

**CHD FIVE**

# Barcode Decoder API Reference



Computer Hardware Design SIA  
CHD Europe SIA

© All rights reserved.  
Computer Hardware Design SIA  
Dzelzavas 120i, Riga, LV-1021  
Latvia  
Phone:+371 67802812  
Fax:+371 67802822  
[www.chdeurope.com](http://www.chdeurope.com)

## Table of Contents

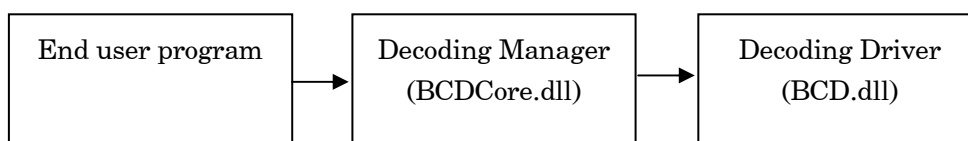
---

<b>1</b>	<b>Understanding barcode decoding subsystem</b>	<b>4</b>
1.1	Decode subsystem organization	4
1.2	Decode programming consideration	4
<b>2</b>	<b>Using the Barcode decode API</b>	<b>5</b>
<b>3</b>	<b>API Definitions</b>	<b>6</b>
3.1	Error Codes	6
3.2	Barcode Identifiers	6
3.3	Result notification methods	6
3.4	Terminator Identifiers	7
3.5	LED Identifiers and Control Defines	7
3.6	Barcode Setting Flags	7
<b>4</b>	<b>Data Structures</b>	<b>10</b>
4.1	Barcode configuration structure	10
4.2	Scanned Data structure	11
<b>5</b>	<b>Exported Functions</b>	<b>12</b>
5.1	BcdOpen	12
5.2	BcdClose	12
5.3	BcdIsEnabled	12
5.4	BcdEnable	12
5.5	BcdTrigTurnOn	12
5.6	BcdSetTrigTimeout	12
5.7	BcdSetResultType	12
5.8	BcdSetHWND	13
5.9	BcdSetUserMsg	13
5.10	BcdSetHEvent	13
5.11	BcdWaitResult	13
5.12	BcdTransmitBarID	13
5.13	BcdGetResult	13
5.14	BcdGetString	13
5.15	BcdGetLength	13
5.16	BcdGetBarType	13
5.17	BcdEnableGoodReadOff	14
5.18	BcdEnableBeep	14
5.19	BcdSetBeepWav	14
5.20	BcdSetBeepVolume	14
5.21	BcdEnableLED	14
5.22	BcdEnableVibrator	14
5.23	BcdSetVibratorInterval	14
5.24	BcdEanbleAutoScan	14
5.25	BcdGetHWND	15
5.26	BcdSetTerminator	15
5.27	BcdSetAutoScanInterval	15
5.28	BcdGetConfig	15
5.29	BcdSetConfig	15
5.30	BcdGetBarTypeString	15
5.31	BcdGetVersion	15
5.32	BcdWriteSetting	15
5.33	BcdSetDefault	15
5.34	BcdGetPrefix	16
5.35	BcdGetPostfix	16
5.36	BcdSetPrefix	16
5.37	BcdGetAcceptFilter	16
5.38	BcdGetDenyFilter	16
5.39	BcdSetAcceptFilter	16
5.40	BcdSetDenyFilter	16
5.41	BcdIsBarcodeEnabled	16
5.42	BcdEnableBarcode	16
5.43	BcdIsBarFlagEnabled	17
5.44	BcdSetBarFlag	17
5.45	BcdGetBarID	17
5.46	BcdSetBarID	17
5.47	BcdMSIGetCheckMode	17
5.48	BcdMSISetCheckMode	17
5.49	BcdMSIGetCheckDigit	17
5.50	BcdMSISetCheckDigit	17
5.51	BcdGetBarLength	18
5.52	BcdSetBarLength	18

## 1 Understanding barcode decoding subsystem

### 1.1 DECODE SUBSYSTEM ORGANIZATION

Decoding on the CHD FiVE is accomplished through the barcode decoder driver(BCD.dll). The it can be controlled with the decode manager(BCDCore.dll). The decode manager provide all the API (Application Programming Interface) functions and definitions.



