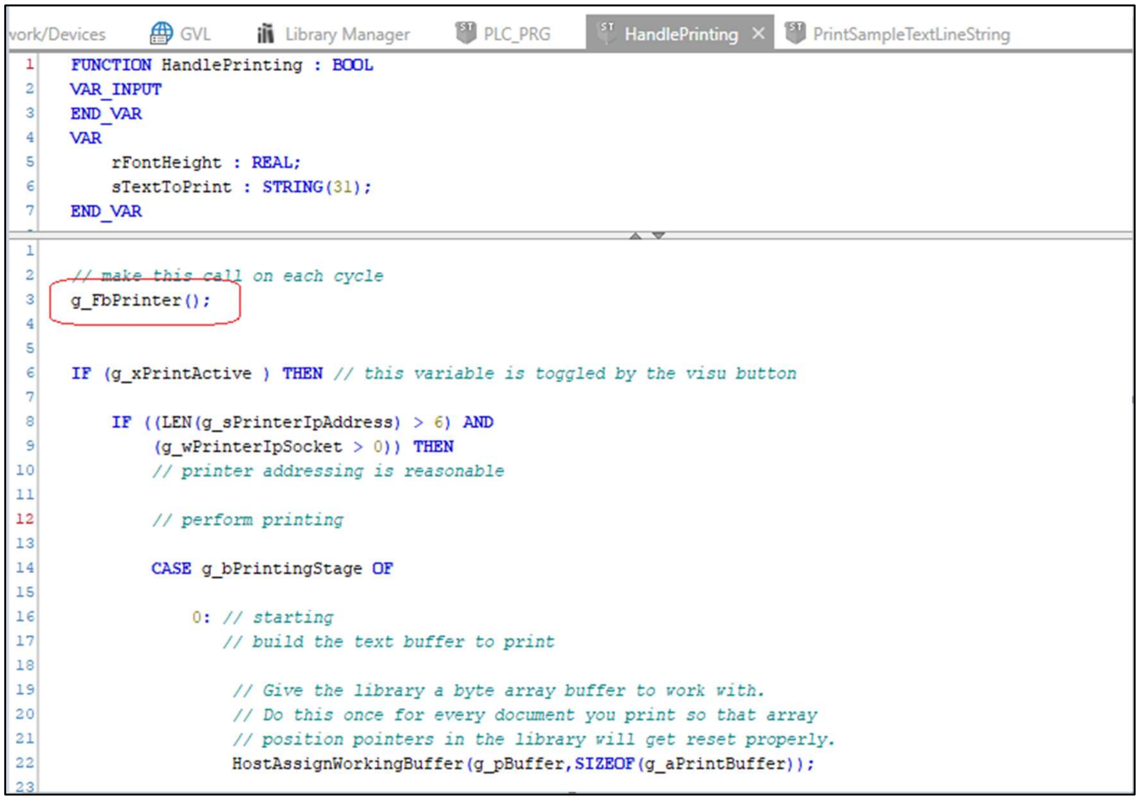


```
//{attribute 'qualified_only'}
VAR_GLOBAL
    g_xPrintActive : BOOL;
    g_aPrintBuffer : ARRAY [0..8191] OF BYTE;
    g_sPrinterIpAddress : STRING(80) := '192.168.2.73';
    g_wPrinterIpSocket : WORD := 9100;
    g_FbPrinter : BhiLibPcl.PclPrinter;
    g_sPrinterMessage : STRING(79);
    g_bPrintingStage : BYTE := 0;
    g_pBuffer : POINTER TO BYTE := ADR(g_aPrintBuffer[0]);

END_VAR
```

Note that the size of the buffer is to your discretion. It needs to be large enough to hold your text with about 10% to 30% extra space for printer control commands.

Call the PclPrinter Function Block on each cycle

The following excerpt from the PrintExample1 program illustrates this.

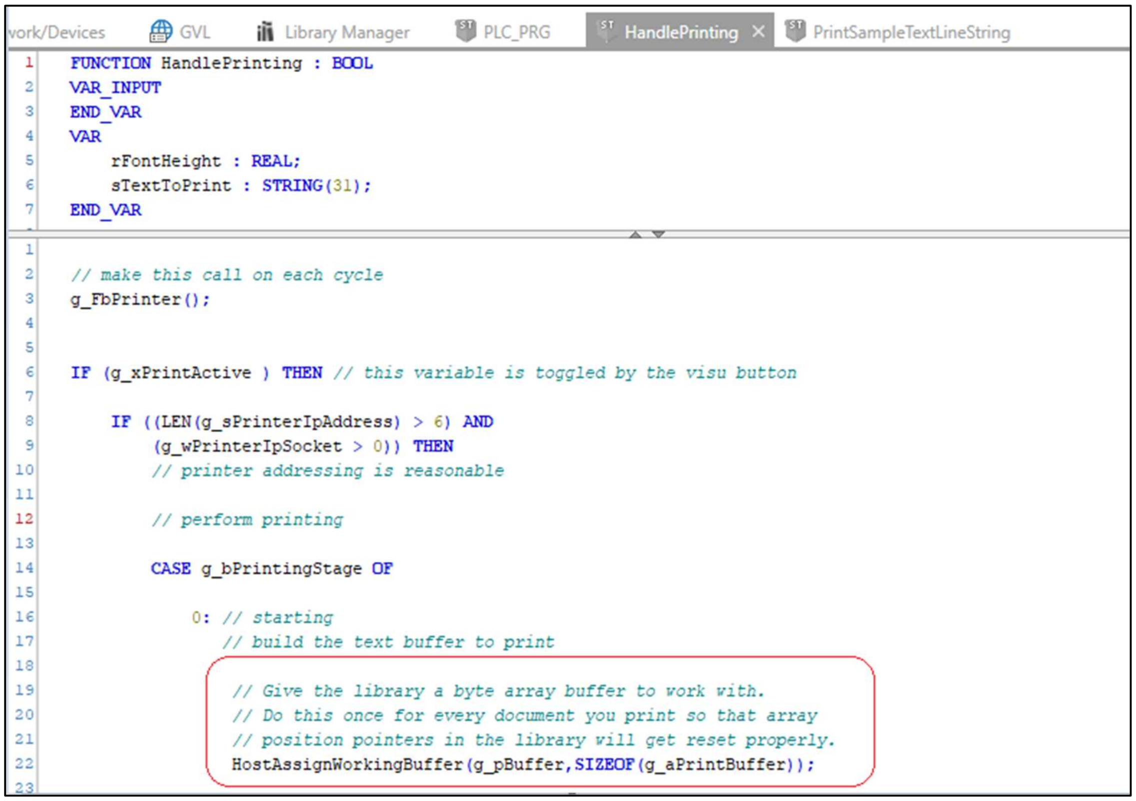


```
FUNCTION HandlePrinting : BOOL
VAR_INPUT
END_VAR
VAR
    rFontHeight : REAL;
    sTextToPrint : STRING(31);
END_VAR
```

```

// make this call on each cycle
g_FbPrinter();


IF (g_xPrintActive ) THEN // this variable is toggled by the visu button

    IF ((LEN(g_sPrinterIpAddress) > 6) AND
        (g_wPrinterIpSocket > 0)) THEN
        // printer addressing is reasonable

        // perform printing

        CASE g_bPrintingStage OF

            0: // starting
              // build the text buffer to print

               // Give the library a byte array buffer to work with.
               // Do this once for every document you print so that array
               // position pointers in the library will get reset properly.
               HostAssignWorkingBuffer(g_pBuffer,SIZEOF(g_aPrintBuffer));
```

