Babbo XP Series

Scanner Manager Library for Palm®OS

Programmer's Manual

Version 2.1

Amendment History

Date	Amended By	Version	Revision Comments
20 October, 2006	JL	1.0	Preliminary Version
27 October, 2006	JL	1.1	Initial Version
20 November, 2006	JL	1.2	**Scan Manager Library:** Add ScanWedge On/Off function **ScanDemo Sample:** Rename the target prc name as JanamDemo.prc Remove "keyDownEvent" for ScanButton (It will handle in Scan Library.) Fix symbology min/max length setting. Allow user set min/max length for symbology support min/max length setting.
1 December, 2006	JL	1.3	Fix handling for long barcode data (greater than 246 characters) in ScanDemo application Add Hardware Utilities Library section Add supported functions: ScnSetTriggeringModes ScnGetTriggeringModes
9 January 2007	JL	1.4	Add ScanWedge Support function: ScnCmdScanWedgeInsertChar
25 May, 2007	JL	1.5	Add functions: ScnCmdGetSetExposureMode ScnCmdGetSetExposureSettings
7 June, 2007	JL	1.6	Add function: ScnCmdGetSetDecodeOptions Add example codes for: ScnCmdGetSetExposureSettings

8 June, 2007	JL	1.7	Fix define in ScanMgrStruct.h "DEC_DECODE_OPTIONS_VER" from 0x3F to 0x03 for latest decoder for the function: ScnCmdGetSetDecodeOptions
17 August, 2007	JL	1.8	Add Bluetooth enable/disable controls: HwrUtilsLibGetBTState HwrUtilsLibSetBTState
8 May, 2007	JL	1.9	Add 5V out controls: HwrUtilsLibGet5Vout HwrUtilsLibSet5Vout
10 September, 2008	JL	2.0	Add centering window control function: ScnCmdGetSetDecodeCenteringWindow
17 December, 2008	JL	2.1	Add keyboard click sound enable function: HwrUtilsLibGetNumericKeyClickSnd HwrUtilsLibSetNumericKeyClickSnd

Table of Content

Introduction

Purpose

The *Barcode Scanner Library Software Manual* contains important information about using the Barcode Scanner Manager Library. This document is intended as a guide for developers interested in creating or modifying application for Babbo series barcode scanner devices

Section Descriptions

- **Using the Babbo Scanner Manager Library** – A guide to create a simple application that can perform barcode scanning and change scanning options
- **Core Functions** – A list of core functions that control the barcode scanning
- **Decoder Functions** – A list of decoder functions that change the scanning behaviour associated with specific types of barcodes.
- **Scanner Functions** – A list of scanner functions that control the barcode scanning behaviour
- **Extended Functions** – A list of extended functions only in Babbo series devices (i.e. 2D symbology configuration)
- **Application Porting –** A guide to migrate existing Symbol application to use the Babbo Scanner Manager library
- **Sample Demo Application** – Description of the demo application included with the SDK.

Using the Babbo Scanner Manager Library

Using the API

The Babbo Scanner Manager software development kit (SDK) is intended for developers interested in creating or modifying Palm 68k application for Babbo scanner devices. The Babbo Scanner Manager shared library API provides mechanism to control and receive data from the barcode scanner.

Any applications that using the Babbo Scanner Manager shared library can use the following steps to perform barcode scanning.

1. Open the scanner library
2. Enable and initiate the scanner
3. Handle returned barcode data or error messages
4. Close the scanner library

There is a sample application, ScanDemo, provided with this SDK package. Please refer to Section 8: Sample Demo Application for a detail explanation.

For a typical third-party application to use Babbo Scan Manager Library:

1. Add the following headers in the "include" sections:

```
#include "ScanMgrDef.h"                    // scan manager constants
#include "ScanMgrStruct.h"                 // scan manager structures
#include "ScanMgr.h"                       // scan manager API functions
```

2. Add the *ScnOpenDecoder* and *ScnCmdScanEnable* functions in the StartApplication function.

```
static void StartApplication(void)
{
        Err error;
        error = ScnOpenDecoder();          //Open the scanner library
        if (!error)
        {
                bOpenDecoderOK = true;     // set global variable that all is well
                ScnCmdScanEnable();        // Enable scanning
        }
        else
                        FrmAlert( OpenFailedAlert );
        }
        // Initialize and draw the main form.
        FrmGotoForm( MainForm );
}
```

3. Add the ScnCmdScanDisable and ScnCloseDecoder functions in the StopApplication function.

```
static void StopApplication(void)
{
	UInt16		cardNo;
	LocalID		dbID;

	// check to be sure that we earlier opened the decoder successfully
	if (bOpenDecoderOK)
	{
		// Disable the scanner, and then close the decoder.
		ScnCmdScanDisable();

		// Close the decoded
		ScnCloseDecoder();

		SysCurAppDatabase (&cardNo, &dbID);
	}
}
```

4. Add the scanDecodeEvent in your form handler function to receive the scan data

```
static Boolean MainFormHandleEvent(EventPtr event)
{
	Boolean		bHandled = false;
	Int16		ret;
	UInt16		cardNo;
	LocalID		dbID;
	UInt16 extendedDataFlag;

	switch( event->eType )
	{
		case frmOpenEvent:
			//form open event
			break;

		case frmUpdateEvent:
			//form update event
			break;

		case scanDecodeEvent:
			//Handle the Scan Decode Event

…
```

5. Add the ScnCmdStartDecode function to start barcode scanning.

```
case ctlSelectEvent:
{
        switch(event->data.ctlEnter.controlID)
        {
            // Scan Button
            case MainSCANButton:
                ret = ScnCmdStartDecode();
                bHandled = true;
                break;
…

case keyDownEvent:

    if (((event->data.keyDown.chr & vchrScanKey) &&
        (event->data.keyDown.keyCode == scanBitHard1)) ||
            ((event->data.keyDown.chr & vchrScanKey) &&
        (event->data.keyDown.keyCode == scanBitHard2)))
        {
            ret = ScnCmdStartDecode();
            bHandled = true;
        }
            break;
    …
```

Most barcode symbologies are disabled by default. They can be enable/disable by using the ScnSetBarcodeEnabled function. Please refer to the section "Decoder Functions" for its usage. Symbologies have extra options such as check digit can be set by using the ScnCmdWriteSymbologyConfig function. For backward compatilbe issue, some functions in "Decoder Functions" such as ScnSetI2of5CheckDigitVerification are also provided although same action can be performed by ScnCmdWriteSymbologyConfig with corresponding parameters.

The following section will describe all functions except extended functions that provided by the Babbo Scanner Manager Library.

Supported Barcodes

Symbologies	Options
Codabar	CHK_ENA|CHK_XMIT|STRT_STP|CONCAT|CONCATREQ
Code39	CHK_ENA|F_CHK_XMIT|F_STRT_STP|F_APPEND|F_FULLASC|F_BASE32
Int2of5	CHK_ENA|F_CHK_XMIT
Code93	
Code2of5	
IATA2of5	
Matrix2of5	
Code11	2CHKDIG
Code128	ISBTCONCAT
Telepen	OLDSTYLE
UPC_A	CHK_XMIT|NUMSYSXMIT|2DIGADDA|5DIGADDA|ADDAREQ|ADDASEP
CouponCode	
UPCE0	CHK_XMIT|EXPAND|NUMSYSXMIT|2DIGADDA|5DIGADDA|ADDAREQ|ADDASEP
UPCE1	
EANJAN13	CHK_XMIT|ADDAREQ|ADDASEP|2DIGADDA|5DIGADDA|ISBN
EANJAN8	CHK_XMIT|ADDAREQ|ADDASEP|2DIGADDA|5DIGADDA
MSI	CHK_XMIT
Plessey	
RSS14	
RSSLtd	
RSSExp	
PosiCode	
TriopticCode	
CodablockF	
Code16K	
Code49	
PDF417	
MicroPDF417	
Composite	
TLC39	
USPS4CB	
IDTAG	
Postnet	CHK_XMIT
PlanetCode	CHK_XMIT
BritishPost	
CanadaPost	
KixPost	
AusPost	
JapanPost	
ChinaPost	
KoreaPost	
QRCode	
DataMatrix	
MaxiCode	
AztecCode	AZTECRUNE

Option Code	Explanation
CHK_ENA	Enable usage of check character
CHK_XMIT	Send check character
STRT_STP	Include the start and stop characters in result string
APPEND	Code39 append mode
FULLASC	Enable Code39 Full ASCII
CONCAT	Codabar concatenate.
2CHKDIG	2 Check Digits
OLDSTYLE	Telepen Old Style mode.
EXPAND	Extended UPC-E
NUMSYSXMIT	UPC-A/UPC-e Send Num Sys
2DIGADDA	Enable 2 digit Addenda (UPC & EAN)
5DIGADDA	Enable 5 digit Addenda (UPC & EAN)
ADDAREQ	Only allow codes with addenda (UPC & EAN)
ADDASEP	Include Addenda separator space in returned string.
AZTECRUNE	Enable Aztec Run
UPCCOMP	Enable UPC/EAN
CONCATREQ	Codabar concatenate.
ISBTCONCAT	ISBT Concat
BASE32	Code 32 conv
ISBN	Convert ISBN

Supported OCR Fonts

OCR_A
OCR_B
OCR_MONEY

Core Functions

Babbo Scanner Manager Core Function Summary

The following is a summary of the core API functions.

Core Function	Summary Description	Page
ScnOpenDecoder	Initiates a connection to the Babbo scan manager library	13
ScnCloseDecoder	Terminates the connection with the Babbo scan manager library	13
ScnCmdSendParams	Transmits the scanner parameter values set by other functions except extended scanner functions	13
ScnCmdGetAllParams	Not Supported	
ScnGetScanManagerVersion	Retrieves the scanner manager version number	14
ScnGetScanPortDriverVersion	Retrieves the scanner port version number	14
ScnGetDecoderVersion	Retrieves the decoder version number	15
ScnGetDecodedData	Retrieves the barcode data of last decode attempt	15
ScnGetExtendedDecodedData	Retrieves the barcode data (larger than 255 bytes) of last decode attempt	16
ScnCmdScanEnable	Enables scanning functions	16
ScnCmdScanDisable	Disables scanning functions	17
ScnCmdStartDecode	Starts barcode scan and decode	17
ScnCmdStopDecode	Stops any decode attempt	17
ScnCmdBeep	Executes the beep sound sequence	18
ScnCmdParamDefaults	Sets all parameter to the factory default values	18
ScnCmdLedOn	Turns on the BABBO Green LED	18
ScnCmdLedOff	Turns of the BABBO Green LED	19
ScnGetLedState	Retrieves the state of the BABBO Green LED	19
ScnCmdAimOn	Turns on the scanner's aimer LED during scanning *(Recompile Required)*	19
ScnCmdAimOff	Turns off the scanner's aimer LED during scanning *(Recompile Required)*	20
ScnGetAimMode	Retrieves the aimer mode *(Recompile Required)*	20

The following is a listing of each core scanning function with its complete description and prototype for each function.