In the above figure, a decoding device represents a physical device. The Decode Manager has the responsibility of being the single interface point for functions common to input data types, such as label programming, and keyboard wedge support. Applications should use it to enumerate the decoding devices present in the system. Once a suitable device is found, they can get data from it by directly communicating with decoding driver.

The decoding system is based on the concept of input data types. The distinction is based on the automatic data collection technology used to collect the data. The currently supported data types are any kind of 1D barcode(Laser, CCD).

### 1.2 DECODE PROGRAMMING CONSIDERATION

The decoder manager supports only single interface. For example, if an application is using Keyboard wedge, and when the other application which use the window message is run, then the previous application cannot get keyboard wedge result until it set the input type again.

Many applications can use barcode decoder, however, the responsibility of the output is depend on the application. So, applications should consider other applications.

Single decoder control

The decoder will be enabled or disabled by last application. So if there are many applications are running, and if an application quit as decoder disabled, other application also can't decode anymore. They should enable the decoder by themselves.

## 2 Using the Barcode decode API

The decoding API is made up of direct Win32 style function calls, which are exported by BCDCore.DLL. To use this DLL in your application do the following:

In your Embedded Visual C++ project, add BCDCore.LIB to the list of import libraries. Under the Project menu, select "Settings". In the tab named "Link", add "BCDCore.LIB" to the box labeled "Object/library modules".

In each source file which makes calls to the decoding API, include the header file BCDAPI.H.

## 3 API Definitions

### 3.1 ERROR CODES

Whenever it scan a barcode, it returns an error code. The following values may be returned:

ERROR_NO_ERROR	Decoding is successful, there is no error
ERROR_NO_READ	There is no values.
ERROR_WRONG_DATA	The data format is wrong.
ERROR_BAD_DATA	The data format is wrong.
ERROR_BAD_TYPE	The barcode type is mismatch with the data.
ERROR_WRONG_TYPE	The barcode type is mismatch with the data.
ERROR_UNKNOWN_TYPE	The barcode type is unknown.
ERROR_NO_SCAN_DATA	The decoder does not response.
ERROR_SCAN_TIMEOUT	The trigger is deactivated after trigger time period.

### 3.2 BARCODE IDENTIFIERS

Pass following values for barcode type in API parameters:

BAR_TYPE_ERROR	Unknown Barcode Type
BAR_UNKNOWN	Unknown Barcode Type
BAR_EAN_13	EAN-13
BAR_EAN_8	EAN-8 / JAN-8
BAR_UPC_E	UPC-E
BAR_CODE_39	CODE 39
BAR_CODABAR_NW7	CODABAR / NW7
BAR_MATRIX_25	MATRIX 2 of 5
BAR_M25	MATRIX 2 of 5
BAR_INDUSTRIAL_25	Industrial 2 of 5
BAR_S25	Industrial 2 of 5
BAR_INTERLEAVE_25	Interleaved 2 of 5
BAR_I25	Interleaved 2 of 5
BAR_CODE_128	CODE 128
BAR_CODE_93	CODE 93
BAR_CODE_11	CODE 11
BAR_MSI_PLESSEY	MSI/Plessey
BAR_CHINA_POSTAGE	China Postage
BAR_DATA_LOGIC_25	Datalogic 2 of 5
BAR_D25	Datalogic 2 of 5
BAR_CODE_35	Korea Postage CODE 35
BAR_CODE_4	CODE 4
BAR_CODE_32	CODE 32 / Italian pharmacy
BAR_ISBN	ISBN
BAR_EAN_8_TO_13	EAN 8 Convert to EAN 13
BAR_UPC_E_TO_A	UPC-E Convert to UPC-A
BAR_EAN_ADD_25	UPC/EAN Add on 2 of 5
BAR_IATA_25	IATA 2 of 5
BAR_UPC_A	UPC-A
BAR_RSS	RSS
BAR_ALL	Every barcodes type

### 3.3 RESULT NOTIFICATION METHODS

Whenever scanning a barcode, driver will notify with following methods:

RESULT_USERMSG	Windows Message type. Set the window handle first. It can be set as user message. Default is WM_SCANNED.
RESULT_KBDMSG	Keyboard Wedge. If the Insertion cursor is blinking, the data will be typed on the cursor location like keyboard typing.
RESULT_COPYPASTE	Keyboard Wedge. If the Insertion cursor is blinking, the Data will be pasted. It'll give quick response than keyboard typing(RERESULT_KBDMSG).
RESULT_EVENT	Driver will notify event which set by API.
RESULT_CALLBACK	Call callback functions. Not supported yet.