Give the buffer to the Library

The following excerpt from the PrintExample1 program illustrates this.

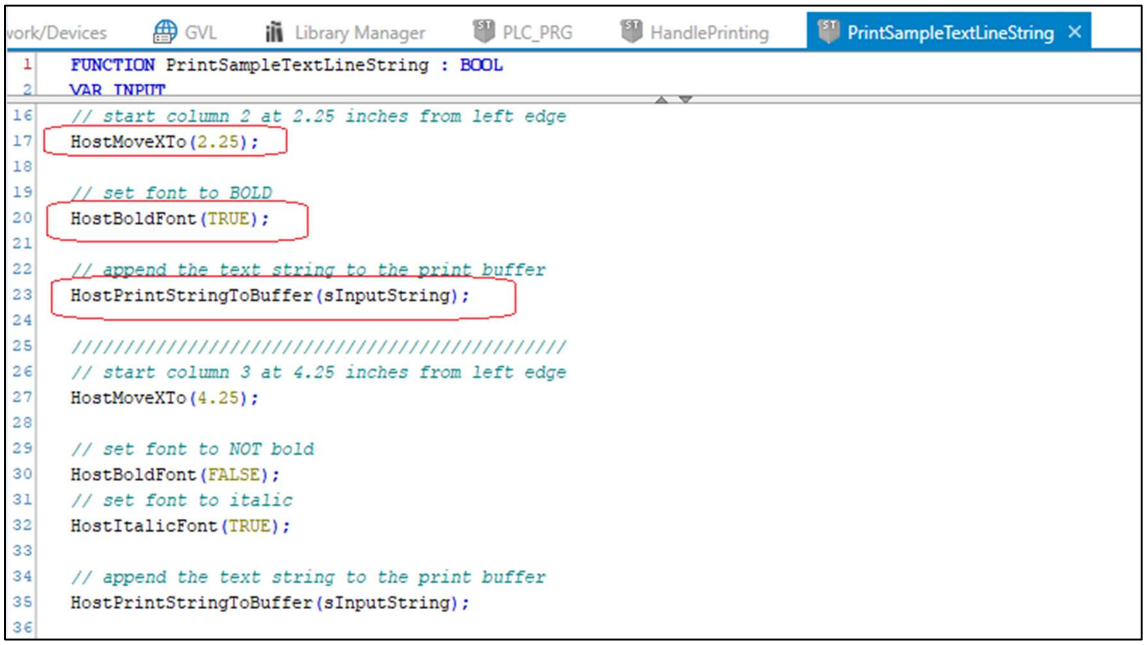


```
FUNCTION HandlePrinting : BOOL
VAR_INPUT
END_VAR
VAR
    rFontHeight : REAL;
    sTextToPrint : STRING(31);
END_VAR
```

```

// make this call on each cycle
g_FbPrinter();


IF (g_xPrintActive ) THEN // this variable is toggled by the visu button

    IF ((LEN(g_sPrinterIpAddress) > 6) AND
        (g_wPrinterIpSocket > 0)) THEN
        // printer addressing is reasonable

        // perform printing

        CASE g_bPrintingStage OF

            0: // starting
              // build the text buffer to print

               // Give the library a byte array buffer to work with.
               // Do this once for every document you print so that array
               // position pointers in the library will get reset properly.
               HostAssignWorkingBuffer(g_pBuffer,SIZEOF(g_aPrintBuffer));
```

Construct your string of text and control commands in the buffer using the library's functions
After assigning a buffer to the library, call the ***Host…*** functions to append text or commands into the buffer. These functions return a Boolean indicating whether the operation was successful. If the operation fails (return of FALSE), then the buffer size has been exceeded.

The following excerpt from the PrintExample1 program illustrates this. Note that in this example, the function return codes are not checked – for the sake of compact code. However, your code should check the return codes from the ***Host…*** functions.