ScnOpenDecoder

Purpose	Initiates a connection to the Babbo scan manager library.
Prototype	Int16 ScnOpenDecoder(void);
Parameters	None
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	The function initiates a connection to the scan engine. The application should call this function before any other API functions. Once an application has connected to the engine, all other API functions can be successfully called.
See Also	ScnCloseDecoder ()

ScnCloseDecoder

Purpose	Terminates the connection with the Babbo scan manager library
Prototype	Int16 ScnCloseDecoder(void);
Parameters	None
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	This function terminates the connection with the scan engine. Any resources used by the scan engine are freed.
See Also	ScnOpenDecoder ()

ScnCmdSendParams

Purpose	Transmits the scanner parameter values set by other functions *(Except the extended scanner functions)*
Prototype	Int16 ScnCmdSendParams(BeepType beep);
Parameters	beep Type of beep sound when execute this function
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	This function send parameter values set by other functions to the scanner manager library
See Also	None

ScnGetScanManagerVersion

Purpose	Retrieves the scanner manager version number
Prototype	Int16 ScnGetScanManagerVersion(Char * pszVer, UInt16 max_length);
Parameters	pszVer Upon successful return, this variable points to the version of scan manager library max_length Maximum length of characters
Return	STATUS_OK NOT_SUPPORTED
Comments	None
See Also	ScnGetScanPortDriverVersion ScnGetDecoderVersion

ScnGetScanPortDriverVersion

Purpose	Retrieves the scanner port version number
Prototype	Int16 ScnGetScanPortDriverVersion(Char * pszVer, UInt16 max_length);
Parameters	pszVer Upon successful return, this variable points to the version of scanner port driver max_length Maximum length of characters
Return	STATUS_OK NOT_SUPPORTED
Comments	None
See Also	ScnGetScanManagerVersion ScnGetDecoderVersion

ScnGetDecoderVersion

Purpose	Retrieves the decoder version number
Prototype	Int16 ScnGetDecoderVersion(Char * pszVer, UInt16 max_length);
Parameters	pszVer Upon successful return, this variable points to the version of decoder library max_length Maximum length of characters
Return	STATUS_OK NOT_SUPPORTED
Comments	None
See Also	ScnGetScanManagerVersion ScnGetScanPortDriverVersion

ScnGetDecodedData

Purpose	Retrieves the barcode data of last decode attempt
Prototype	Int16 ScnGetDecodedData(MESSAGE *ptr);
Parameters	ptr Upon successful return, this variable points to the decoded message data
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	This function is used to retrieve the results of the last decode attempt. It should be called in response to the notification that the decode attempt has finished.
See Also	ScnGetExtendedDecodedData

ScnGetExtendedDecodedData

Purpose	Retrieves the barcode data (larger than 255 bytes) of last decode attempt
Prototype	Int16 ScnGetExtendedDecodedData(Int16 length, Int16 *type, UInt8 *extendedData);
Parameters	length Size of the allocated buffer type Upon successful return, this variable points to the decoded data type extendedData Upon successful return, this variable points to the extended barcode data
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	This function is used to retrieve the results of the last decode attempt. It should be called in response to the notification that the decode attempt has finished and when extended data flag is set to indicate that extended data has been decoded.
See Also	ScnGetExtendedDecodedData

ScnCmdScanEnable

Purpose	Enables scanning functions
Prototype	Int16 ScnCmdScanEnable(void);
Parameters	None
Return	STATUS_OK COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	Permits the scanner to activate
See Also	ScnCmdScanDisable

ScnCmdScanDisable

Purpose	Disables scanning functions
Prototype	Int16 ScnCmdScanDisable(void);
Parameters	None
Return	STATUS_OK COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	Permits the scanner to activate
See Also	ScnCmdScanDisable

ScnCmdStartDecode

Purpose	Starts barcode scan and decode
Prototype	Int16 ScnCmdStartDecode(void);
Parameters	None
Return	STATUS_OK COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	This function is used to start a decode attempt by the scan engine. This function returns immediately to the calling application after starting the decode attempt process. The decode attempt process will continue to run in the background. When decode completed, "scanDecodeEvent" will be triggered.
See Also	ScnCmdStopDecode

ScnCmdStopDecode

Purpose	Stops any decode attempt
Prototype	Int16 ScnCmdStopDecode(void);
Parameters	None
Return	STATUS_OK COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	This function is used terminate a decode attempt.
See Also	ScnCmdStartDecode

ScnCmdBeep

Purpose	Executes the beep sound sequence immediately
Prototype	Int16 ScnCmdBeep(BeepType beep);
Parameters	beep the type of beep sound
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None.
See Also	None.

ScnCmdParamDefaults

Purpose	Sets all parameter to the factory default values
Prototype	Int16 ScnCmdParamDefaults(void);
Parameters	None
Return	STATUS_OK COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None.
See Also	None.

ScnCmdLedOn

Purpose	Turns on the Green LED
Prototype	Int16 ScnCmdLedOn(void);
Parameters	None
Return	STATUS_OK NOT_SUPPORTED
Comments	None.
See Also	ScnCmdLedOff ScnGetLedState

ScnCmdLedOff

Purpose	Turns of the Green LED
Prototype	Int16 ScnCmdLedOff(void);
Parameters	None
Return	STATUS_OK NOT_SUPPORTED
Comments	None.
See Also	ScnCmdLedOn ScnGetLedState

ScnGetLedState

Purpose	Retreives the state of the Green LED
Prototype	Int16 ScnGetLedState(void);
Parameters	None
Return	0 – OFF 1 – ON
Comments	None.
See Also	ScnCmdLedOn ScnCmdLedOff

ScnCmdAimOn

Purpose	Turns on the scanner's aimer LED during scanning
Prototype	Int16 ScnCmdAimOn(void);
Parameters	None
Return	STATUS_OK NOT_SUPPORTED
Comments	To use this function, application is required to recompile with the updated library provided in this SDK package. ScnCmdGetSetLightsMode can provide more controls
See Also	ScnCmdAimOff ScnGetAimMode ScnCmdGetSetLightsMode

ScnCmdAimOff

Purpose	Turns off the scanner's aimer LED during scanning
Prototype	Int16 ScnCmdAimOn(void);
Parameters	None
Return	STATUS_OK NOT_SUPPORTED
Comments	To use this function, application is required to recompile with the updated library provided in this SDK package ScnCmdGetSetLightsMode can provide more controls
See Also	ScnCmdAimOn ScnGetAimMode ScnCmdGetSetLightsMode

ScnGetAimMode

Purpose	Retrieves the aimer mode
Prototype	Int16 ScnGetAimMode(void);
Parameters	None
Return	0 – OFF 1 – ON
Comments	To use this function, application is required to recompile with the updated library provided in this SDK package ScnCmdGetSetLightsMode can provide more controls
See Also	ScnCmdAimOn ScnCmdAimOff ScnCmdGetSetLightsMode

Decoder Functions

Babbo Scanner Manager Decoder Function Summary

The following is a summary of the decoder API functions.

Decoder Function	Summary Description	Page
ScnSetBarcodeEnabled	Enables a specified symbology or all symbology	22
ScnGetBarcodeEnabled	Determines whether the specified symbology is enabled	22
ScnSetBarcodeLengths	Not Supported	
ScnGetBarcodeLengths	Not Supported	
ScnSetPrefixSuffixValues	Not Supported	
ScnGetPrefixSuffixValues	Not Supported	
ScnSetConvert	Converts decoded data to specified format	23
ScnGetConvert	Retrieves the conversion settings	23
ScnSetTransmitCheckDigit	Enables the specified code check digit	24
ScnGetTransmitCheckDigit	Retrieves the check digit settings	24
ScnSetNotisEditing	Strips the start and stop characters from a 14-character Codabar symbol	25
ScnGetNotisEditing	Retrieves the Notis editing setting	25
ScnSetUpcPreamble	Set whether the specified UPC code is to be transmitted with lead-in characters	26
ScnGetUpcPreamble	Retrieves the UPC Preamble setting	26
ScnSetEanZeroExtend	Sets leading zeros option to decoded EAN-8 symbols	27
ScnGetEanZeroExtend	Retrieves the EAN zero extend setting	27
ScnSetCode32Prefix	Set the Code32 Prefix setting	27
ScnGetCode32Prefix	Retrieves the Code32 Prefix setting	28
ScnSetCode39CheckDigitVerification	Sets Code39 Check Digit option	28
ScnGetCode39CheckDigitVerification	Reads Code39 Check Digit Verification setting	28
ScnSetCode39FullAscii	Sets Code39 Full ASCII Support option	29
ScnGetCode39FullAscii	Reads Code39 Full ASCII Support setting	29
ScnSetI2of5CheckDigitVerification	Sets I2of5 Check Digit Verification option	29
ScnGetI2of5CheckDigitVerification	Reads I2of5 Check Digit Verification setting	30
ScnSetMsiPlesseyCheckDigits	Not Supported	
ScnGetMsiPlesseyCheckDigits	Not Supported	
ScnSetMsiPlesseyCheckDigitAlgorithm	Not Supported	
ScnGetMsiPlesseyCheckDigitAlgorithm	Not Supported	
ScnSetClsiEditing	Not Supported	
ScnGetClsiEditing	Not Supported	
ScnSetUpcEanSecurityLevel	Not Supported	
ScnGetUpcEanSecurityLevel	Not Supported	
ScnSetDecodeUpcEanSupplementals	Not Supported	
ScnGetDecodeUpcEanSupplementals	Not Supported	
ScnSetDecodeUpcEanRedundancy	Not Supported	
ScnGetDecodeUpcEanRedundancy	Not Supported	

The following is a listing of each decoder function with its complete description and prototype for each function.

ScnSetBarcodeEnabled

Purpose	Enables a specified symbology or all symbology
Prototype	Int16 ScnSetBarcodeEnabled(BarType barcodeType, Boolean bEnable);
Parameters	barcodeType One of the symbology enumerated types bEnable True to enable, false to disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	This function enables or disables a specified symbology or all symbologies from decoding.
See Also	ScnGetBarcodeEnabled ()

ScnGetBarcodeEnabled

Purpose	Determines whether the specified symbology is enabled
Prototype	Int16 ScnGetBarcodeEnabled(BarType barcodeType);
Parameters	barcodeType One of the symbology enumerated types
Return	0 – Disabled 1 – Enabled
Comments	This function determines if the given symbology is enabled or not
See Also	ScnSetBarcodeEnabled ()

ScnSetConvert

Purpose	Converts decoded data to specified format
Prototype	Int16 ScnSetConvert(ConvertType conversion, Boolean bEnable);
Parameters	conversion One of the conversion enumerated types bEnable True to enable, false to disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	This function only support: CONVERT_CODE39_TO_CODE32 CONVERT_UPCE_TO_UPCA CONVERT_EAN8_TO_EAN13
See Also	ScnGetConvert ()

ScnGetConvert

Purpose	Retrieves the conversion settings
Prototype	Int16 ScnGetConvert(ConvertType conversion);
Parameters	conversion One of the conversion enumerated types
Return	0 – Disabled 1 – Enabled
Comments	This function only support: CONVERT_CODE39_TO_CODE32 CONVERT_UPCE_TO_UPCA CONVERT_EAN8_TO_EAN13
See Also	ScnSetConvert ()

ScnSetTransmitCheckDigit

Purpose	Enables the specified code check digit
Prototype	Int16 ScnSetTransmitCheckDigit(BarType barType, UInt16 check_digit);
Parameters	barcodeType One of the symbology enumerated types check_digit 1 to enable; 0 to disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	This function only support: TRANSMIT_CODE39_CHECK_DIGIT TRANSMIT_I2OF5_CHECK_DIGIT TRANSMIT_UPCA_CHECK_DIGIT TRANSMIT_UPCE_CHECK_DIGIT TRANSMIT_UPCE1_CHECK_DIGIT TRANSMIT_MSI_PLESSEY_CHECK_DIGIT
See Also	ScnGetTransmitCheckDigit ()