### 3.4 TERMINATOR IDENTIFIERS

The Decoded data can get following terminators:

TERMINATOR_NONE	Remove Terminator
TERMINATOR_CRLF	Add CR/LF
TERMINATOR_CR	Add CR
TERMINATOR_LF	Add LF
TERMINATOR_SPACE	Add Space
TERMINATOR_TAB	Add Tab
TERMINATOR_STXETX	Add STX(0x02) before first character, ETX(0x03) after last character.

### 3.5 LED IDENTIFIERS AND CONTROL DEFINES

Barcode Scanner LED(Top of the terminal) defined like following:

LED_RED	Red color LED
LED_GREEN	Green color LED
LED_AMBER	Red and Green LEDs. If we turn them on together, the color will be like amber color
LED_OFF	Turn LED off
LED_ON	Turn LED on
LED_BLINK	Blink LED

### 3.6 BARCODE SETTING FLAGS

Decoder can be set with following flags with API:

Flags for CODE 39

CODE39_FULLASCII	Disabled : Standard Enabled : Full ASCII
CODE39_STARTEND	Disabled : Ignore Start/End code Enabled : Send Start/End code
CODE39_VERIFY	Disabled : Do not verify checksum Enabled : Verify checksum
CODE39_CHKSUM	Disabled : Do not transmit Checksum Enabled : Transmit Checksum

Flags for CODE 11

CODE11_CHK_1_DIGIT	Disabled : Check 2 digits Enabled : Check 1 digit
CODE11_CHK_DIGIT_TX	Disabled : Do not transmit Check digit Enabled : Transmit Check digit

Flags for Interleaved 2 of 5

I25_VERIFY	Disabled : Do not verify checksum Enabled : Verify checksum
I25_DIGIT_TX	Disabled : Do not transmit Checksum Enabled : Transmit Checksum
I25_FIXED	Disabled : Verify the minimum/maximum length Enabled : Verify the fixed length
I25_MIN_LEN	Set minimum Length, default is 2.
I25_MAX_LEN	Set maximum Length, default is 30.

Flags for Industrial 2 of 5

S25_VERIFY	Disabled : Do not verify checksum Enabled : Verify checksum
S25_DIGIT_TX	Disabled : Do not transmit Checksum Enabled : Transmit Checksum
S25_FIXED	Disabled : Verify the minimum/maximum length Enabled : Verify the fixed length
S25_MIN_LEN	Set minimum Length, default is 2.
S25_MAX_LEN	Set maximum Length, default is 30.