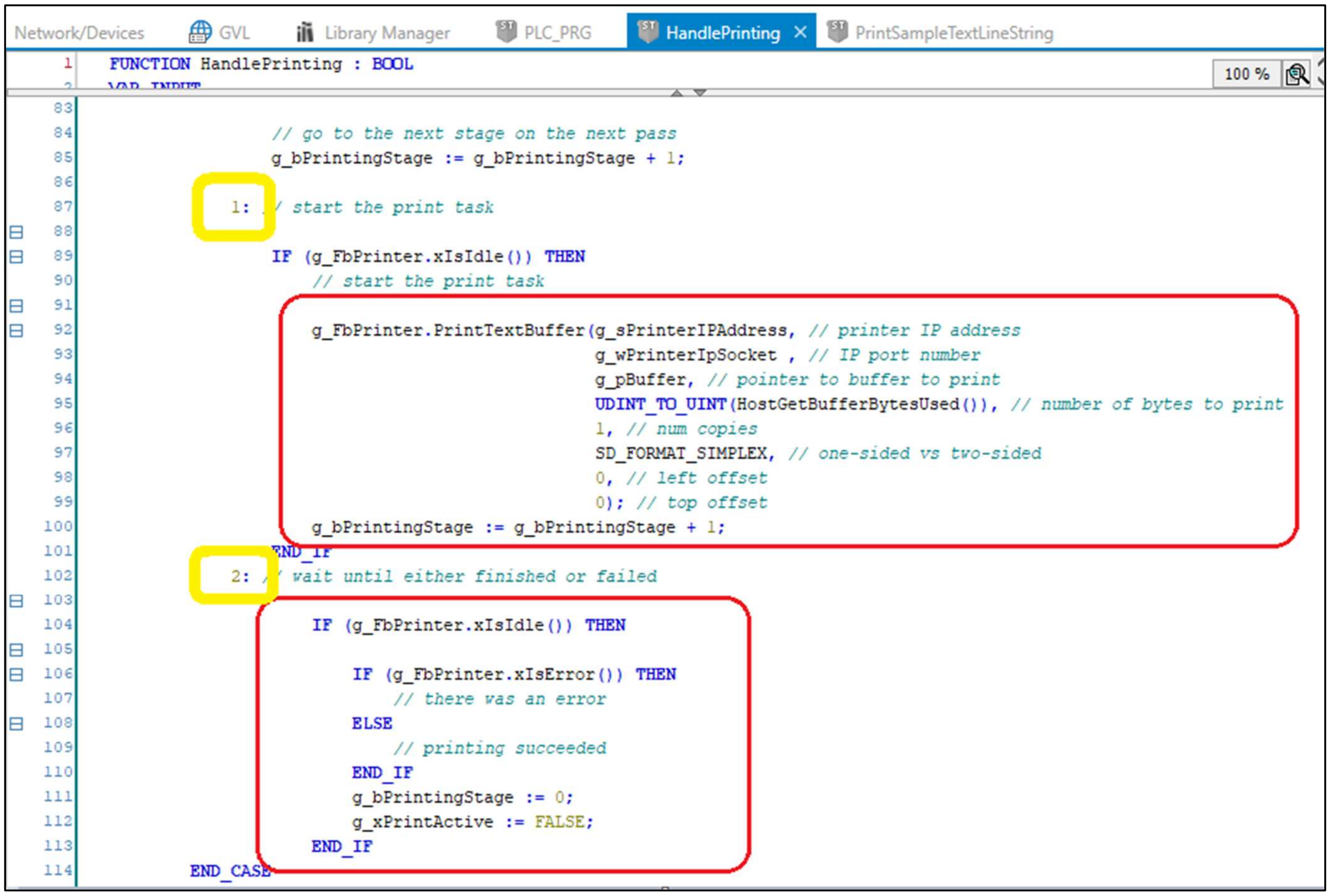
```
FUNCTION PrintSampleTextLineString : BOOL
VAR_INPUT
// start column 2 at 2.25 inches from left edge
HostMoveXTo(2.25);

// set font to BOLD
HostBoldFont(TRUE);

// append the text string to the print buffer
HostPrintStringToBuffer(sInputString);

/////////////////////////////////////////////////
// start column 3 at 4.25 inches from left edge
HostMoveXTo(4.25);

// set font to NOT bold
HostBoldFont(FALSE);
// set font to italic
HostItalicFont(TRUE);

// append the text string to the print buffer
HostPrintStringToBuffer(sInputString);

```

See the appendix of this documents for descriptions of each library ***Host…*** function.

Call the PclPrinter function block's PrintTextBuffer method and then the xIsIdle method
When the program has finished constructing the buffer full of text and printer commands, it must call the *PrintTextBuffer* Method of the *PclPrinter* function block instance. This method uses underlying Wago socket facilities to converse with the printer. These Wago sockets require multiple scans of the PLC to complete. Therefore, your program must call the method repeatedly – until the function block's *xIsIdle* method returns TRUE.

The following excerpt from the PrintExample1 program illustrates this. This code is within a ***case*** statement controlled by the value of the variable *g_bPrintStage*. When *g_bPrintStage* has a value of 1, the *PrintTextBufffer* method is called and the value of *g_bPrintStage* is changed to 2.