ScnGetTransmitCheckDigit

Purpose	Retrieves the check digit settings
Prototype	Int16 ScnGetTransmitCheckDigit(BarType barType);
Parameters	barcodeType One of the symbology enumerated types
Return	0 – Disabled 1 – Enabled
Comments	This function will support: TRANSMIT_CODE39_CHECK_DIGIT TRANSMIT_I2OF5_CHECK_DIGIT TRANSMIT_UPCA_CHECK_DIGIT TRANSMIT_UPCE_CHECK_DIGIT TRANSMIT_UPCE1_CHECK_DIGIT TRANSMIT_MSI_PLESSEY_CHECK_DIGIT
See Also	ScnSetTransmitCheckDigit ()

ScnSetNotisEditing

Purpose	Strips the start and stop characters from a 14-character Codabar symbol
Prototype	Int16 ScnSetNotisEditing(Boolean bEnable);
Parameters	bEnable True to enable, false to disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None
See Also	ScnGetNotisEditing()

ScnGetNotisEditing

Purpose	Retrieves the Notis editing setting
Prototype	Int16 ScnGetNotisEditing(void);
Parameters	None
Return	0 – Disabled 1 – Enabled
Comments	None
See Also	ScnSetNotisEditing()

ScnSetUpcPreamble

Purpose	Sets whether the specified UPC code is to be transmitted with lead-in characters
Prototype	Int16 ScnSetUpcPreamble(BarType barcodeType, Int16 preamble);
Parameters	barcodeType One of the symbology enumerated types preamble NO_PREAMBLE SYSTEM_CHARACTER SYSTEM_CHARACTER_COUNTRY_CODE
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None
See Also	ScnGetUpcPreamble ()

ScnGetUpcPreamble

Purpose	Retrieves the UPC Preamble setting
Prototype	Int16 ScnGetUpcPreamble(BarType barcodeType);
Parameters	barcodeType One of the symbology enumerated types
Return	0 – Disabled 1 – Enabled
Comments	None
See Also	ScnSetUpcPreamble ()

ScnSetEanZeroExtend

Purpose	Sets leading zeros option to decoded EAN-8 symbols
Prototype	Int16 ScnSetEanZeroExtend(Boolean bEnable);
Parameters	bEnable True to enable, false to disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None
See Also	ScnGetEanZeroExtend ()

ScnGetEanZeroExtend

Purpose	Retrieves the EAN zero extend setting
Prototype	Int16 ScnGetEanZeroExtend(void);
Parameters	None
Return	0 – Disabled 1 – Enabled
Comments	None
See Also	ScnSetEanZeroExtend ()

ScnSetCode32Prefix

Purpose	Sets Code32 Prefix option
Prototype	Int16 ScnSetCode32Prefix(Boolean bEnable);
Parameters	bEnable true – enable false - disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	Pre-pending a Code32 data with the capital letter 'A'
See Also	ScnGetCode32Prefix ()

ScnGetCode32Prefix

Purpose	Retrieves the Code32 Prefix setting
Prototype	Int16 ScnGetCode32Prefix(void);
Parameters	None
Return	0 – Disabled 1 – Enabled
Comments	None
See Also	ScnSetCode32Prefix

ScnSetCode39CheckDigitVerification

Purpose	Sets Code39 Check Digit option
Prototype	Int16 ScnSetCode39CheckDigitVerification(UInt16 check_digit);
Parameters	check_digit 1 – enable 0 - disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None
See Also	ScnGetCode39CheckDigitVerification

ScnGetCode39CheckDigitVerification

Purpose	Reads Code39 Check Digit Verification setting
Prototype	Int16 ScnGetCode39CheckDigitVerification(void);
Parameters	None
Return	0 – Disabled 1 – Enabled
Comments	None
See Also	ScnSetCode39CheckDigitVerification

ScnSetCode39FullAscii

Purpose	Sets Code39 Full ASCII Support option
Prototype	Int16 ScnSetCode39FullAscii(Boolean bEnable);
Parameters	bEnable true – enable false - disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None
See Also	ScnGetCode39FullAscii

ScnGetCode39FullAscii

Purpose	Reads Code39 Full ASCII Support setting
Prototype	Int16 ScnGetCode39FullAscii(void);
Parameters	None
Return	0 – Disabled 1 – Enabled
Comments	None
See Also	ScnSetCode39FullAscii

ScnSetI2of5CheckDigitVerification

Purpose	Sets I2of5 Check Digit Verification option
Prototype	Int16 ScnSetI2of5CheckDigitVerification (UInt16 check_digit);
Parameters	check_digit 1 – enable 0 - disable
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None
See Also	ScnGetI2of5CheckDigitVerification

ScnGetI2of5CheckDigitVerification

Purpose	Reads I2of5 Check Digit Verification setting
Prototype	Int16 ScnGetI2of5CheckDigitVerification(void);
Parameters	None
Return	0 – Disabled 1 – Enabled
Comments	None
See Also	ScnSetI2of5CheckDigitVerification

Scanner Functions

Babbo Scanner Manager Scanner Function Summary

The following is a summary of the scanner API functions.

Scanner Function	Summary Description	Page
ScnIsJanamUnit	Checks if the Janam device exist	32
ScnSetDecodeLedOnTime	Set the Green LED on time upon successful scan	32
ScnGetDecodeLedOnTime	Get the duration setting of the Green LED	32
ScnSetBeepFrequency	Set the Beep frequency upon successful scan	33
ScnSetBeepDuration	Set the Beep duration upon successful scan	33
ScnSetBeepAfterGoodDecode	Enables Beep sound after success decode	33
ScnSetTriggeringModes	Set the Single/Continuous Triggering Mode	34
ScnGetTriggeringModes	Get the Single/Continuous Triggering Mode	34
ScnSetAngle	Not Supported	
ScnGetAngle	Not Supported	
ScnSetAimDuration	Not Supported	
ScnGetAimDuration	Not Supported	
ScnSetTimeOutBetweenSameSymbol	Not Supported	
ScnGetTimeOutBetweenSameSymbol	Not Supported	
ScnSetLinearCodeTypeSecurityLevel	Not Supported	
ScnGetLinearCodeTypeSecurityLevel	Not Supported	
ScnSetBidirectionalRedundancy	Not Supported	
ScnGetBidirectionalRedundancy	Not Supported	
ScnSetTransmitCodeIdCharacter	Not Supported	
ScnGetTransmitCodeIdCharacter	Not Supported	
ScnSetHostSerialResponseTimeOut	Not Supported	
ScnGetHostSerialResponseTimeOut	Not Supported	
ScnSetScanDataTransmissionFormat	Not Supported	
ScnGetScanDataTransmissionFormat	Not Supported	
ScnCmdTrigSledOn	Not Supported	
ScnCmdTrigSledOff	Not Supported	
ScnGetTrigSledMode	Not Supported	
ScnSetLaserOnTime	Not Supported	
ScnGetLaserOnTime	Not Supported	

The following is a listing of each decoder function with its complete description and prototype for each function.

ScnIsJanamUnit

Purpose	Checks if the Janam device exist
Prototype	Int16 ScnIsJanamUnit(void);
Parameters	None
Return	0 – Not Present 1 – Present
Comments	None
See Also	None

ScnSetDecodeLedOnTime

Purpose	Sets the Green LED on time upon successful scan
Prototype	Int16 ScnSetDecodeLedOnTime(UInt16 led_on_time);
Parameters	led_on_time [0…99] 0.0 to 9.9 seconds
Return	STATUS_OK BAD_PARAM COMMUNICATIONS_ERROR NOT_SUPPORTED
Comments	None
See Also	ScnGetDecodeLedOnTime()

ScnGetDecodeLedOnTime

Purpose	Gets the duration setting of the Green LED
Prototype	Int16 ScnGetDecodeLedOnTime(void);
Parameters	led_on_time [0…99] 0.0 to 9.9 seconds
Return	Integer in the range 0 to 99, representing 0.0 to 9.9 seconds.
Comments	None
See Also	ScnGetDecodeLedOnTime()

ScnSetBeepFrequency

Purpose	Sets the Beep frequency upon successful scan
Prototype	Int16 ScnSetBeepFrequency(FrequencyType type, Int16 beep_freq);
Parameters	type One of the frequency enumerated types beep_freq Beep frequency in Hertz (Hz)
Return	STATUS_OK BAD_PARAM
Comments	None
See Also	ScnSetBeepDuration ()

ScnSetBeepDuration

Purpose	Sets the Beep duration upon successful scan
Prototype	Int16 ScnSetBeepDuration(DurationType type, Int16 beep_duration);
Parameters	type One of the frequency enumerated types beep_duration Beep duration in milliseconds (ms)
Return	STATUS_OK BAD_PARAM
Comments	None
See Also	ScnSetBeepFrequency ()

ScnSetBeepAfterGoodDecode

Purpose	Enables/disables Beep sounds after successful scan
Prototype	Int16 ScnSetBeepAfterGoodDecode(Boolean bEnableBeep);
Parameters	bEnableBeep
Return	STATUS_OK BAD_PARAM
Comments	None
See Also	ScnSetBeepFrequency ()

ScnSetTriggeringModes

Purpose	Set the Single/Continuous Triggering Mode
Prototype	Int16 ScnSetTriggeringModes(UInt16 triggering_mode);
Parameters	triggering_mode LEVEL – Single triggering mode PULSE – Continuous triggering mode
Return	STATUS_OK BAD_PARAM
Comments	For single triggering mode, when users press scan button, scanner will stop scanning after successful scan or timeout For continues triggering mode, scanner will continuous scan until users release the scan button.
See Also	ScnGetTriggeringModes ()

ScnGetTriggeringModes

Purpose	Get the Single/Continuous Triggering Mode
Prototype	Int16 ScnGetTriggeringModes(void);
Parameters	Get the current triggering mode setting
Return	LEVEL – Single triggering mode PULSE – Continuous triggering mode
Comments	None
See Also	ScnSetTriggeringModes ()

Extended Functions

Babbo Scanner Manager Extended Function Summary

The following is a summary of the extended API functions.

The following is a listing of each extended function with its complete description and prototype for each function.