Flags for Matrix 2 of 5	
M25_VERIFY	Disabled : Do not verify checksum Enabled : Verify checksum
M25_DIGIT_TX	Disabled : Do not transmit Checksum Enabled : Transmit Checksum
M25_FIXED	Disabled : Verify the minimum/maximum length Enabled : Verify the fixed length
M25_MIN_LEN	Set minimum Length, default is 2.
M25_MAX_LEN	Set maximum Length, default is 30.
Flags for Datalogic 2 of 5	
D25_VERIFY	Disabled : Do not verify checksum Enabled : Verify checksum
D25_DIGIT_TX	Disabled : Do not transmit Checksum Enabled : Transmit Checksum
D25_FIXED	Disabled : Verify the minimum/maximum length Enabled : Verify the fixed length
D25_MIN_LEN	Set minimum Length, default is 2.
D25_MAX_LEN	Set maximum Length, default is 30.
Flags for CODABAR/NW7	
NW7_UPPER_LOWER	Disabled : Transfer Start/End character as capital. Enabled : Transfer Start/End character as lower.
NW7_ABCD_TNE	Disabled : Transfer Start/End character as ABCD Enabled : Transfer Start/End character as TN*E
NW7_STARTEND	Disabled : Do not transmit Start/End character. Enabled : Transmit Start/End character.
Flags for EAN-13	
EAN13_LEADING_DIGIT	Disabled : Do not transmit First Digit Enabled : Transmit First Digit
EAN13_LEADING_ZERO	Disabled : Do not Transmit Second Digit Enabled : Transmit First Digit
EAN13_TX_CHK_DIGIT	Disabled : Do not transmit Check Character Enabled : Transmit Check Character
Flags for UPC-A	
UPC_A_LEADING_DIGIT	Disabled : Do not transmit First Digit Enabled : Transmit First Digit
UPC_A_LEADING_ZERO	Disabled : Do not Transmit Second Digit Enabled : Transmit First Digit
UPC_A_TX_CHK_DIGIT	Disabled : Do not transmit Check Character Enabled : Transmit Check Character
Flags for EAN-8	
EAN8_LEADING_DIGIT	Disabled : Do not transmit Leading Digit Enabled : Transmit Leading Digit
EAN8_TX_CHK_DIGIT	Disabled : Do not transmit Check Character Enabled : Transmit Check Character
Flags for UPC-E	
UPC_E_LEADING_DIGIT	Disabled : Do not transmit Leading Digit Enabled : Transmit Leading Digit
UPC_E_TX_CHK_DIGIT	Disabled : Do not transmit Check Character Enabled : Transmit Check Character
UPC_E_EQUAL_UPCA	Disabled : Transmit as UPC-E Enabled : Transmit as UPC-A
Flags for CODE 93	
CODE93_CONCAT	Disabled : Disable Concatenation Enabled : Enable Concatenation
Flags for EAN-8 convert to EAN-13	
EAN8TOEAN13_ZERO_POSITION	Disabled : Add Zeros into middle of data Enabled : Add Zeors in front of the data

Flags for UPC/EAN add on 2 of 5

UPCEANADDON\_READ\_ONLY      Disabled : Read UPC/EAN and UPC/EAN add on 2 of 5  
Enabled : Read UPC/EAN add on 2 of 5 only

Flags for RSS

RSS\_TX\_CHKSUM      Disabled : Do not transmit Checksum  
Enabled : Transmit Checksum  
RSS14\_TO\_UCC128 Disabled : Do not Transfer RSS14 to UCC 128  
Enabled : Transfer RSS14 to UCC 128

Flags for MSI

MSI\_CKDMODE\_10      Use Check mode 10  
MSI\_CKDMODE\_10\_10      Use Check mode 10-10  
MSI\_CKDMODE\_11\_10      Use Check mode 11-10  
MSI\_0\_CHECKDIGIT Do not transmit check digit  
MSI\_1\_CHECKDIGIT Transmit 1 Check digit  
MSI\_2\_CHECKDIGIT Transmit 2 Check digits

Flags for CODE 128

CODE128\_ENABLE\_UCCEAN128      Disabled : Enable UCC/EAN 128  
Enabled : Disable UCC/EAN 128

## 4 Data Structures

### 4.1 BARCODE CONFIGURATION STRUCTURE

```
typedef struct _BCD_CONFIG
{
    BOOL bEnabled;

    BYTE uResultType;

    BOOL bGoodReadOff;
    BYTE uTerminator;
    BOOL bBarID;
    TCHAR szPrefix[MAX_PATH];
    TCHAR szPostfix[MAX_PATH];
    TCHAR szAcceptFilter[MAX_PATH];
    TCHAR szDenyFilter[MAX_PATH];

    BOOL bAutoScan;
    DWORD dwAutoScanInterval;

    DWORD dwTrigTimeout;

    BOOL bBeep;
    TCHAR szSuccessWav[MAX_PATH];
    TCHAR szFailWav[MAX_PATH];
    INT nBeepVolume;

    BOOL bLED;

    BOOL bVibrator;
    DWORD dwVibratorSuccessInterval;
    DWORD dwVibratorFailInterval;
} BCD_CONFIG, *PBCD_CONFIG;
```