When *g_bPrintStage* has a value of 2 (on the subsequent cycles), the *xIsIdle* method is called. Until *xIsIdle* returns TRUE, the code just keeps checking for completion.



```
FUNCTION HandlePrinting : BOOL

                // go to the next stage on the next pass
                g_bPrintingStage := g_bPrintingStage + 1;

            1: // start the print task

                IF (g_FbPrinter.xIsIdle()) THEN
                    // start the print task

                    g_FbPrinter.PrintTextBuffer(g_sPrinterIPAddress, // printer IP address
                                                g_wPrinterIpSocket , // IP port number
                                                g_pBuffer, // pointer to buffer to print
                                                UDINT_TO_UINT(HostGetBufferBytesUsed()), // number of bytes to print
                                                1, // num copies
                                                SD_FORMAT_SIMPLEX, // one-sided vs two-sided
                                                0, // left offset
                                                0); // top offset
                    g_bPrintingStage := g_bPrintingStage + 1;
                END_IF
            2: // wait until either finished or failed

                IF (g_FbPrinter.xIsIdle()) THEN

                    IF (g_FbPrinter.xIsError()) THEN
                        // there was an error
                    ELSE
                        // printing succeeded
                    END_IF
                    g_bPrintingStage := 0;
                    g_xPrintActive := FALSE;
                END_IF
        END_CASE
```

Use the PclPrinter function block's methods to check success of the print job

The *xIsError* method returns TRUE if an error occurred. The *GetPrintTaskMessage* method provides details of the print job as a string. If there were no errors, this method will return "Success". Otherwise, an error message will be provided by this method.

Using BhiLibPCL to print LACT Batch Reports

For examples of how to print batch reports, consult the Developer's guide for the Beyond HMI's LACTExample1 project.

Appendix A: BhiLibPcl API

PclPrinter Function Block

The PclPrinter function block is the core element of the library. This function block interacts with the TCP/IP socket and passes data to the printer. This function block must be called on each cycle of the main e!COCKPIT project/program task.

PclPrinter Public Methods

Most of the functions of the PclPrinter function block are hidden from the end user. However, a few public methods are available.

PclPrinter.GetPrintMessage

Scope	Name	Type	Comment
Return	GetPrintMessage	String	Text description of the current status of the printer interaction on the TCP/IP socket.

PclPrinter.GetPrintTaskMessage

Scope	Name	Type	Comment
Return	GetPrintTaskMessage	String	Text description of the current status of the print job. This method returns the most comprehensive error messaging for the print job.

PclPrinter.GetLayoutLoadMessage

Relevant if using the library to print BhiLibLACT batches. For these types of jobs, the batch report layout file must be read from the file system so the batch information can be formatted.

Scope	Name	Type	Comment
Return	GetLayoutLoadMessage	String	Text description of the current status of the logic which reads layouts from file.

PclPrinter.Initialize

This method is optional and can be called to set the printer's TCP/IP socket addressing.

Scope	Name	Type	Comment
Input	isPrinterIPAddress	String	"dotted" TCP/IP address where the printer is accessed
Input	iwPrinterPort	WORD	TCP/IP port number to use for communications with the printer (typically 9100)
Return	Initialize	BOOL	Always Returns TRUE.

PclPrinter.PrintLactBatchWithLayout

Use this method to print a BhiLibLACT batch using a specific layout file to control formatting.

Scope	Name	Type	Comment

Input	isPrinterIPAddress	String	"dotted" TCP/IP address where the printer is accessed
Input	iwPrinterPort	WORD	TCP/IP port number to use for communications with the printer (typically 9100)
Input	itBatch	Type Batch	An instance of the batch struct populated with data from a specific batch
Input	isLayoutName	String	The name of the markup file to use in formatting the batch report – ***without*** the ".txt" extension [1]
Input	isMeterSN	String	The meter name/serial number of the meter which produced the batch
Input	ipBuffer	POINTER TO BYTE	Pointer to a buffer that the library can use to buffer the layout file for formatting[2]
Input	IuiBufferSize	UINT	Size of the buffer pointed to by ipBuffer[2]
Input	ibNumCopies	BYTE	Number of copies to print
Input	ibSimplexDuplex	BYTE	Enumerated value defining simplex/duplex page printing[3]
Input	iiLeftOffset	INT	Offset from the left edge of the printable page area where printing can begin. This value is in 1/300ths of an inch.
Input	iiTopOffset	INT	Offset from the top edge of the printable page area where printing can begin. This value is in 1/300ths of an inch.
Return	PrintLactBatchWithLayout	BOOL	Always Returns TRUE.

1 - Layout files must end with the extension ".txt", so the ".txt" extension is omitted from the isLayoutName parameter. Note that a file of this identical name must have been previously installed on the PLC using the **BLTool** utility program. For example, if a markup file called "my_batch_format.txt" is created by the end user, the "my_batch_format.txt" file must be installed (using BLTool) on the PLC and the isLayoutName argument should be "my_batch_format".

2 – the reference buffer must be large enough to contain the contents of the specified layout file. For a single-page batch report, 4096 bytes is probably sufficient.

3 – this value will be ignored if the printer does not support duplex formats. Valid values for this parameter are:

Value	**Meaning**
0	Simplex (one-sided)
1	Duplex Long (two-sided with page turning along the long edge of the paper)
2	Duplex Short (two-sided with page turning along the short edge of the paper)

PclPrinter.PrintLactBatchWithLayout_X

Use this method to print a BhiLibLACT batch using a specific layout file to control formatting – when the printer is connected via Ethernet but is not PCL-compatible and the printer does support simple text printing (such as Star TSP800ii).

Scope	Name	Type	Comment
Input	isPrinterIPAddress	String	"dotted" TCP/IP address where the printer is accessed
Input	iwPrinterPort	WORD	TCP/IP port number to use for communications with the printer (typically 9100)
Input	itBatch	Type Batch	An instance of the batch struct populated with data from a specific batch
Input	isLayoutName	String	The name of the markup file to use in formatting the batch report – ***without*** the ".txt" extension [1]
Input	isMeterSN	String	The meter name/serial number of the meter which produced the batch
Input	ipBuffer	POINTER TO BYTE	Pointer to a buffer that the library can use to buffer the layout file for formatting[2]
Input	IuiBufferSize	UINT	Size of the buffer pointed to by ipBuffer[2]
Input	ibNumCopies	BYTE	Ignored – use zero
Input	ibSimplexDuplex	BYTE	Ignored – use zero
Input	iiLeftOffset	INT	Ignored – use zero
Input	iiTopOffset	INT	Ignored – use zero
Return	PrintLactBatchWithLayout	BOOL	Always Returns TRUE.