ScnCmdReadSymbologyConfig

Purpose	Reads configuration item for a given symbology
Prototype	Int16 ScnCmdReadSymbologyConfig(UInt8 Query, Int16 nSymId, SymFlagsRange_t *pSymStruct);
Parameters	Query Use QT_CURRENT for the current settings, or QT_DEFAULT for the default settings. nSymId One of the symbology enumerated types pSymStruct Pointer to the appropriate structure based on symbolgy ID. The structure will be either a SymFlagsOnly_t or a SymFlagsRange_t
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	This function reads configuration items for a given symbology. Individual items to be read are specified by adding the appropriate mask bit (OR'd) to the mask member of the structure to which it belongs. Only items whose mask bits are set are read, all other items are ignored.
See Also	ScnCmdWriteSymbologyConfig

ScnCmdWriteSymbologyConfig

Purpose	Writes configuration items for a given symbology
Prototype	Int16 ScnCmdWriteSymbologyConfig(Int16 nSymId, SymFlagsRange_t *pvSym);
Parameters	nSymId One of the symbology enumerated types pSymStruct Pointer to the appropriate structure based on symbolgy ID. The structure will be either a SymFlagsOnly_t or a SymFlagsRange_t
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	This function writes configuration items for a given symbology. Individual items to be written are specified by adding the appropriate mask bit (OR'd) to the mask member of the structure to which it belongs. Only items whose mask bits are set are written, all other items are ignored.are ignored.
See Also	ScnCmdReadSymbologyConfig

ScnCmdSetupOCR

Purpose	Sets the OCR configuration settings
Prototype	Int16 ScnCmdSetupOCR(UInt8 nFont, char* pszTemplate, char* pszGroupG, char* pszGroupH, char* pszCheckChar);
Parameters	nFont This determines which OCR fonts (if any) are selected for decoding. The following values are used: OCR_DISABLED OCR_A OCR_B OCR_MONEY pszTemplate A null-terminated string that indicates one or more template patterns for the OCR decode. All characters in the font are matched as is, except for the following: a - alphanumeric character c - check character d - a digit from 0 - 9 e - any character g - any character specified in group G h - any character specified in group H l - alphabetic letter r - delimits a row t - delimits multiple templates pzGroupG A null-terminated string that defines the set of characters matching group "g" in a template. pszGroupH A null-terminated string that defines the set of characters matching group "h" in a template. pzCheckChar A null-terminated string that defines the legal characters for checksum computation in a decoded message. Use the string constant "0123456789" for modulo 10 checksums, and the string constant "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ" for modulo 36 checksums.
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	None
See Also	ScnCmdGetSetupOCR

ScnCmdGetSetupOCR

Purpose	Retrieves the OCR configuration settings
Prototype	Int16 ScnCmdGetSetupOCR(UInt8 QueryType, UInt8* pnFont, char* pszTemplate, char* pszGroupG, char* pszGroupH, char* pszCheckChar);
Parameters	QueryType Use QT_CURRENT for the current settings, or QT_DEFAULT for the default settings. pnFont This determines which OCR fonts (if any) are selected for decoding. The following values are used: OCR_DISABLED OCR_A OCR_B OCR_MONEY pszTemplate A null-terminated string that indicates one or more template patterns for the OCR decode. All characters in the font are matched as is, except for the following: a - alphanumeric character c - check character d - a digit from 0 - 9 e - any character g - any character specified in group G h - any character specified in group H l - alphabetic letter r - delimits a row t - delimits multiple templates pzGroupG A null-terminated string that defines the set of characters matching group "g" in a template. pszGroupH A null-terminated string that defines the set of characters matching group "h" in a template. pzCheckChar A null-terminated string that defines the legal characters for checksum computation in a decoded message. Use the string constant "0123456789" for modulo 10 checksums, and the string constant "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ" for modulo 36 checksums.
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	None
See Also	ScnCmdGetSetupOCR()

ScnCmdAimerOn

Purpose	Turns on/off the aimer LED immediately
Prototype	Int16 ScnCmdAimerOn(Int8 bEnable);
Parameters	bEnable TRUE turns the aimer on, FALSE turns it off.
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	This function controls the operation of the engine's aimer mechanism.
See Also	ScnCmdLightsOn()

ScnCmdLightsOn

Purpose	Turns on/off the illum LED immediately
Prototype	Int16 ScnCmdLightsOn(Int8 bEnable);
Parameters	bEnable TRUE turns the aimer on, FALSE turns it off.
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	This function controls the operation of the engine's illumination LED's.
See Also	ScnCmdLightsOn()

ScnCmdGetSetLightsMode

Purpose	Sets/retrieves the aimer - illumination combination
Prototype	Int16 ScnCmdGetSetLightsMode(Int8 Set, UInt8 QueryType, UInt8 *pLightsMode);
Parameters	Set TRUE, the lights mode will be set FALSE, the lights mode will be returned QueryType QT_CURRENT to get or set the current setting QT_DEFAULT to get or set the default setting pLightsMode Pointer to a LightsMode_t variable used to pass the lights mode value.
Return	STATUS_OK COMMUNICATIONS_ERROR
Comments	This function is used to set and retrieve the aimer - illumination combination to be used during scanning.
See Also	None

ScnCmdScanWedge

Purpose	On/Off ScanWedge
Prototype	void ScnCmdScanWedge(Boolean state);
Parameters	state TRUE, ScanWedge will be ON FALSE, ScanWedge will be OFF
Return	NONE
Comments	None
See Also	ScnCmdScanWedgeInsertChar

ScnCmdScanWedgeInsertChar

Purpose	ScanWedge Insert Character setting
Prototype	void ScnCmdScanWedgeInsertChar(UInt8 char0, UInt8 char1, UInt8 char2, UInt8 char3)
Parameters	Char0 – 1st Character/Barcode Char1 – 2nd Character/Barcode Char2 – 3rd Character/Barcode Char3 – 4th Character/Barcode With the following defines: BARCODE CARRIGE_RETURN LINE_FEED TEMINATE
Return	NONE
Comments	For example, if you want ScanWedge insert barcode and character in the following order: Barcode, Line Feed, Carrige Return, you can do it by: ScnCmdScanWedgeInsertChar(BARCODE, LINE_FEED, CARRIGE_RETURN, TEMINATE);
See Also	ScnCmdScanWedge

ScnCmdGetSetExposureMode

Purpose	set or get current exposure mode
Prototype	Int16 ScnCmdGetSetExposureMode(Int8 Set, Int8 *pExpMode)
Parameters	Set – If TRUE, then the value of pExpMode will be used to set the exposure mode, otherwise the current exposure mode will be stored in pExpMode. pExpMode – pointer to HHPScanDriverAutoExposureMode_t value that specifies the exposure mode
Return	NONE
Comments	Sets or gets the current exposure mode. The exposure mode chooses which type of exposure system is used, including fixed manual exposure (no automatic control), image engine on-chip auto exposure, and Hand Held Products proprietary soft-auto exposure, which usually gives the optimal results for decoding in mixed lighting conditions. In fixed lighting conditions, fixed exposure may provide some improvement to read time.
See Also	ScnCmdGetSetExposureSettings

ScnCmdGetSetExposureSettings

Purpose	Set or get parameters that control auto exposure
Prototype	Int16 ScnCmdGetSetExposureSettings(Int8 Set, Int32 * pArray, Int32 dwArrayLength)
Parameters	Set – If TRUE, then the values in pArray will be set into the scan driver as the current exposure settings. If FALSE, the current requested exposure settings will be stored in pArray. pArray – pointer to an array dwArrayLength – The length of pArray
Return	NONE
Comments	DEC_ES_EXPOSURE_METHOD The auto-exposure method to use. Must be one of: DEC_AE_METHOD_UNIFORM The auto exposure system uses the entire image and gives equal weight to all areas in determing the current exposure. DEC_AE_METHOD_CENTER_ONLY The auto exposure system uses only a center region of the image in order to determine the current exposure.

	DEC_AE_METHOD_CENTER_WEIGHTED The auto exposure system uses the entire image but gives increasing weight towards the center of the image in determing the current image exposure. This is the default, and provides best results for typical conditions. DEC_ES_TARGET_VALUE The "target white point" for the image. This value sets what pixel value is considered a white pixel and the boundary of DEC_ES_TARGET_PERCENTILE. DEC_ES_TARGET_PERCENTILE The percent of pixels that must be have values below the target white point set by DEC_ES_TARGET_VALUE. DEC_ES_TARGET_ACCEPT_GAP The maximum delta that the actual white point may be from the specified one, in pixel counts. or example, a value of 40 with the default target white of 150, an image with a 97 percent white point as low as 110 or as high as 190 would be considered acceptable. DEC_ES_MAX_EXP The maximum exposure that may be used in capturing an image, measured in units of 127 µs at 30 frames per second. DEC_ES_MAX_GAIN The maximum multiplicative analog gain factor that may be used in capturing an image. In general, the higher the gain, the more noise will be present in the captured image. DEC_ES_FRAME_RATE The effective frame rate of the image engine. On the IT5x00 this will not change the actual frame rate, but rather should be set to the actual frame rate that the imager is running at. Typically this would be set only by hardware integrators. DEC_ES_CONFORM_IMAGE Whether the image must meet the acceptable exposure before being returned to the caller for processing or for decoding. For decoding this value should typically not be changed from its default. See DEC_ES_CONFORM_TRIES. When this setting is FALSE, all images captured will be sent to the caller. DEC_ES_CONFORM_TRIES The number of images to capture and discard that do not meet the exposure specifications. If an image that meets specifications is not captured after this many tries, it will be returned anyway.

	DEC_ES_SPECULAR_EXCLUSION The relative amount of high intensity pixels to discard in the image as being specularly reflected, ranging from 0 (assume no pixels are specular, specular exclusion is off) to 3 (assume many bright pixels are specular). The value 4 allows the specular exclusion to be controlled manually with the following two parameters. DEC_ES_SPECULAR_SAT The threshold pixel value for which pixels will be classified as specular. DEC_ES_SPECULAR_LIMIT A limiting control on the fraction of pixels that may be considered specular before the entire image is considered unusably overexposed. This value is a power-of-two divisor. For example, if this value is 3, then if more than 1/8th of the auto exposure sample points are marked specular, the image is marked overexposed. DEC_ES_FIXED_EXP The exposure to use when the exposure control is set to fixed exposure. This value is measured in units of 127µs at 30 frames per seconds. DEC_ES_FIXED_GAIN The multiplicative analog gain factor to be used when capturing an image when the exposure control is set to fixed exposure. In general, the higher the gain, the more noise will be present in the image. Default Exposure Settings for IT5x00 Parameter Default Value DEC_ES_EXPOSURE_METHDO DEC_AE_METHOD_CENTER_WEIGHTED DEC_ES_TARGET_VALUE 150 DEC_ES_TARGET_PERCENTILE 97 DEC_ES_TARGET_ACCEPT_GAP 40 DEC_ES_MAX_EXPOSURE 420 DEC_ES_CONFORM_IMAGE TRUE DEC_ES_CONFORM_TRIES 6 DEC_ES_SPECULAR_EXCLUSION 3 (MODERATE) DEC_ES_SPECULAR_SAT 0 DEC_ES_SPECULAR_LIMIT 3 DEC_ES_FIXED_EXP 240 DEC_ES_FIXED_GAIN 4

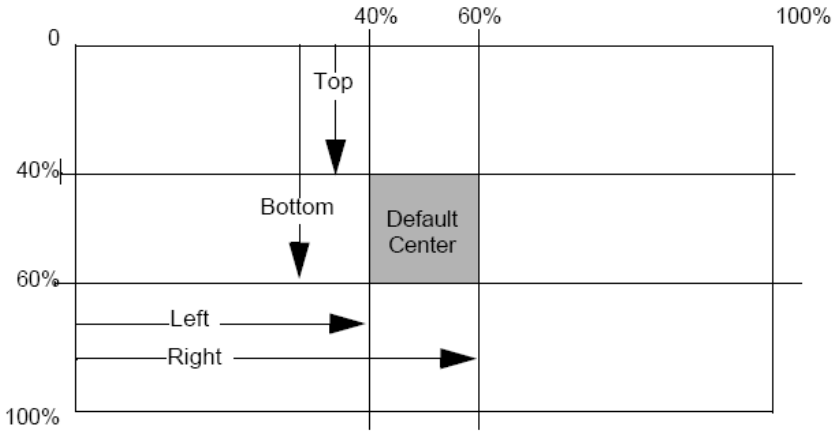
	Example:
	static Int32 expSettingsArray[] = {
	DEC_ES_EXPOSURE_METHOD, 0,
	DEC_ES_TARGET_VALUE, 0,
	DEC_ES_TARGET_PERCENTILE, 0,
	DEC_ES_TARGET_ACCEPT_GAP, 0,
	DEC_ES_MAX_EXP, 0,
	DEC_ES_MAX_GAIN, 0,
	DEC_ES_STARTING_EXP, 0,
	DEC_ES_STARTING_GAIN, 0,
	DEC_ES_CONFORM_IMAGE, 0,
	DEC_ES_CONFORM_TRIES, 0,
	DEC_ES_SPECULAR_EXCLUSION, 0,
	DEC_ES_SPECULAR_SAT, 0,
	DEC_ES_SPECULAR_LIMIT, 0, };
	ScnCmdGetSetExposureSettings (1, expSettingsArray,
	sizeof(expSettingsArray));