bEnabled	TRUE : Decoder is Enabled. It's able to scan. FALSE : Decoder is Disabled. It's not able to scan.
uResultType	Current notification methods
bGoodReadOff	TRUE : Trigger turns off when the Trigger timeout passed FALSE : Trigger turns off when Trigger Turn Off command sent
uTerminator	Current terminator
bBarID	TRUE : Transmit Barcode ID FALSE : Do not transmit barcode ID
szPrefix	Add szPrefix in front of data
szPostfix	Add szPostfix after data
szAcceptFilter	Accept the barcode which includes szAcceptFilter string
szDenyFilter	Deny the barcode which includes szDenyFilter string
bAutoScan	TRUE : Turn on trigger automatically after decoding FALSE : Do not rescan after decoding
dwAutoScanInterval	Current Auto scanning interval, unit is milliseconds
dwTrigTimeout	Current Trigger Timeout. Trigger turn off when it pass timeout
bBeep	TRUE : Play beep sound FALSE : Mute beep sound
szSuccessWav	Wav file which plays when the decoding successful
szFailWav	Wav file which plays when the decoding fail
nBeepVolume	Beep volume, unit is percentage(%)
bLED	TRUE : Turn on LED when decode FALSE : Do not turn on LED when decode
bVibrator	TRUE : Vibrate the unit when decode FALSE : Do not Vibrate the unit when decode
dwVibratorSuccessInterval	Current Vibrating interval when decoding successful
dwVibratorFailInterval	Current Vibrating interval when decoding fail

## 4.2 SCANNED DATA STRUCTURE

```
typedef struct _SCAN_RESULT
{
    TCHAR          szScanValue[MAX_SCAN_LENGTH];
    BAR_TYPE       barType;
    UINT           nLength;
} SCAN_RESULT, *PSCAN_RESULT;
```

szScanValue     Decoded barcode value includes barcode ID, Prefix, Postfix, Terminator.  
barType         Barcode Type  
nLength         String length of szScanValue

## 5 Exported Functions

### 5.1 BCDOPEN

---

Syntax	HANDLE BcdOpen(void)
Description	This opens the decoder handle. To use APIs, it should be called at first.
Return Values	Handle of the decoder

### 5.2 BCDCLOSE

---

Syntax	void BcdClose(void)
Description	This closes the current decoder handle.

### 5.3 BCDISENABLED

---

Syntax	BOOL BcdIsEnabled(void)
Description	This gets the status of decoder, enabled or disabled.
Return Values	Status of decoder. If it is TRUE, the decoder is ready to scan. If it is FALSE, the decoder can't turn on the trigger.

### 5.4 BCDENABLE

---

Syntax	void BcdEnable(BOOL bEnable)
Description	Enable or Disable the decoder.
Parameters	
<i>bEnable</i>	Set TRUE to Enable the decoder or set FALSE to Disable the decoder If it is TRUE, the decoder is ready to scan. If it is FALSE, the decoder can't turn on the trigger.

### 5.5 BCDTRIGTURNON

---

Syntax	void BcdTrigTurnOn(BOOL bTurnOn)
Description	This turns on or off the trigger by software. If the trigger turned on already and turn on the trigger again, this will be ignored.
Parameters	
<i>bTurnOn</i>	Set TRUE to turn on the trigger or set FALSE to turn off the decoder

### 5.6 BCDSETTRIGTIMEOUT

---

Syntax	void BcdSetTrigTimeout(DWORD dwMilliseconds)
Description	This sets the trigger timeout periods. When the trigger turned on, after timeout periods, trigger will be turned off automatically.
Parameters	
<i>dwMilliseconds</i>	Period of trigger working as milliseconds.

### 5.7 BCDSETRESULTTYPE

---

Syntax	void BcdSetResultType(DWORD dwType)
Description	This sets the result notification methods.
Parameters	
<i>dwType</i>	Refer the Result notification methods.

## 5.8 BCDSETHWND

---

Syntax	void BcdSetHwnd(HWND hWnd)
Description	This sets the Window Handle which receives the decoding result. It's only useful, when the notification method is RESULT_USERMSG. If the handle is invalid, the Window Message will not be sent.
Parameter <i>hWnd</i>	Window handle which receives the decoding result.

## 5.9 BCDSETUSERMSG

---

Syntax	void BcdSetUserMsg(UINT uUserMsg)
Description	Set the User window message. If it isn't used, Decoder will send the WM_SCANNED(WM_USER+0x7777) message when decoding.
Parameters <i>uUserMsg</i>	Window message which is sent to the Result Window.

## 5.10 BCDSETHEVENT

---

Syntax	void BcdSetHEvent(HANDLE hEvent)
Description	Set the Event Handle which is sent after decoding
Parameters <i>hEvent</i>	Event Handle which is sent after decoding. This can be got using CreateEvent function.