1 - Layout files must end with the extension ".txt", so the ".txt" extension is omitted from the isLayoutName parameter. Note that a file of this identical name must have been previously installed on the PLC using the **BLTool** utility program. For example, if a markup file called "my_batch_format.txt" is created by the end user, the "my_batch_format.txt" file must be installed (using BLTool) on the PLC and the isLayoutName argument should be "my_batch_format".

2 – the reference buffer must be large enough to contain the contents of the specified layout file. For a single-page batch report, 4096 bytes is probably sufficient.

PclPrinter.PrintTextBuffer

Use this method to print a buffer full of text and printer control commands. This buffer can be built outside of the library or it can be constructed using the ***Host…*** library functions.

Scope	Name	Type	Comment
Input	isPrinterIPAddress	String	"dotted" TCP/IP address where the printer is accessed

Input	iwPrinterPort	WORD	TCP/IP port number to use for communications with the printer (typically 9100)
Input	ipBuffer	POINTER TO BYTE	Pointer to the buffer of bytes that needs to be sent to the printer
Input	IuiBufferSize	UINT	Size of the buffer pointed to by ipBuffer
Input	ibNumCopies	BYTE	Number of copies to print
Input	ibSimplexDuplex	BYTE	Enumerated value defining simplex/duplex page printing[3]
Input	iiLeftOffset	INT	Offset from the left edge of the printable page area where printing can begin. This value is in 1/300ths of an inch.
Input	iiTopOffset	INT	Offset from the top edge of the printable page area where printing can begin. This value is in 1/300ths of an inch.
Return	PrintTextBuffer	BOOL	Always Returns TRUE.

3 – this value will be ignored if the printer does not support duplex formats. Valid values for this parameter are:

Value	**Meaning**
0	Simplex (one-sided)
1	Duplex Long (two-sided with page turning along the long edge of the paper)
2	Duplex Short (two-sided with page turning along the short edge of the paper)

PclPrinter.xIsError
Use this method to check whether the status of the previous print job is in error or successful. This method only returns valid values when *xIsIdle* returns TRUE.

Scope	Name	Type	Comment
Return	xIsError	BOOL	Indication of whether the previous print job resulted in an error

PclPrinter.xIsIdle
Use this method to check whether the commanded print job is complete. After calling either *PrintLactBatchWithLayout* or *PrintTextBuffer*, this method will return FALSE until the job completes (successfully or with an error). After this method returns TRUE, use *xIsError* to determine whether the job succeeded. If the job produced an error, use *GetPrintTaskMessage* to obtain a text description of the error.

Scope	Name	Type	Comment
Return	xIsIdle	BOOL	Indication of whether the previous print job is active (FALSE) or complete (TRUE)

Host… Functions

The library provides an array of functions which can be used to append text or printer control commands to a buffer. The *PrintTextBuffer* method of the *PclPrinter* function block can then be used to print pages that contain the specified text at the specified locations on the page in the specified formats. These formatting functions all begin with the word "Host" – indicating that they are to be used by the library host program.

The basic process for printing a page is:

- Create a buffer for the printed text and commands
- Use the *HostAssignWorkingBuffer* function to give the library access to the buffer
- Use the **Host…** functions to append text and printer commands to the buffer
- Use the PrintTextBuffer method of the PclPrinter object to initiate printing

HostAssignWorkingBuffer

Use this method to provide the library with a location to buffer the text and printer commands. The library will begin using this buffer at the beginning and will keep track of how many bytes have been used.

Scope	Name	Type	Comment
Input	pBuff	POINTER TO BYTE	Pointer to the buffer of bytes that will be used by the library for assembling the document
Input	udBuffSize	UDINT	Size of the buffer pointed to by pBuff
Return	HostAssignWorkingBuffer	BOOL	Always Returns TRUE.

HostBoldFont

Enables or disables the bold stroke weight characteristic of the current font for all following characters by appending the appropriate PLC control characters to the working buffer.

Scope	Name	Type	Comment
Input	xBoldness	BOOL	Indication of whether to change to bold font (TRUE) or regular font (FALSE)
Return	HostBoldFont	BOOL	TRUE if operations successful, FALSE if buffer is full

HostGetBufferBytesUsed

Returns the number of bytes used in the working buffer.

Scope	Name	Type	Comment
Return	HostGetBufferBytesUsed	UDINT	number of bytes used in the working buffer

HostItalicFont

Enables or disables the italic stroke style characteristic of the current font for all following characters by appending the appropriate PLC control characters to the working buffer.

Scope	Name	Type	Comment
Input	xItalic	BOOL	Indication of whether to change to italic font (TRUE) or upright font (FALSE)
Return	HostItalicFont	BOOL	TRUE if operations successful, FALSE if buffer is full

HostMoveToNextLineAtXPosition
Moves the cursor down the page by one line and places the next text at the specified X Position by appending the appropriate PLC control characters to the working buffer.