See Also	ScnCmdGetSetExposureMode

ScnCmdGetSetDecodeOptions

Purpose	Set or retrieve various decoder options that affect decode
Prototype	Int16 ScnCmdGetSetDecodeOptions(Int8 Set,UInt8 QueryType, DecodeOptions_t *pDecodeOptions)
Parameters	Set If TRUE, the values passed to pDecodeOptions should be used to set the decoder options, If FALSE, then the current decoder option values should be written to pDecodeOptions. QueryType When Set is FALSE, whether the current settings (QT_CURRENT), or the default settings should be returned (QT_DEFAULT). pDecodeOptions structure of type DecodeOptions_t that will receive or set the decoder options.
Return	NONE
Comments	Sets or retrieves, depending on the value of Set, various internal decoder options that are described in DecodeOptions_t. Most of these values have reasonable defaults, and should not be changed except for special operating circumstances. Example: DecodeOptions_t decopts; decopts.Version = DEC_DECODE_OPTIONS_VER; ScnCmdGetSetDecodeOptions (1, QT_CURRENT, &decopts);
See Also	None

ScnCmdGetSetDecodeCenteringWindow

Purpose	Set or retrieve imager's field of view
Prototype	Int16 ScnCmdGetSetDecodeCenteringWindow(Int8 bSet, UInt8 QueryType, Int8 *pbEnabled, RECT *pInt16ersectRect)
Parameters	bSet - If true, the call will change the current settings, if FALSE the settings will be queried. QueryType: When bSet is FALSE, if QT_CURRENT, the currently active settings will be returned, if QT_DEFAULT, the system defaults will be returned. When bSet is TRUE, if QT_CURRENT, the settings provided to pbEnabled and pIntersectRect will be programmed, if QT_DEFAULT then the system default values will be made active. pbEnabled: Value of 0 indicates decode centering is disabled. A value of 1 or 2 enables decode centering. A value of 1 specifies that the intersection rectangle is relative to the aimer center, while a value of 2 indicates the intersection rectangle is relative to the image center without regards to the aimer center. pIntersectRect: The rectangle that decoded barcodes must intersect. The values for the structure members are in percentages (0-100).
Return	Zero indicates success
Comments	The default centering window is a 128x96 pixel area in the center of the imager's field of view. The following diagram illustrates the default top, bottom, left, and right pixel positions, measured from the top and the left side of the imager's field of view, which is 640 by 480 pixels.  If a bar code is not within the predefined window, it will not be decoded or output by the imager.

	The following example shows how to change the current imager's field of view: Example: Int8 Enable = 1; RECT IntersectRect; IntersectRect.left = 0; IntersectRect.top = 0; IntersectRect.right = 50; IntersectRect.bottom = 100; ScnCmdGetSetDecodeCenteringWindow(true, QT_CURRENT, &Enable, &IntersectRect); ScnCmdSendParams(No_Beep);
See Also	None

Hardware Utility Library Functions

Babbo Hardware Utility Library Function Summary

The following is a summary of the Hardware Utility Library functions.

HwrUtilsLib Function	Summary Description	Page
HwrUtilsLibOpen	Open the Hardware Utility Library	
HwrUtilsLibClose	Close the Hardware Utility Library	
HwrUtilsLibGetKeyBacklight	Get the Key Backlight On/Off state	
HwrUtilsLibSetKeyBacklight	Set the Key Backlight On/Off state	
HwrUtilsLibGetBuzHiVol	Get the Buzzer High Volume On/Off state	
HwrUtilsLibSetBuzHiVol	Set the Buzzer High Volume On/Off state	
HwrUtilsLibGetVibrator	Get the Vibrator On/Off state	
HwrUtilsLibSetVibrator	Set the Vibrator On/Off state	
HwrUtilsLibGetRedLED	Get the Red LED On/Off state	
HwrUtilsLibSetRedLED	Set the Red LED On/Off state	
HwrUtilsLibGetGreenLED	Get the Green LED On/Off state	
HwrUtilsLibSetGreenLED	Set the Green LED On/Off state	
HwrUtilsLibGetBTState	Get the current Bluetooth state	
HwrUtilsLibSetBTState	Set the Bluetooth On/Off state	
HwrUtilsLibGetNumericKeyClickSnd	Get the current keyboard click sound state	
HwrUtilsLibSetNumericKeyClickSnd	Set the keyboard click sound state	

HwrUtilsLibOpen

Purpose	Open the Hardware Utility Library
Prototype	Err HwrUtilsLibOpen(UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	Zero if no error Non-zero if error
Comments	None
See Also	HwrUtilsLibClose

HwrUtilsLibClose

Purpose	Close the Hardware Utility Library
Prototype	Err HwrUtilsLibClose (UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	Zero if no error Non-zero if error
Comments	None
See Also	HwrUtilsLibClose

HwrUtilsLibGetKeyBacklight

Purpose	Get the Key Backlight On/Off state
Prototype	UInt32 HwrUtilsLibGetKeyBacklight(UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	0 if current state is OFF 1 if current state is ON
Comments	None
See Also	HwrUtilsLibSetKeyBacklight

HwrUtilsLibSetKeyBacklight

Purpose	Set the Key Backlight On/Off state
Prototype	void HwrUtilsLibSetKeyBacklight(UInt16 refnum, UInt32 value)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad value - 0: OFF; 1: ON
Return	None
Comments	None
See Also	HwrUtilsLibGetKeyBacklight

HwrUtilsLibGetBuzHiVol

Purpose	Get the Buzzer High Volume On/Off state
Prototype	UInt32 HwrUtilsLibGetBuzHiVol(UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	None
Comments	None
See Also	HwrUtilsLibSetBuzHiVol

HwrUtilsLibSetBuzHiVol

Purpose	Set the Buzzer High Volume On/Off state
Prototype	void HwrUtilsLibSetBuzHiVol(UInt16 refnum, UInt32 value)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad value - 0: OFF; 1: ON
Return	None
Comments	None
See Also	HwrUtilsLibGetBuzHiVol

HwrUtilsLibGetVibrator

Purpose	Get the Vibrator On/Off state
Prototype	UInt32 HwrUtilsLibGetVibrator(UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	None
Comments	None
See Also	HwrUtilsLibSetVibrator

HwrUtilsLibSetVibrator

Purpose	Set the Vibrator On/Off state
Prototype	void HwrUtilsLibSetVibrator(UInt16 refnum, UInt32 value)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad value - 0: OFF; 1: ON
Return	None
Comments	None
See Also	HwrUtilsLibGetVibrator

HwrUtilsLibGetRedLED

Purpose	Get the Red LED On/Off state
Prototype	UInt32 HwrUtilsLibGetRedLED(UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	None
Comments	None
See Also	HwrUtilsLibSetRedLED

HwrUtilsLibSetRedLED

Purpose	Set the Red LED On/Off state
Prototype	void HwrUtilsLibSetRedLED(UInt16 refnum, UInt32 value)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad value - 0: OFF; 1: ON
Return	None
Comments	None
See Also	HwrUtilsLibGetRedLED

HwrUtilsLibGetGreenLED

Purpose	Get the Green LED On/Off state
Prototype	UInt32 HwrUtilsLibGetGreenLED(UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	None
Comments	None
See Also	HwrUtilsLibSetGreenLED

HwrUtilsLibSetGreenLED

Purpose	Set the Green LED On/Off state
Prototype	void HwrUtilsLibSetGreenLED(UInt16 refnum, UInt32 value)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad value - 0: OFF; 1: ON
Return	None
Comments	None
See Also	HwrUtilsLibGetGreenLED

HwrUtilsLibGetBTState

Purpose	Get the current Bluetooth state
Prototype	UInt32 HwrUtilsLibGetBTState (UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	None
Comments	None
See Also	HwrUtilsLibSetBTState

HwrUtilsLibSetBTState

Purpose	Enable/Disable the bluetooth device
Prototype	void HwrUtilsLibSetBTState (UInt16 refnum, UInt32 value)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad value - 0: OFF; 1: ON
Return	None
Comments	None
See Also	HwrUtilsLibGetBTState

HwrUtilsLibGet5VOut

Purpose	Get the current 5V Out state
Prototype	UInt32 HwrUtilsLibGet5VOut (UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	0 if current state is OFF 1 if current state is ON
Comments	None
See Also	HwrUtilsLibSet5VOut

HwrUtilsLibSetBTState

Purpose	Enable/Disable the 5V out
Prototype	void HwrUtilsLibSet5VOut(UInt16 refnum, UInt32 value)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad value - 0: OFF; 1: ON
Return	None
Comments	None
See Also	HwrUtilsLibGet5VOut

HwrUtilsLibGetNumericKeyClickSnd

Purpose	Get the current keyboard click sound state
Prototype	UInt32 HwrUtilsLibGetNumericKeyClickSnd (UInt16 refnum)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad
Return	0 if current state is OFF 1 if current state is ON
Comments	None
See Also	HwrUtilsLibSetNumericKeyClickSnd

HwrUtilsLibSetNumericKeyClickSnd

Purpose	Enable/Disable the keyboard click
Prototype	void HwrUtilsLibSetNumericKeyClickSnd (UInt16 refnum, UInt32 value)
Parameters	refnum - The Palm library reference number returned by functions SysLibFind/ SysLibLoad value - 0: OFF; 1: ON
Return	None
Comments	None
See Also	HwrUtilsLibGetNumericKeyClickSnd

Application Porting

Porting applications to use the Babbo Scanner Manager Library

To modify applications using the Babbo Scanner Manager library, please follow the following steps:

1. Add the "JanamScnCompatible.h" header in front of all Scan Manager headers.

#include "JanamScnCompatible.h"

#include "ScanMgrDef.h"

#include "ScanMgrStruct.h"

#include "ScanMgr.h"

2. Verify if the unsupported functions affect your application refer to the function summary

3. If you need to add new feature such as OCR scanning which support by this SDK, please use the extended functions for configuration. *ScanDemo* and *Appendix A* provide examples to use OCR functions

4. Re-compile your application with the ScanMgr.lib provided by this SDK.

Sample Application

A typical application program uses the Babbo Scan Manager library to do the following:

- Open the scanner library
- Enable and initiate the scanner
- Handle returned barcode data or error messages
- Close the scanner library

In the following section, code segments of a simple sample application will be given.

A Simple Application Program Sample

Include Files

The following #include statement provides Scan Manager library interface definitions.

```
#include "ScanMgrDef.h"          // scan manager constants

#include "ScanMgrStruct.h"       // scan manager structures

#include "ScanMgr.h"             // scan manager API functions
```

AppStart Function

The AppStart() function shows what you need to do to initialize the barcode scanner. It will perform the following things:

- Open the scanner library
- Enable the scanner
- Add a variable to check if the library has open successfully
- Allocates memory if needed

```
static void AppStart(void)
{
	Err error;

	//Check if the BarCode Scanner Device is connected.
	if (ScnIsJanamUnit())
	{
		error = ScnOpenDecoder();  //Open the scanner library
		if (!error)
		{
			bOpenDecoderOK = true;
			ScnCmdScanEnable(); // Enable scanning
		}
		else
			FrmAlert( OpenFailedAlert );
	}
	else
		FrmAlert( WrongHardwareAlert );

	// Initialize and draw the main form.
	FrmGotoForm( MainForm );
}
```

HandleEvent Function

The MainFormHandleEvent() function is an event handler for main form. If there is barcode data ready for read, a scanDecodeEvent will be received. ScanGetDecodedData () should be called to read barcode data and display it. If the extendedData flag is set, ScanGetExtendedDecodedData() should also be called.