## 5.11 BCDWAITRESULT

---

Syntax	void BcdWaitResult()
Description	If the notification method is RESULT_EVENT, then this function waits until the event which is set happen. You'd better to use the thread to have the event.

## 5.12 BCDTRANSMITBARID

---

Syntax	void BcdTransmitBarID(BOOL bTransmit)
Description	This sets to transmit or not to transmit the Barcode ID.
Parameters <i>bTransmit</i>	Set to TRUE to transmit the Barcode ID Set to FALSE not to transmit the barcode ID.

## 5.13 BCDGETRESULT

---

Syntax	PSCAN_RESULT BcdGetResult(void)
Description	It returns the Scanning result structure pointer which has the value, barcode type, and length.
Return Value	Scanned Data structure pointer.

## 5.14 BCDGETSTRING

---

Syntax	LPWSTR BcdGetString(void)
Description	It returns the latest scanned string value.
Return value	Scanned string pointer.

## 5.15 BCDGETLENGTH

---

Syntax	UINT BcdGetLength(void)
Description	It returns the scanned string length of latest scanned.
Return Value	scanned string length.

## 5.16 BCDGETBARTYPE

---

Syntax	BAR_TYPE BcdGetBarType(void);
Description	It returns the scanned barcode type of latest scanned.
Return Value	scanned barcode type.

### 5.17 BCDENABLEGOODREADOFF

---

Syntax	void BcdEnableGoodReadOff(BOOL bGoodReadOff)
Description	It enables or disables Good Read off. If Good Read Off enabled, the trigger off command will be ignored. And it'll turn off the trigger after trigger time out automatically.
Parameters	
<i>bGoodReadOff</i>	Set to True to enable Good Read Off. Set to FALSE to disable Good Read Off

### 5.18 BCDENABLEBEEP

---

Syntax	void BcdEnableBeep(BOOL bBeep)
Description	This plays or mutes Beep sound after decoding.
Parameters	
<i>bBeep</i>	Set to TRUE to play the beep sound after decoding. Set to FALSE not to play the beep sound.

### 5.19 BCDSETBEEPWAV

---

Syntax	void BcdSetBeepWav(TCHAR *szSuccessWav, TCHAR *szFailWav)
Description	Set the WAV file for beep sound. It should includes the extension and path.
Parameters	
<i>szSuccessWav</i>	Beep WAV file when decoding is successful. It can be NULL, if it is NULL, the beep sounds will not be played.
<i>szFailWav</i>	Beep WAV file when decoding is fail. It can be NULL, if it is NULL, the beep sounds will not be played.

### 5.20 BCDSETBEEPVOLUME

---

Syntax	void BcdSetBeepVolume(INT nVolume)
Description	Set the volume of Beep sound as percentage(%).
Parameters	
<i>nVolume</i>	Volume of Beep sound as percentage(%).

### 5.21 BCDENABLELED

---

Syntax	void BcdEnableLED(BOOL bLED)
Description	Set to Enable or Disable the LED works.
Parameters	
<i>bLED</i>	Set to TRUE to turn on the LED during and after decoding. Set to FALSE not to turn on the LED during and after decoding.

### 5.22 BCDENABLEVIBRATOR

---

Syntax	void BcdEnableVibrator(BOOL bVibrator)
Description	Set to Enable or Disable to vibrate the unit after decoding.
Parameters	
<i>bVibrator</i>	Set to TRUE to vibrate after decoding. Set to FALSE not to vibrate after decoding.

### 5.23 BCDSETVIBRATORINTERVAL

---

Syntax	void BcdSetVibratorInterval(DWORD dwSuccessInterval, DWORD dwFailInterval)
Description	Set the interval to vibrate for successful or failure decoding.
Parameters	
<i>dwSuccessInterval</i>	The interval to vibrate when decoding successful. This can be 0, then vibrator will not work.
<i>dwFailInterval</i>	The interval to vibrate when decoding failed. This can be 0, then vibrator will not work.

### 5.24 BCDENABLEAUTOSCAN

---

Syntax	void BcdEanbleAutoScan(BOOL bAutoScan)
Description	Enable or Disable the Auto Scan. If it's enabled, after decoding, it'll automatically turn on the trigger.
Parameters	
<i>bAutoScan</i>	Set to TRUE to enable the Auto Scan. Set to FALES to disable the Auto Scan.