Scope	Name	Type	Comment
Input	rNewXPos	REAL	New X Position (in inches)
Return	**HostMoveToNextLineAtXPosition**	BOOL	TRUE if operations successful, FALSE if buffer is full

HostMoveXTo
Moves the cursor to the specified X position (across the page), on the current line by appending the appropriate PLC control characters to the working buffer. Subsequent text will start at this new position

Scope	Name	Type	Comment
Input	rPosInInches	REAL	Floating point number of inches
Return	**HostMoveXTo**	BOOL	TRUE if operations successful, FALSE if buffer is full

HostMoveYDownNumLines
Moves the cursor down the page by the specified number of lines by appending the appropriate PLC control characters to the working buffer. The X position of the cursor is not changed by this command. Subsequent text will start at this new position. Line spacing is based on the font height of the current font.

See MoveToNextLineAtXPos for a more compact function to move down a single line and place the next text at a specific X position.

Scope	Name	Type	Comment
Input	iNumLines	INT	Number of lines to move down
Return	**HostMoveYDownNumLines**	BOOL	TRUE if operations successful, FALSE if buffer is full

HostNextPage
Causes the printer to move to the next page by appending the appropriate PLC control characters to the working buffer. Subsequent text will start on the new page.

Scope	Name	Type	Comment

Return	**HostNextPage**	BOOL	TRUE if operations successful, FALSE if buffer is full

HostPrintStringToBuffer

Causes the specified text to be printed by appending the appropriate PLC control characters to the working buffer.

Scope	Name	Type	Comment
Input	sInputString	STRING	String containing text to print
Return	**HostPrintStringToBuffer**	BOOL	TRUE if operations successful, FALSE if buffer is full

HostProportionalFont

Attempts to change font between available proportionally-spaced font and fixed-spaced font by appending the appropriate PLC control characters to the working buffer.

Scope	Name	Type	Comment
Input	xProportional	BOOL	Indication of whether to change to proportionally-spaced font (TRUE) or fixed-spaced font (FALSE)
Return	HostProportionalFont	BOOL	TRUE if operations successful, FALSE if buffer is full

HostSetDefaultFontHeight

If the print job does not begin with detailed specification of font characteristics, the library does not know what font will be used by the printer. However, the library needs to move the Y cursor in order to "move to the next line". That Y-direction movement is calculated based on the current font height. By default, the library assumes a 10-point font height – assuming that the default printer font is a 10-point font. If that assumption is incorrect, then line spacing will not work properly until a specific font height is set.

This function (used at the beginning of the job) can be used to correct line spacing problems in cases where you are attempting to use the default printer font and the default printer font height is not 10.

Scope	Name	Type	Comment
Input	rDefaultFontSize	BOOL	The font height to use in line spacing calculations
Return	HostSetDefaultFontHeight	BOOL	TRUE if operations successful, FALSE if buffer is full

HostSetFontHeight

Attempts to change the Font Height characteristic of the current font by appending the appropriate PLC control characters to the working buffer. Depending upon what fonts are loaded in the printer and the other characteristics of the current font, this command may or may not change the font. The printer will choose the font that most nearly matches the currently-defined characteristics.

Scope	Name	Type	Comment
Input	rFontHeight	BOOL	The font height to use for subsequent text
Return	HostSetFontHeight	BOOL	TRUE if operations successful, FALSE if buffer is full

HostSetFontPitch

Attempts to change the Font Pitch characteristic of the current font by appending the appropriate PLC control characters to the working buffer. Depending upon what fonts are loaded in the printer and the other characteristics of the current font, this command may or may not change the font. The printer will choose the font that most nearly matches the currently-defined characteristics.

Scope	Name	Type	Comment
Input	rFontPitch	BOOL	The font pitch to use for subsequent text
Return	HostSetFontPitch	BOOL	TRUE if operations successful, FALSE if buffer is full

HostSetFontSpacing

Attempts to change the Font Spacing characteristic of the current font by appending the appropriate PLC control characters to the working buffer. Depending upon what fonts are loaded in the printer and the other characteristics of the current font, this command may or may not change the font. The printer will choose the font that most nearly matches the currently-defined characteristics.

Scope	Name	Type	Comment
Input	bSpacing	BYTE	The font spacing to use for subsequent text. Valid Values are 0 (Fixed) and 1(Proportional)
Return	HostSetFontSpacing	BOOL	TRUE if operations successful, FALSE if buffer is full

HostSetFontStrokeWeight

Attempts to change the Font Spacing characteristic of the current font by appending the appropriate PLC control characters to the working buffer. Depending upon what fonts are loaded in the printer and the other characteristics of the current font, this command may or may not change the font. The printer will choose the font that most nearly matches the currently-defined characteristics.

Valid stroke weights include:

INT Value	Description
-7	Ultra Thin
-6	Extra Thin
-5	Thin
-4	Extra Light
-3	Light
-2	Demi Light
-1	Semi-Light
0	Medium
1	Semi Bold
2	Demi Bold
3	Bold
4	Extra Bold
5	Black
6	Extra Black
7	Ultra Black

Scope	Name	Type	Comment
Input	iStrokeWeight	INT	The font stroke weight to use for subsequent text
Return	HostSetFontStrokeWeight	BOOL	TRUE if operations successful, FALSE if buffer is full

HostSetFontStyle

Attempts to change the Font Style characteristic of the current font by appending the appropriate PLC control characters to the working buffer. Depending upon what fonts are loaded in the printer and the other characteristics of the current font, this command may or may not change the font. The printer will choose the font that most nearly matches the currently-defined characteristics.

Valid styles include:

INT Value	Description
0	Upright
1	Italic
4	Condensed
8	Compressed
24	Expanded
32	Outlined
64	Inlined
128	Shadowed

Scope	Name	Type	Comment
Input	iFontStyle	INT	The font style to use for subsequent text
Return	HostSetFontStyle	BOOL	TRUE if operations successful, FALSE if buffer is full

HostSetFontSymbolSet
Attempts to change the Font Symbol Set characteristic of the current font by appending the appropriate PLC control characters to the working buffer. Depending upon what fonts are loaded in the printer and the other characteristics of the current font, this command may or may not change the font. The printer will choose the font that most nearly matches the currently-defined characteristics.