In MainFormHandleEvent ():

```
case scanDecodeEvent:
//A Scan Decode Event has performed
extendedDataFlag = ((ScanEventPtr)event)->scanData.scanGen.data1;
extendedDataLength  = (Int16)(((ScanEventPtr)event)->scanData.scanGen.data2);
if (extendedDataFlag & EXTENDED_DATA_FLAG)
       extendedData = true;
else
       extendedData = false;

OnDecoderData();

if (Auto_Scan == true)
{
       SysCurAppDatabase (&cardNo, &dbID);
       AlmSetAlarm (cardNo, dbID, 0, (TimGetSeconds()+time), 0);
}
bHandled = true;
break;
```

In OnDecoderData():

```
       Boolean OnDecoderData()
       {
              static Char BarTypeStr[300]=" ";
              MESSAGE      decodeDataMsg;
              Int16 status;


              if (extendedData == true){
                     hExtendedData = MemHandleNew(extendedDataLength+1);
                     pExtendedData = (UInt8 *) MemHandleLock( hExtendedData );
                     status = ScnGetExtendedDecodedData(extendedDataLength,
                     &extendedDataType, pExtendedData);
                     pExtendedData[g_ExtendedDataLength] = '\0';
              }
              else{
                     status = ScnGetDecodedData( &decodeDataMsg );
                     extendedDataType = decodeDataMsg.type;
                     extendedDataLength = decodeDataMsg.length;
                     hExtendedData = MemHandleNew(extendedDataLength+1);
                     pExtendedData = (UInt8 *) MemHandleLock( hExtendedData );
                     pExtendedData[extendedDataLength] = '\0';
                     MemMove( &pExtendedData[0], &decodeDataMsg.data[0],
                     extendedDataLength+1 );
              }
```

```
        if( status == STATUS_OK ) // if we successfully got the decode
        data from the API...
        {

               if( extendedDataType == 0){
                      SetFieldText(MainBarTypeField, "No Scan", 30, true);
                      SetFieldText(MainScandataField, "No Scan", 30, true);
                      MemHandleUnlock(hExtendedData);
                      MemHandleFree(hExtendedData);
               }
               else{

                      ScanGetBarTypeStr(extendedDataType, BarTypeStr, 30);
                      SetFieldText( MainBarTypeField, BarTypeStr, 30,
                      true);

                      // Place the barcode data into the field and display
                      SetFieldText(MainScandataField, (char
                      *)&pExtendedData[0], extendedDataLength+1, true);
                      MemHandleUnlock(hExtendedData);
                      MemHandleFree(hExtendedData);
               }
        }
        else
                      DbgMessage ("Decode Status not ok.\n");
        return(0);
}
```

AppStop Function

The AppStop() function is called at the end of PilotMain(). It

- Closes the Scan Manager library.
- Disable and power off the scanner device
- Frees the global handle if needed

```
static void AppStop(void)
{
	UInt16		cardNo;
	LocalID			dbID;

	// check to be sure that we earlier opened the decoder successfully
	if (bOpenDecoderOK)
	{
		// Disable the scanner, and then close the decoder.
		ScnCmdScanDisable();

		// Close the decoded
		ScnCloseDecoder();

		SysCurAppDatabase (&cardNo, &dbID);
		AlmSetAlarm (cardNo, dbID, 0, 0, 0);		// Clear any alarm
	}
}
```

Symbology Configuration

The SymbologyConfigSetControls() function is used to set options for each symbology

```
static void SymbologyConfigSetControls(int sel)
{
	FormType *frmP = FrmGetActiveForm();
	unsigned i, ckn;//, ndx;
	struct symConfigTblEnt *symConfigEnt;
	SymFlagsRange_t symSetRange;
	Boolean bEnabled = false;
	UInt32 testbit=0;
	ControlType *ckP[8];

	ckP[0] = (ControlPtr) GetObjectPtr ( SymbologyCheck0Checkbox );
	ckP[1] = (ControlPtr) GetObjectPtr ( SymbologyCheck1Checkbox );
	ckP[2] = (ControlPtr) GetObjectPtr ( SymbologyCheck2Checkbox );
	ckP[3] = (ControlPtr) GetObjectPtr ( SymbologyCheck3Checkbox );
	ckP[4] = (ControlPtr) GetObjectPtr ( SymbologyCheck4Checkbox );
	ckP[5] = (ControlPtr) GetObjectPtr ( SymbologyCheck5Checkbox );
	ckP[6] = (ControlPtr) GetObjectPtr ( SymbologyCheck6Checkbox );
	ckP[7] = (ControlPtr) GetObjectPtr ( SymbologyCheck7Checkbox );

	//Read the Symbology setting for current selection
	symConfigEnt = &symConfigTbl[sel];
	if (symConfigEnt->txtTblBits & F_MINMAX)
		symSetRange.dwStructSize = sizeof(SymFlagsRange_t);
	else
		symSetRange.dwStructSize = sizeof(SymFlagsOnly_t);

	symSetRange.dwFlags = 0;
	symSetRange.dwMask = SYM_MASK_ALL;
	symConfigEnt = &symConfigTbl[sel];

	//add to chk if enabled
	ScanCmdReadSymbologyConfig(QT_CURRENT, symConfigEnt->symID, &symSetRange);

	if (symConfigEnt->txtTblBits & F_MINMAX)

	{
		MemHandle		fldHandle;
		FieldType *fldP;
		char *preText;
		fldP = (FieldPtr)
		GetObjectPtr( SymbologySymbologyConfigFormMinFldField );
		fldHandle = FldGetTextHandle(fldP);
		preText = (Char * ) MemHandleLock(fldHandle);
		symSetRange.dwMinLen = StrAToI(preText);
		MemHandleUnlock(fldHandle);

		fldP = (FieldPtr)
		GetObjectPtr( SymbologySymbologyConfigFormMaxFldField );
		fldHandle = FldGetTextHandle(fldP);
```

```
            preText = (Char * ) MemHandleLock(fldHandle);
            symSetRange.dwMaxLen = StrAToI(preText);
            MemHandleUnlock(fldHandle);

        }
        ckn=0;
        for (i = 0; i <= F_FLAGMAX; ++i)
        {
            //int ndx;
            if (!(symConfigEnt->txtTblBits & (PrvSymbolConvert(i))))
                continue;

            if (CtlGetValue(ckP[ckn]))
                symSetRange.dwFlags |= (symFlagTbl[i].flagBit);
            else
                symSetRange.dwFlags &= ~(symFlagTbl[i].flagBit);
            ckn++;
        }
        //Send the Command to ScanCmd Library
        ScanCmdWriteSymbologyConfig(symConfigEnt->symID, &symSetRange);
}
```

OCR Configuration

The OCRConfigFormSaveConfig () function is used to set options for OCR scanning

```
static void OCRConfigFormSaveConfig()
{
       FieldType *fldP;
       ListType *listP;
       char *pTemplate, *pGSet, *pHSet, *pCheck;
       UInt16 font;
       fldP = (FieldPtr) GetObjectPtr( OCROCRConfigFormTemplateField );
       pTemplate = FldGetTextPtr(fldP);

       fldP = (FieldPtr) GetObjectPtr( OCROCRConfigFormGSetField );
       pGSet = FldGetTextPtr(fldP);

       fldP = (FieldPtr) GetObjectPtr( OCROCRConfigFormHSeField );
       pHSet = FldGetTextPtr(fldP);

       fldP = (FieldPtr) GetObjectPtr( OCROCRConfigFormCheckField );
       pCheck = FldGetTextPtr(fldP);

       listP = (ListType*) GetObjectPtr(OCROCREnableList);
       font = LstGetSelection(listP);

       //Send the configuration to the ScanCmd Library
       ScanCmdSetupOCR(font, pTemplate, pGSet, pHSet, pCheck);
}
```

Appendix A

Extended Function Code Examples

Example using the decWriteSymbologyConfig function

Here is an example to activated the enable Aztec Runes option for Aztec symbology

```
AZTEC_T symSetRange;
symSetRange.dwMask          = SYM_MASK_FLAGS;
symSetRange.dwStructSize    = sizeof(AZTEC_T);
decReadSymbologyConfig(QT_CURRENT, barAZTEC, &symSetRange);
symSetRange.dwFlags         |= SYMBOLOGY_AZTEC_RUNE;
decWriteSymbologyConfig(barAZTEC, &symSetRange );
```

Example using the decReadSymbologyConfig function

Here is an example to check if Code93 Symbology has been enabled

```
CODE93_T symSetRange;
symSetRange.dwMask = SYM_MASK_FLAGS;
symSetRange.dwStructSize = sizeof(CODE93_T);
decReadSymbologyConfig(QT_CURRENT, Code93, &symSetRange);
if(symSetRange.dwFlags & SYMBOLOGY_ENABLE)
        //Symbology Code93 has been enabled
else
//Symbology Code93 has been disabled
```

Example using the ScanCmdGetSetupOCR function

Here is an example to retrieve the current OCR symbology setting:

```
MemHandle memTemplate, memGSet, memHSet, memCheck;
char *pTemplate, *pGSet, *pHSet, *pCheck;
OCRMode_t font;

memTemplate = MemHandleNew(256);
memGSet = MemHandleNew(256);
memHSet = MemHandleNew(256);
memCheck = MemHandleNew(256);

if (!memTemplate || !memGSet || !memHSet || !memCheck )
{
       MemHandleFree(memTemplate);
       MemHandleFree(memGSet);
       MemHandleFree(memHSet);
       MemHandleFree(memCheck);
       return;
}

pTemplate = (char *) MemHandleLock(memTemplate);
pGSet = (char *) MemHandleLock(memGSet);
pHSet = (char *) MemHandleLock(memHSet);
pCheck = (char *) MemHandleLock(memCheck);
ScanCmdGetSetupOCR(QT_CURRENT, (UInt8*) &font, pTemplate, pGSet, pHSet, pCheck);
```

Example using the ScanCmdSetupOCR function

Here is an example to modify option for OCR symbology to scan a OCR-B code with character "0123ABCDK".

```
MemHandle memTemplate, memGSet, memHSet, memCheck;
char *pTemplate, *pGSet, *pHSet, *pCheck;
OCRMode_t font;

memTemplate = MemHandleNew(256);
memGSet = MemHandleNew(256);
memHSet = MemHandleNew(256);
memCheck = MemHandleNew(256);

if (!memTemplate || !memGSet || !memHSet || !memCheck )
{
       MemHandleFree(memTemplate);
       MemHandleFree(memGSet);
       MemHandleFree(memHSet);
       MemHandleFree(memCheck);
       return;
}

pTemplate = (char *) MemHandleLock(memTemplate);
pGSet = (char *) MemHandleLock(memGSet);
pHSet = (char *) MemHandleLock(memHSet);
pCheck = (char *) MemHandleLock(memCheck);

StrCopy(pTemplate, "ddddeeeee");
StrCopy(pGSet,"");
StrCopy(pHSet,"");
StrCopy(pCheck,"");
ScanCmdSetupOCR(font, pTemplate, pGSet, pHSet, pCheck);
```

Methods to retrieve OCR scanning and barcode scanning data are the same.