### 5.25 BCDGETHWND

---

Syntax	HWND BcdGetHWND(void)
Description	This returns the Current window handle which receives the window message. You can use this to returns the handle after new job.
Return Value	Current window handle which receives the window message.

### 5.26 BCDSETTERMINATOR

---

Syntax	void BcdSetTerminator(BYTE cTerminator)
Description	This sets the terminator which is added after the data. This is useful when you use the keyboard wedge.
Parameters	
<i>cTerminator</i>	Terminator define. Refer the Definition of Terminator.

### 5.27 BCDSETAUTOSCANINTERVAL

---

Syntax	void BcdSetAutoScanInterval(DWORD dwInterval)
Description	This sets the Interval for rescanning after decoding.
Parameters	
<i>dwInterval</i>	Interval for rescanning after decoding as milliseconds.

### 5.28 BCDGETCONFIG

---

Syntax	void BcdGetConfig(PBCD_CONFIG pbcdConfig)
Description	This gets the major configurations with the Decoder configuration structure.
Parameters	
<i>pbcdConfig</i>	Pointer of the decoder configuration structure which has the configuration data.

### 5.29 BCDSETCONFIG

---

Syntax	void BcdSetConfig(PBCD_CONFIG pbcdConfig)
Description	This sets the major configurations at once.
Parameters	
<i>pbcdConfig</i>	Pointer of the decoder configuration structure which you want to set.

### 5.30 BCDGETBARTYPESTRING

---

Syntax	LPWSTR BcdGetBarTypeString(BAR_TYPE barType)
Description	Returns the barcode type as string(English).
Parameters	
<i>barType</i>	Barcode type which you want to know the name.
Return Value	String(English) of Barcode type.

### 5.31 BCDGETVERSION

---

Syntax	BOOL BcdGetVersion(TCHAR* szVersion, UINT nBufLen)
Description	Read decoder firmware version
Parameters	
<i>szVersion</i>	String buffer which will receives the version of decoder firmware.
<i>nBufLen</i>	Length of szVersion
Return Value	If it is TRUE, function works successfully. If it is FALSE, the function was failed to get the version.

### 5.32 BCDWRITESETTING

---

Syntax	BOOL BcdWriteSetting()
Description	Write current firmware setting. It'll permanently save the setting into the EEPROM.

### 5.33 BCDSETDEFAULT

---

Syntax	BOOL BcdSetDefault()
Description	Recover the setting as factory defaults.

---

### 5.34 BCDGETPREFIX

---

Syntax void BcdGetPrefix(TCHAR \*szPrefix)  
 Description Get the Prefix which will be added in front of the data.  
 Parameters  
   *szPrefix* Prefix string pointer.

### 5.35 BCDGETPOSTFIX

---

Syntax void BcdGetPostfix(TCHAR \*szPostfix)  
 Description Get the Postfix which will be added after the data.  
 Parameters  
   *szPostfix* Postfix string pointer.

---

### 5.36 BCDSETPREFIX

---

Syntax void BcdSetPrefix(TCHAR \*szPrefix)  
 Description Set the Prefix which will be added in front of the data.  
 Parameters  
   *szPrefix* Prefix string pointer.

Syntax void BcdSetPostfix(TCHAR \*szPostfix)  
 Description Set the Postfix which will be added after the data.  
 Parameters  
   *szPostfix* Postfix string pointer.

---

### 5.37 BCDGETACCEPTFILTER

---

Syntax void BcdGetAcceptFilter(TCHAR \*szFilter)  
 Description Get the filter string which will be accepted when this filter included in.  
 Parameters  
   *szFilter* Filter string pointer.

---

### 5.38 BCDGETDENYFILTER

---

Syntax void BcdGetDenyFilter(TCHAR \*szFilter)  
 Description Get the filter string which will be denied when this filter included in.  
 Parameters  
   *szFilter* Filter string pointer.

---

### 5.39 BCDSETACCEPTFILTER

---

Syntax void BcdSetAcceptFilter(TCHAR \*szFilter)  
 Description Set the filter string which will be accepted when this filter included in.  
 Parameters  
   *szFilter* Filter string pointer.