Valid symbol sets include:

STRING Value	**Description**
"1F"	ISO069 French
"0N"	ISO8859 Latin
"0U"	ISO6 ASCII
"1U"	Legal
"8U"	Roman 8 (normal default symbol set)
"10U"	PC8
"0Y"	3 of 9 Barcode
"29U"	Windows 3pt Latin

Scope	Name	Type	Comment
Input	sSymbolsSet	STRING	The symbol set to use for subsequent text
Return	HostSetFontSymbolSet	BOOL	TRUE if operations successful, FALSE if buffer is full

HostSetTypefaceFamily
Attempts to change the Font Typeface Family characteristic of the current font by appending the appropriate PLC control characters to the working buffer. Depending upon what fonts are loaded in the printer and the other characteristics of the current font, this command may or may not change the font. The printer will choose the font that most nearly matches the currently-defined characteristics.

Check your printer documentation for valid typeface families loaded on the printer. Alternatively, you can load fonts onto most printers to augment the default font set.

Some common typeface families include:

UINT Value	**Description**

0	Line Printer
4099	Courier

Scope	Name	Type	Comment
Input	uiFamily	UINT	The typeface family to use for subsequent text
Return	HostSetTypeFaceFamily	BOOL	TRUE if operations successful, FALSE if buffer is full

Formatting Utility Functions

The library contains two functions which can be used to format date/times and numbers into strings.

FormatDT

Formats a CODESYS DT datatype into a string with customary U.S. "day/month/year (H)H:MM:SS am/pm" format.

- To get the date only as a string, set xIncludeDate to TRUE and xIncludeTime to FALSE
- To get the time only as a string, set xIncludeDate to FALSE and xIncludeTime to TRUE
- To get a string including both date and time, set xIncludeDate to TRUE and xIncludeTime to TRUE

Scope	Name	Type	Comment
Input	dtTmsp	DT	The date/time to format
Input	xIncludeDate	BOOL	Include the date in the string
Input	xIncludeTime	BOOL	Include the time in the string
Return	FormatDT	STRING	The formatted string

FormatWithDecimals

Formats an LREAL into a string with the specified number of decimal places. If the number is less than 1, the formatted value is on traditional format with leading zeros – rather than scientific notation.

The last digit is rounded (0-4 rounds down, 5-9 rounds up).

- Works correctly values down to 0.000000001

Scope	Name	Type	Comment
Input	rValue	LREAL	The number to Format
Input	bNumDecimals	BTYE	The number of decimal places desired
Return	FormatWithDecimals	STRING	The formatted string

Debugging Utility Functions

One utility function is provided to assist in debugging.

DebugBuffer
Converts a portion of a byte array to a readable string.

Note: the input string (sOstr) should be allocated as STRING(255) and uiBufLen should not exceed 255.

Scope	Name	Type	Comment
Input	pByte	POINTER TO BYTE	Pointer to the buffer where characters exist
Input	udiBufLen	UDINT	Number of bytes to translate (must be less than 255)
Input_Output	sOstr	STRING(255) ref	Host-program allocated STRING(255) where readable string will be returned.
Return	DebugBuffer	BOOL	Always returns TRUE

Other Utility Functions

Other utility functions are available to copy or append specific printer commands into a user-supplied buffer. It is normally not necessary to use these functions.

AppendCrLfToPrinterBufferLine
Appends carriage return and line feed characters to the specified buffer at the specified location.

Scope	Name	Type	Comment
Input	pBuff	POINTER TO BYTE	Pointer to the buffer where characters are to be appended
Input	udiBuffLen	UDINT	Total length of the buffer
Input_Output	udiMessageLen	UDINT ref	Index into the array where characters are to be appended. This value is incremented by 2 inside the function.
Return	AppendCrLfToPrinterBufferLine	BOOL	TRUE if operations successful, FALSE if buffer is full

AppendFfToPrinterBufferLine
Appends a form feed character to the specified buffer at the specified location.

Scope	Name	Type	Comment
Input	pBuff	POINTER TO BYTE	Pointer to the buffer where character is to be appended
Input	udiBuffLen	UDINT	Total length of the buffer
Input_Output	udiMessageLen	UDINT ref	Index into the array where character is to be appended. This

			value is incremented by 1 inside the function.
Return	AppendFfToPrinterBufferLine	BOOL	TRUE if operations successful, FALSE if buffer is full

AppendToPrinterBufferLine
Appends a specified number of bytes from a byte buffer to the specified buffer at the specified location.

Scope	Name	Type	Comment
Input	pBuff	POINTER TO BYTE	Pointer to the buffer where character is to be appended
Input	udiBuffLen	UDINT	Total length of the buffer
Input	pTextToAppend	POINTER TO BYTE	The buffer containing the bytes you want to append to the pBuff buffer
Input	udNumBytesToAppend	UDINT	The number of bytes to append
Input_Output	udiMessageLen	UDINT ref	Index into the array where character is to be appended. This value is incremented by udNumBytesToAppend inside the function.
Return	AppendToPrinterBufferLine	BOOL	TRUE if operations successful, FALSE if buffer is full

ASCIIBYTE
Returns a Byte representing the ASCII code for the first byte in the input STRING. For example, ASCIIBYTE('B') returns a decimal value of 66.

Scope	Name	Type	Comment
Input	sInStr	STRING	A string containing at least one character
Return	ASCIIBYTE	BYTE	ASCII code for the FIRST character in the string

AssignLayoutBuffer
Used when the library is being used to print formatted batches from BhiLibLACT. This function provides the library with a host-program-allocated buffer to use in buffering report layout files. The size of this buffer must be large enough to hold the largest layout file used by the library on the target PLC instance.