Example using ScnCmdGetSetLightsMode function

Here is an example to set the light mode as aimer only mode

```
LightsMode_t nLightMode = LM_AIMER_ONLY;
ScnCmdGetSetLightsMode(true, QT_CURRENT, (UInt8*) &nLightMode);
```

Example using ScnCmdScanWedge function

Here is an example to turn on/off the ScanWedge function

```
ScnCmdScanWedge (true);      //turn on ScanWedge

ScnCmdScanWedge (false);     //turn off ScanWedge
```

Appendix B

Structures and Defines

Beep frequency and duration defines

#define MIN_BEEP_DURATION	0
#define MAX_BEEP_DURATION	10000
#define MIN_BEEP_FREQUENCY	0
#define MAX_BEEP_FREQUENCY	15000
#define MIN_UPCEAN_REDUNDANCY	2
#define MAX_UPCEAN_REDUNDANCY	20
#define MIN_LASER_ON_TIME	5
#define MAX_LASER_ON_TIME	99
#define MAX_AIM_DURATION	99
#define MAX_TIMEOUT_BETWEEN_SYMBOL	10
#define MAX_DECODE_LED_ON_TIME	100
#define SHORT_BEEP_DUR	0x20
#define MEDIUM_BEEP_DUR	0x21
#define LONG_BEEP_DUR	0x22
#define HIGH_FREQ	0x23
#define MEDIUM_FREQ	0x24
#define LOW_FREQ	0x25
#define DECODE_BEEP_DUR	0x27
#define DECODE_BEEP_FREQ	0x28
Error Messages	
#define STATUS_OK	0
#define NOT_SUPPORTED	-2
#define COMMUNICATIONS_ERROR	-3
#define BAD_PARAM	-4
#define BATCH_ERROR	-5
#define ERROR_UNDEFINED	-6
Triggering Modes	
#define LEVEL	0x00
#define PULSE	0x02
#define HOST	0x08
Enable/Disable	
#define DISABLE	0x00
#define ENABLE	0x01
Security Level	
#define SECURITY_LEVEL0	0x00
#define SECURITY_LEVEL1	0x01
#define SECURITY_LEVEL2	0x02
#define SECURITY_LEVEL3	0x03
#define SECURITY_LEVEL4	0x04

Supplementals

#define IGNORE_SUPPLEMENTALS	0x00
#define DECODE_SUPPLEMENTALS	0x01
#define AUTODISCRIMINATE_SUPPLEMENTALS	0x02

Transmit Check Digit

#define DO_NOT_TRANSMIT_CHECK_DIGIT	0x00
#define TRANSMIT_CHECK_DIGIT	0x01

Preamble

#define NO_PREAMBLE	0x00
#define SYSTEM_CHARACTER	0x01
#define SYSTEM_CHARACTER_COUNTRY_CODE	0x02

Discrete Length

#define ANY_LENGTH	0x00
#define ONE_DISCRETE_LENGTH	0x01
#define TWO_DISCRETE_LENGTHS	0x02
#define LENGTH_WITHIN_RANGE	0x03

Check Digit Verification

#define DISABLE_CHECK_DIGIT	0x00
#define USS_CHECK_DIGIT	0x01
#define OPCC_CHECK_DIGIT	0x02

MSI Plessy Check Digits

#define ONE_CHECK_DIGIT	0x00
#define TWO_CHECK_DIGITS	0x01

MSI Plessey check digit algorithms

#define MOD10_MOD11	0x00
#define MOD10_MOD10	0x01

Aimer Code ID

#define AIM_CODE_ID_CHARACTER	0x01
#define SYMBOL_CODE_ID_CHARACTER	0x02

Prefix and Suffix Values

#define PREFIX_SUFFIX_VALUES_P	0x69
#define PREFIX_SUFFIX_VALUES_S1	0x68
#define PREFIX_SUFFIX_VALUES_S2	0x6A

Scan Data Formats

#define DATA_AS_IS	0x00
#define DATA_SUFFIX1	0x01
#define DATA_SUFFIX2	0x02
#define DATA_SUFFIX1_SUFFIX2	0x03
#define PREFIX_DATA	0x04
#define PREFIX_DATA_SUFFIX1	0x05
#define PREFIX_DATA_SUFFIX2	0x06
#define PREFIX_DATA_SUFFIX1_SUFFIX2	0x07

Scan Angles

#define SCAN_ANGLE_WIDE	0xB6
#define SCAN_ANGLE_NARROW	0xB5

Maximum packet length

#define MAX_PACKET_LENGTH	258

Maximum DCD length

#define MAX_DCD_LENGTH	3000

Maximum Batch Param

#define MAX_BATCH_PARAM	247

Decode Parameters

#define PACKET_CONTINUATION	0x0002
#define MAX_DECODER_PARMS	1024

Barcode data types returned

#define BCTYPE_NOT_APPLICABLE	0x00
#define BCTYPE_CODE39	0x01
#define BCTYPE_CODABAR	0x02
#define BCTYPE_CODE128	0x03
#define BCTYPE_D2OF5	0x04
#define BCTYPE_IATA2OF5	0x05
#define BCTYPE_I2OF5	0x06
#define BCTYPE_CODE93	0x07
#define BCTYPE_UPCA	0x08
#define BCTYPE_UPCA_2SUPPLEMENTALS	0x48
#define BCTYPE_UPCA_5SUPPLEMENTALS	0x88
#define BCTYPE_UPCE0	0x09
#define BCTYPE_UPCE0_2SUPPLEMENTALS	0x49
#define BCTYPE_UPCE0_5SUPPLEMENTALS	0x89
#define BCTYPE_EAN8	0x0A
#define BCTYPE_EAN8_2SUPPLEMENTALS	0x4A
#define BCTYPE_EAN13_5SUPPLEMENTALS	0x8B
#define BCTYPE_EAN8_5SUPPLEMENTALS	0x8A
#define BCTYPE_EAN13	0x0B
#define BCTYPE_EAN13_2SUPPLEMENTALS	0x4B
#define BCTYPE_MSI_PLESSEY	0x0E
#define BCTYPE_EAN128	0x0F
#define BCTYPE_UPCE1	0x10
#define BCTYPE_UPCE1_2SUPPLEMENTALS	0x50
#define BCTYPE_UPCE1_5SUPPLEMENTALS	0x90
#define BCTYPE_CODE39_FULL_ASCII	0x13
#define BCTYPE_TRIOPTIC_CODE39	0x15
#define BCTYPE_BOOKLAND_EAN	0x16
#define BCTYPE_COUPON_CODE	0x17
#define BCTYPE_ISBT128	0x19

Added

#define BCTYPE_CODE32	0x20
#define BCTYPE_PDF417	0x11

Extended support by BABBO (Defined at a range of 0x31-0x3F, 0x60-0x6D)

#define BCTYPE_AZTEC	0x31
#define BCTYPE_CODE11	0x32

```
#define BCTYPE_CODE49                       0x33
#define BCTYPE_COMPOSITE                    0x34
#define BCTYPE_DATAMATRIX                   0x35
#define BCTYPE_MAXICODE                     0x36
#define BCTYPE_MICROPDF                     0x37
#define BCTYPE_POSTNET                      0x38
#define BCTYPE_OCR                          0x39
#define BCTYPE_QR                           0x3A
#define BCTYPE_RSS                          0x3B
#define BCTYPE_ISBT                         0x3C
#define BCTYPE_BPO                          0x3D
#define BCTYPE_CANPOST                      0x3E
#define BCTYPE_AUSPOST                      0x3F

#define BCTYPE_IATA25                       0x60
#define BCTYPE_CODABLOCK                    0x61
#define BCTYPE_JAPOST                       0x62
#define BCTYPE_PLANET                       0x63
#define BCTYPE_DUTCHPOST                    0x64
#define BCTYPE_MATRIX25                     0x65
#define BCTYPE_CHINAPOST                    0x66
#define BCTYPE_KOREAPOST                    0x67
#define BCTYPE_TELEPEN                      0x68
#define BCTYPE_CODE16K                      0x69
#define BCTYPE_POSICODE                     0x6A
#define BCTYPE_COUPONCODE                   0x6B
#define BCTYPE_USPS4CB                      0x6C
#define BCTYPE_IDTAG                        0x6D

#define EXTENDED_DATA_FLAG                  0x01

Decoder configuration definitions for each symbology
#define SYMBOLOGY_ENABLE                    0x00000001
#define SYMBOLOGY_CHECK_ENABLE              0x00000002
#define SYMBOLOGY_CHECK_TRANSMIT            0x00000004
#define SYMBOLOGY_START_STOP_XMIT           0x00000008
#define SYMBOLOGY_ENABLE_APPEND_MODE        0x00000010
#define SYMBOLOGY_ENABLE_FULLASCII          0x00000020
#define SYMBOLOGY_ENABLE_BASE32             0x00000040
#define SYMBOLOGY_NUM_SYS_TRANSMIT          0x00000040
#define SYMBOLOGY_2_DIGIT_ADDENDA           0x00000080
#define SYMBOLOGY_5_DIGIT_ADDENDA           0x00000100
#define SYMBOLOGY_ADDENDA_REQUIRED          0x00000200
#define SYMBOLOGY_ADDENDA_SEPARATOR         0x00000400
#define SYMBOLOGY_EXPANDED_UPCE             0x00000800
#define SYMBOLOGY_ISBT_CONCATENATE          0x00000040
#define SYMBOLOGY_COMPOSITE_UPC             0x00002000
#define SYMBOLOGY_AZTEC_RUNE                0x00004000
#define SYMBOLOGY_AUSTRALIAN_BAR_WIDTH      0x00010000

#define SYMBOLOGY_TELEPEN_OLD_STYLE         0x04000000
#define SYMBOLOGY_POSICODE_LIMITED_1        0x08000000
#define SYMBOLOGY_POSICODE_LIMITED_2        0x10000000
#define SYMBOLOGY_CODABAR_CONCATENATE       0x20000000
#define SYMBOLOGY_CODABAR_CONCAT_REQ        0x40000000
#define SYMBOLOGY_EAN13_ISBN_XLATE          0x00080000
```

Symbology structure set masks

```
#define SYM_MASK_FLAGS                  0x00000001
#define SYM_MASK_MIN_LEN                0x00000002
#define SYM_MASK_MAX_LEN                0x00000004
#define SYM_MASK_ALL                    0x00000007
```