---

### 5.40 BCDSETDENYFILTER

---

Syntax void BcdSetDenyFilter(TCHAR \*szFilter)  
 Description Set the filter string which will be denied when this filter included in.  
 Parameters  
   *szFilter* Filter string pointer.

---

### 5.41 BCDISBARCODEENABLED

---

Syntax BOOL BcdIsBarcodeEnabled(BAR\_TYPE barType)  
 Description Check the Enabled status of the barcode.  
 Parameter  
   *barType* Barcode type which you want to check the status.  
 Return Value If it is TRUE, the barcode type is enabled to decode. If it is FALSE, the barcode type will not return any value after decoding.

---

### 5.42 BCDENABLEBARCODE

---

Syntax void BcdEnableBarcode(BAR\_TYPE barType, BOOL bEnable)

Description	Enable or Disable the barcode type.
Parameters	
<i>barType</i>	Barcode Type which you want to enable or disable.
<i>bEnable</i>	Set to TRUE to Enable the barcode or Set to FALSE to Disable the barcode.

### 5.43 BCDISBARFLAGENABLED

---

Syntax	BOOL BcdIsBarFlagEnabled(BAR_TYPE barType, BYTE uFlag)
Description	Check the Barcode Flag enabled or disabled.
Parameters	
<i>barType</i>	Barcode Type which you want to check the status.
<i>uFlag</i>	Barcode Flag which you want to check the status. It needs to use carefully. The flag definition should related with the barcode type( <i>barType</i> ). The function doesn't verify the flag is related with the barcode.

### 5.44 BCDSETBARFLAG

---

Syntax	void BcdSetBarFlag(BAR_TYPE barType, BYTE uFlag, BOOL bEnable)
Description	Set the Barcode Flag enabled or disabled.
Parameters	
<i>barType</i>	Barcode Type which you want to set the status.
<i>uFlag</i>	Barcode Flag which you want to set the status. It needs to use carefully. The flag definition should related with the barcode type( <i>barType</i> ). The function doesn't verify the flag is related with the barcode.

### 5.45 BCDGETBARID

---

Syntax	CHAR BcdGetBarID(BAR_TYPE barType)
Description	Get the defined Barcode ID.
Parameter	
<i>barType</i>	Barcode Type which you want to know the barcode ID.
Return value	It returns the barcode ID as a character.

### 5.46 BCDSETBARID

---

Syntax	void BcdSetBarID(BAR_TYPE barType, char chID)
Description	Set the Barcode ID related with the barcode type.
Parameter	
<i>barType</i>	Barcode Type which you want to set the barcode ID.
<i>chID</i>	Barcode ID as a character.

### 5.47 BCDMSIGETCHECKMODE

---

Syntax	int BcdMSIGetCheckMode()
Description	Get the MSI Check Mode 10 or 10-10, or 11-10.
Return value	It returns MSI Check Mode.

### 5.48 BCDMSISETCHECKMODE

---

Syntax	void BcdMSISetCheckMode(int nMode)
Description	Set the MSI Check Mode.
Parameter	
<i>nMode</i>	MSI Check Mode. Refer the MSI Flags

### 5.49 BCDMSIGETCHECKDIGIT

---

Syntax	int BcdMSIGetCheckDigit()
Description	Get the MSI Check Digit.
Return value	It returns MSI Check Digit.

### 5.50 BCDMSISETCHECKDIGIT

---

Syntax	void BcdMSISetCheckDigit(int nDigit)
Description	Set the MSI Check Digit.
Parameter	

---

*nMode* MSI Check Digit. Refer the MSI Flags

### 5.51 BCDGETBARLENGTH

---

Syntax void BcdGetBarLength(BAR\_TYPE barType, PBYTE puMin, PBYTE puMax)  
Description Get the available length of barcode data. You can use this as one of accept filter.  
Parameter  
*barType* Barcode Type which you want to get the length.  
*puMin* Minimum length of barcode data pointer  
*puMax* Maximum length of barcode data pointer.

### 5.52 BCDSETBARLENGTH

---

Syntax void BcdSetBarLength(BAR\_TYPE barType, BYTE uMin, BYTE uMax)  
Description Set the available length of barcode data. You can use this as one of accept filter.  
Parameter  
*barType* Barcode Type which you want to set the length.  
*uMin* Minimum length of barcode data.  
*uMax* Maximum length of barcode data.