Scope	Name	Type	Comment
Input	iPBuffer	POINTER TO BYTE	Pointer to the host-program-allocated buffer
Input	iuiBufferSize	UINT	Size of the buffer

Return	AssignLayoutBuffer	BOOL	Always returns TRUE

BhiLastSigDig
Returns a Byte representing the number of significant digits in a LREAL with a value of less than 1.0.

Scope	Name	Type	Comment
Input	lriVal	LREAL	An LREAL with value < 1.0
Return	BhiLastSigDigit	BYTE	Decimal place where the last significant digit is located (ex: 0.001234 returns 6)

BhiLrDigit
Returns a Byte representing the integer value of an LREAL with value 0.0 < x < 9.9999(repeating).

Scope	Name	Type	Comment
Input	lriVal	LREAL	An LREAL with value 0.0 < x < 9.9999(repeating)
Return	BhiLrDigit	BYTE	Byte representation (always rounded down) of the input value (ex: 1.67 returns 1)

BoldFont
Fills the provided buffer with characters to change font to/from bold. Function assumes that the buffer is large enough to hold the printer command characters.

Scope	Name	Type	Comment
Input	xMakeBold	BOOL	TRUE to make font bold, FALSE to make font normal
Input	pSendBuffer	POINTER TO BYTE	Pointer to the buffer
Return	BoldFont	BYTE	Number of byes used in the buffer

BuildJobSeparationMessage
Fills the provided buffer with characters to end a print job

Scope	Name	Type	Comment
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Input	udiBuffLen	UDINT	Length of Buffer
Return	BuildJobSeparationMessage	UDINT	Number of byes used in the buffer

BuildJobHeaderMessage
Fills the provided buffer with characters to begin a print job.

Scope	Name	Type	Comment

Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Input	udiBuffLen	UDINT	Length of Buffer
Input	bNumCopies	BYTE	Number of copies to print
Input	bSimplexDuplex	BYTE	0 = one-sided
Input	iTopOffset	INT	Offset from top edge of printable area where text should start - in 1/300ths of an inch
Input	iLeftOffset	INT	Offset from left edge of printable area where text should start - in 1/300ths of an inch
Return	BuildJobHeaderMessage	UDINT	Number of byes used in the buffer

BuildPrinterResetMessage
Fills the provided buffer with characters to reset the printer.

Scope	Name	Type	Comment
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Input	udiBuffLen	UDINT	Length of Buffer
Return	BuildPrinterResetMessage	UDINT	Number of byes used in the buffer

BuildUniversalExitMessage
Fills the provided buffer with characters to terminate the PCL language interpreter in the printer.

Scope	Name	Type	Comment
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Input	udiBuffLen	UDINT	Length of Buffer
Return	BuildUniversalExitMessage	UDINT	Number of byes used in the buffer

BuildVMIMessageMessage
Fills the provided buffer with characters to set the vertical measurement units used when a carriage return is applied.

Scope	Name	Type	Comment
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Input	udiBuffLen	UDINT	Length of Buffer
Input	rPrintableHeightInches	REAL	Size of printable height of the page
Input	bLinesPerPage	BYTE	Number of lines per page
Return	BuildVMIMessage	UDINT	Number of byes used in the buffer

MoveXTo

Fills the provided buffer with characters to set the horizontal location of the next printed characters. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment
Input	rXposInInch	REAL	X-position in inches
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	MoveXTo	BYTE	Number of byes used in the buffer

MoveYDownNumLines

Fills the provided buffer with characters to move the cursor down the page to the location of the next line – based on the specified font height. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment
Input	iNumLines	INT	Number of lines to move down
Input	rFontHeightPoints	REAL	Font height in points
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	MoveYDownNumLines	BYTE	Number of byes used in the buffer

SetFontHeight

Fills the provided buffer with characters to attempt to change the font height for subsequent characters. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment
Input	rFontHeight	REAL	Font height in points
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	SetFontHeight	BYTE	Number of byes used in the buffer

SetFontPitch

Fills the provided buffer with characters to attempt to change the font pitch for subsequent characters. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment
Input	rPitch	REAL	Font pitch
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	SetFontPitch	BYTE	Number of byes used in the buffer

SetFontSpacing
Fills the provided buffer with characters to attempt to change the font spacing for subsequent characters. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment
Input	bSpacing	BYTE	0 = fixed, 1 = proportional
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	SetFontSpacing	BYTE	Number of byes used in the buffer

SetFontStrokeWeight
Fills the provided buffer with characters to attempt to change the font weight for subsequent characters. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment
Input	iStrokeWeight	INT	The stroke weight Range: STROKE_WEIGHT_ULTRA_THIN (-7) to STROKE_WEIGHT_ULTRA_BLACK (+7)
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	SetFontWeight	BYTE	Number of byes used in the buffer

SetFontStyle
Fills the provided buffer with characters to attempt to change the font style for subsequent characters. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment
Input	iStyle	INT	The stroke style
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	SetFontStyle	BYTE	Number of byes used in the buffer

SetFontSymbolSet
Fills the provided buffer with characters to attempt to change the font symbol set for subsequent characters. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment

Input	sSymbolsSet	STRING	The symbol set
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	SetFontSymbolSet	BYTE	Number of byes used in the buffer

SetFontTypefaceFamily

Fills the provided buffer with characters to attempt to change the font typeface family for subsequent characters. The function assumes that the provided buffer is large enough to hold the characters.

Scope	Name	Type	Comment
Input	uiTypefaceFamily	UINT	typeface family code
Input	pBuff	POINTER TO BYTE	Pointer to the buffer
Return	SetFontTypefaceFamily	BYTE	Number of byes used in the buffer