Define aliases for each symbology structure

```
#define AZTEC_T             SymFlagsRange_t
#define CODABAR_T           SymFlagsRange_t
#define CODE11_T            SymFlagsRange_t
#define CODE128_T           SymFlagsRange_t
#define CODE39_T            SymFlagsRange_t
#define CODE49_T            SymFlagsRange_t
#define CODE93_T            SymFlagsRange_t
#define COMPOSITE_T         SymFlagsRange_t
#define DATAMATRIX_T        SymFlagsRange_t
#define EAN8_T              SymFlagsOnly_t
#define EAN13_T             SymFlagsOnly_t
#define INT25_T             SymFlagsRange_t
#define MAXICODE_T          SymFlagsRange_t
#define MICROPDF_T          SymFlagsRange_t
#define PDF417_T            SymFlagsRange_t
#define POSTNET_T           SymFlagsOnly_t
#define QR_T                SymFlagsRange_t
#define RSS_T               SymFlagsOnly_t
#define RSSLIM_T            SymFlagsOnly_t
#define RSSEXP_T            SymFlagsRange_t
#define UPCA_T              SymFlagsOnly_t
#define UPCE_T               SymFlagsOnly_t
#define UPCE1_T             SymFlagsOnly_t
#define BPO_T               SymFlagsOnly_t
#define CANPOST_T           SymFlagsOnly_t
#define AUSPOST_T           SymFlagsOnly_t
#define IATA25_T            SymFlagsRange_t
#define CODABLOCK_T         SymFlagsRange_t
#define JAPOST_T            SymFlagsOnly_t
#define PLANET_T            SymFlagsOnly_t
#define DUTCHPOST_T         SymFlagsOnly_t
#define MSI_T               SymFlagsRange_t
#define TLCODE39_T          SymFlagsOnly_t
#define MATRIX25_T          SymFlagsRange_t
#define KORPOST_T           SymFlagsRange_t
#define TRIOPTIC_T          SymFlagsOnly_t
#define CODE25_T            SymFlagsRange_t
#define PLESSEY_T           SymFlagsRange_t
#define CHINAPOST_T         SymFlagsRange_t
#define TELEPEN_T           SymFlagsRange_t
#define CODE16K_T           SymFlagsRange_t
#define POSICODE_T          SymFlagsRange_t
#define COUPONCODE_T        SymFlagsOnly_t
#define USPS4STATE_T        SymFlagsOnly_t
#define IDTAG_T             SymFlagsOnly_t
```

ScanEvent Structure

```
typedef struct
{
        eventsEnum     eType;
        Boolean        penDown;
        Int16          screenX;
        Int16          screenY;
        union scanData
        {
                struct scanGen
                {
                        UInt16 data1;
                        UInt16 data2;
                        UInt16 data3;
                        UInt16 data4;
                        UInt16 data5;
                        UInt16 data6;
                        UInt16 data7;
                        UInt16 data8;
                } scanGen;

                struct
                {
                        UInt16 batteryLevel;
                        UInt16 batteryErrorType;
                } batteryError;

        } scanData;         // End of union

} ScanEventType;
typedef ScanEventType *ScanEventPtr;
```

Barcode Message Structure

```
typedef struct  tagMESSAGE
{
        Int16 length;
        Int16 type;
        Int16 status;
        UInt8 data[MAX_PACKET_LENGTH];
} MESSAGE;
```

DCD Message Structure

```
typedef struct  tagMESSAGE_Dcd
{
        Int16 length;
        Int16 type;
        Int16 status;
        UInt8 symLetter;
        UInt8 symModifier;
        UInt8 data[MAX_DCD_LENGTH];
} MESSAGE_Dcd;
```

Batch Structure

```
typedef struct tagBATCH
{
        Int16 length;
        Int16 data[MAX_BATCH_PARAM + 3];
} BATCH;
```

Beep Sound enum

```
typedef enum tagBeepType
{
        No_Beep                         = 0x00,
        One_Short_High,
        Two_Short_High,
        Three_Short_High,
        Four_Short_High,
        Five_Short_High,

        One_Short_Low,
        Two_Short_Low,
        Three_Short_Low,
        Four_Short_Low,
        Five_Short_Low,

        One_Long_High,
        Two_Long_High,
        Three_Long_High,
        Four_Long_High,
        Five_Long_High,

        One_Long_Low,
        Two_Long_Low,
        Three_Long_Low,
        Four_Long_Low,
        Five_Long_Low,

        Fast_Warble,
        Slow_Warble,
        Mix1,
        Mix2,
        Mix3,
        Mix4,

        Decode_Beep,
        Bootup_Beep,
        Parameter_Entry_Error_Beep,
        Parameter_Defaults_Beep,
        Parameter_Entered_Beep,
        Host_Convert_Error_Beep,
        Transmit_Error_Beep,
        Parity_Error_Beep,

        Last_Beep
} BeepType;
```

```
Conversion enum
typedef enum tagConvertType
{
        convertUpcEtoUpcA,
        convertUpcE1toUpcA,
        convertCode39toCode32,
        convertEan8toEan13,
        convertI2of5toEan13
} ConvertType;

Beep Duration enum
typedef enum
{
        decodeDuration                  = DECODE_BEEP_DUR,
        shortDuration                   = SHORT_BEEP_DUR,
        mediumDuration                  = MEDIUM_BEEP_DUR,
        longDuration                    = LONG_BEEP_DUR
} DurationType;

Beep Frequency enum
typedef enum
{
        decodeFrequency                 = DECODE_BEEP_FREQ,
        lowFrequency                    = LOW_FREQ,
        mediumFrequency                 = MEDIUM_FREQ,
        highFrequency                   = HIGH_FREQ
} FrequencyType;

Button Type enum
typedef enum
{
        buttonDateBook                  = 0,
        buttonAddressBook,
        buttonTodoList,
        buttonMemoPad
} ButtonType;

Launch Code enum
typedef enum
{
        myAppCmdBatteryAlert        = sysAppLaunchCmdCustomBase,
        myAppCmdS24BatteryAlert,
        myAppCmdNotUsed
} MyAppCustomActionCodes;
```

Bar Code Enum

```
typedef enum tagBarType
{
        barCODE39           = 0x00,
        barUPCA             = 0x01,
        barUPCE             = 0x02,
        barEAN13            = 0x03,
        barEAN8             = 0x04,
        barD25              = 0x05,
        barI2OF5            = 0x06,
        barCODABAR          = 0x07,
        barCODE128          = 0x08,
        barCODE93           = 0x09,
        barTRIOPTIC39       = 0x0D,
        barUCC_EAN128       = 0x0E,
        barMSI_PLESSEY      = 0x0B,
        barUPCE1            = 0x0C,
        barBOOKLAND_EAN     = 0x53,
        barISBT128          = 0x54,
        barCOUPON           = 0x55,
        barCode32           = 0x56,
        barAZTEC            = 0xC0,
        barCODE11           = 0xC1,
        barCODE49           = 0xC2,
        barCOMPOSITE        = 0xC3,
        barDATAMAXTRIX      = 0xC4,
        barMAXICODE         = 0xC5,
        barMICROPDF         = 0xC6,
        barOCR              = 0xC7,
        barPDF417           = 0xC8,
        barPOSTNET          = 0xC9,
        barQR               = 0xCA,
        barRSS              = 0xCB,
        barBPO              = 0xCC,
        barCANPOST          = 0xCD,
        barAUSPOST          = 0xCE,
        barIATA25           = 0xCF,
        barCODABLOCK        = 0xD0,
        barJAPOST           = 0xD1,
        barPLANET           = 0xD2,
        barDUTCHPOST        = 0xD3,
        barTLCODE39         = 0xD4,
        barMATRIX25         = 0xD5,
        barCHINAPOST        = 0xD6,
        barKOREAPOST        = 0xD7,
        barTELEPEN          = 0xD8,
        barCODE16K          = 0xD9,
        barPOSICODE         = 0xDA,
        barUSPS4CB          = 0xDB,
        barIDTAG            = 0xDC,
        barRSS_LIM          = 0xDD,
        barRSS_EXP          = 0xDE,
        barMSI              = 0xA0,
        barPlessey          = 0xA1,
        barALL              = 0xFF
} BarType;
```

Decode Option Structure

```
typedef struct
{
        Int32                   Version;
        Int32                   PrintWeight;
        Int32                   DecodeMode;
        Int32                   LinearRange;
        Int32                   VideoReverse;
        Int32                   DecAttemptLimit
        Int32                   SearchLimit;
        unsigned short          AddendaDelay;
        unsigned short          UpcComDelay;
        unsigned short          ConcatDelay;
        unsigned short          MldEnaMask;
        Int32                   Flags;
} DecodeOptions_t;

typedef struct _tagSymFlagsOnly
{
        Int32   dwStructSize;
        Int32   dwMask;
        Int32   dwFlags;
} SymFlagsOnly_t, *PSymFlagsOnly_t;

Structure for symbologies with min & max length.
typedef struct _tagSymFlagsRange
{
        Int32   dwStructSize;
        Int32   dwMask;
        Int32   dwFlags;
        Int32   dwMinLen;
        Int32   dwMaxLen;
} SymFlagsRange_t, *PSymFlagsRange_t;

Structure of structures, one for each symbology.
typedef struct _tagSymCfg
{
        Int32                   dwStructSize;
        CODABAR_T               codabar;
        CODE11_T                code11;
        CODE128_T               code128;
        CODE39_T                code39;
        CODE49_T                code49;
        CODE93_T                code93;
        COMPOSITE_T             composite;
        DATAMATRIX_T            datamatrix;
        EAN8_T                  ean8;
        EAN13_T                 ean13;
        IATA25_T                iata25;
        INT25_T                 int2of5;
        MSI_T                   msi;
        UPCA_T                  upcA;
        UPCE_T                  upcE;
        UPCE1_T                 upcE1;
        AUSPOST_T               australiaPost;
```

```
        BPO_T               britishPost;
        CANPOST_T           canadaPost;
        DUTCHPOST_T         dutchPost;
        JAPOST_T            japanPost;
        PLANET_T            usPlanet;
        POSTNET_T           usPostnet;
        AZTEC_T             aztec;
        CODABLOCK_T         codablock;
        MAXICODE_T          maxicode;
        MICROPDF_T          microPDF;
        PDF417_T            pdf417;
        QR_T                qr;
        RSS_T               rss;
        TLCODE39_T          tlCode39;
        MATRIX25_T          matrix25;
        KORPOST_T           koreaPost;
        TRIOPTIC_T          triopticCode;
        CODE25_T            code2of5;
        PLESSEY_T           plesseyCode;
        CHINAPOST_T         chinaPost;
        TELEPEN_T           telepen;
        CODE16K_T           code16k;
        POSICODE_T          posiCode;
        COUPONCODE_T        couponCode;
        USPS4STATE_T        usps4Post;
        IDTAG_T             idtagPost;
        RSSLIM_T            rssLim;
        RSSEXP_T            rssExp;
} SymCfg_t, *PSymCfg_t;

typedef enum
{
        IF_RAW_BINARY = 0,
        IF_RAW_GRAY
} ImageFormat_t;

typedef enum
{
        LM_ILLUM_AIMER_OFF=0,
        LM_AIMER_ONLY,
        LM_ILLUM_ONLY,
        LM_ILLUM_AIMER,
} LightsMode_t;
```

```
typedef struct
{
        int AspectRatio;
        int OffsetX;
        int OffsetY;
        unsigned int width;
        unsigned int height;
        int resolution;
        ImageFormat_t format;
        unsigned int destNaturalWidthInPixels;
        unsigned int destNaturalHeightInPixels;
        unsigned int destWidthInPixels;
        unsigned int destHeightInPixels;
} IntellImgDesc_t;

typedef enum
{
        OCR_DISABLED = 0,
        OCR_A,
        OCR_B,
        OCR_MONEY,
        OCR_MICR,
}OCRMode_t;

typedef struct _RECT
{
        LONG left;
        LONG top;
        LONG right;
        LONG bottom;
} RECT, *PRECT, *LPRECT;

typedef struct
{
        DWORD       dwSize;
        DWORD       dwEngineID;
        DWORD       dwImagerRows;
        DWORD       dwImagerCols;
        DWORD       dwBitsPerPixel;
        DWORD       dwRotation;
        DWORD       dwAimerXoffset;
        DWORD       dwAimerYoffset;
        DWORD       dwYDepth;
} ImagerProperties_t;
```