DENSO

Bar Code Handy Terminal

BHT-400-CE

API Reference Manual

Contents

Chapter 1. Software Requirements for the BHT-400 ... 1
1.1. Operating System (OS) on the BHT-400 ... 1
1.2. Application Development Software on the PC ... 1
1.2.1. Application Development Tool ... 1
1.2.2. Software Development Kit ... 1
Chapter 2. Application Development Environment ... 2
2.1. Required Hardware (PC to be used for application development) ... 2
2.2. Required Software ... 2
2.3. Installation ... 2
Chapter 3. Output to the LCD Screen ... 3
3.1. Screen Fonts ... 3
Chapter 4. Backlight Control ... 4
4.1. Outline ... 4
4.2. Setting the Backlight Function On/Off Key ... 4
4.3. Setting the Backlight Illumination Time ... 5
4.4. Setting the Backlight Brightness and Power Saving Mode ... 5
4.5. Controlling the Backlight with the Backlight Control Key ... 6
4.6. Controlling the Backlight with the Backlight Control Function ... 7
Chapter 5. Beeper and Vibrator Control ... 8
5.1. Outline ... 8
5.2. Setting the Beeper/Vibrator ... 9
5.3. Starting/Stopping the Beeper/Vibrator ... 10
5.4. Priority Orders between Events that Activate the Beeper/Vibrator ... 10
5.5. Beeper Volume Patterns ... 10
Chapter 6. Keys and Trigger Switch Control ... 11
6.1. Outline ... 11
6.2. Setting the Keys and Trigger Switch ... 12
6.3. Shift Key Operation Mode ... 13
6.4. Magic Key Control ... 13
6.5. Assigning a User-Defined Key Code to the Magic Keys ... 14
6.5.1. Assignment Method ... 14
6.5.2. User-Defined Code Settings File (MKeyDef.txt) ... 14
6.6. Key Input Modes ... 15
6.6.1. Numeric Entry Mode ... 15
6.6.2. Alphabet Entry Mode ... 15
6.7. Function Mode ... 17
6.8. Key Clicks ... 17
6.9. Acquisition of Keypad Type ... 17
Chapter 7. LCD Status Indication ... 18
7.1. Outline ... 18
7.2. Setting the LCD Status Indication ... 19
Chapter 8. Power Management ... 20
8.1. Outline ... 20
8.2. Standby ... 21
8.2.1. Switching to the Standby State ... 21
8.2.2. Standby Transition Prohibited Events ... 21
8.2.3. Setting the Standby Transition Timeout ... 21
8.3. Suspend ... 22
8.3.1. Setting the Standby Transition Timeout ... 22
8.3.2. Suspend Transition Prohibited Events ... 22
8.3.3. Setting the Auto Power-off Timeout ... 22
8.3.4. Setting the Effective Held-down Time of the Power Key for Switching to the Suspend State ... 22
Chapter 9. Battery State ... 23
9.1. Outline ... 23
9.2. Battery Voltage Acquisition ... 23
9.3. Battery Voltage Icon ... 23
9.4. Battery Voltage Warning ... 23
Chapter 10. Backup Battery ... 24

10.1. Outline 24
10.2. Service Life Warning 24
10.3. Window Message Notification 24
Chapter 11. LED 25
11.1. Outline 25
11.2. LED Control 25
11.2.1. Display LED 25
11.2.2. Charge LED 25
Chapter 12. Data Communication 26
12.1. Outline 26
12.2. Programming for Data Communication 26
12.3. ActiveSync 27
12.3.1. Establishing an ActiveSync Connection 27
12.3.2. ActiveSync Auto Connection Setting Method 27
Chapter 13. Wireless Communication 28
13.1. Outline 28
13.1.1. Spread Spectrum Communications Method 28
13.1.2. Configuration of Spread Spectrum System 28
13.2. Programming for Wireless Communication 29
13.2.1. Wireless Communication Parameters 30
13.2.2. Opening and Closing the Wireless Communications Device 32
13.2.3. Checking Synchronization with the Access Point 33
Chapter 14. Bar Code Reading 34
14.1. Outline 34
14.1.1. Enable Reading 34
14.1.2. Specify Options in the **BHT_EnableBar** Function 35
14.1.3. Barcode Buffer 35
14.2. Programming 36
14.2.1. Code Mark 36
14.2.2. Multiple Code Reading 36
14.2.3. Read Mode of the Trigger Switch 36
14.2.4. Generating a Check Digit of Barcode Data 36
14.2.5. Controlling the Indicator LED and Beeper/Vibrator as a Confirmation of Successful Reading 37
14.3. Barcode Reading Using the Virtual COM Port 38
14.3.1. Outline 38
14.3.2. Programming 38
14.3.3. How to Use 38
Chapter 15. System Messages 39
Chapter 16. Updating OS 41
Chapter 17. Starting the BHT 42
17.1. Setting up the BHT 42
17.2. Warm Boot / Cold Boot 42
17.3. Specifying the Reboot Modes in Application Programs 43
Chapter 18. System Functions 44
18.1. If a System Parameter Value is DWORD 45
18.2. If a System Parameter Value is a Character String 47
18.3. System Parameter Values That Can be Set/Obtained 49
Chapter 19. Device Control Functions 54
19.1. Barcode API 56
19.2. Backlight API 85
19.3. Battery API 87
19.4. LED API 89
19.5. Beeper/Vibrator API 94
19.6. Wireless Communication API 99
19.7. Flash File System API 113
19.8. OS Updating API 115
19.9. Bluetooth API 116
19.10. Other APIs 119
Chapter 20. Programming Using OCX (OLE Customer Control) 125
20.1. System Requirements 125
20.2. Installation 125

20.3. Using OCX 125
20.4. Scanner Control 128
20.4.1. Properties 128
20.4.2. Methods 129
20.4.3. Event Callback Function 130
20.4.4. Error Codes 131
20.4.5. Coding Sample 131
20.5. File Transfer Control 132
20.5.1. Properties 132
20.5.2. Methods 133
20.5.3. Event Callback Functions 138
20.5.4. Coding Sample 140
Chapter 21. Error Codes 141
Appendix A. Keyboard Arrangement, Virtual Key Codes and Character Codes 142
A.1. 31-key pad 142
A.1.1. Keyborard Arrangement 142
A.1.2. Virtual Key Codes and Character Codes 143
A.1.3. Character Codes in Alphabet Entry Mode 144
A.2. 50-key pad (Phone-type key layout) 145
A.2.1. Keyborard Arrangement 145
A.2.2. Virtual Key Codes and Character Codes 146
A.3. 50-key pad (Calculator-type key layout) 148
A.3.1. Keyborard Arrangement 148
A.3.2. Virtual Key Codes and Character Codes 149

Chapter 1. Software Requirements for the BHT-400

1.1. Operating System (OS) on the BHT-400

The OS running on the BHT-400 is Microsoft Windows CE 5.0.

1.2. Application Development Software on the PC

1.2.1. Application Development Tool

The application development tool for the BHT-400 is Microsoft eMbedded Visual C++ 4.0 (Service Pack 4)

1.2.2. Software Development Kit

The BHT-400 Software Development Kit provides the application development environment for Windows CE set up on the BHT-400. It includes the following libraries:

(1) Help files

(2) Windows standard header files

(3) Windows standard library files

(4) BHT-dedicated header file : BHTLIB.h

- Includes statements for declaring BHT-dedicated APIs prototypes and macro definition of constants.
- To use the BHT-dedicated APIs, the BHTLIB.h should be included.

(5) BHT-dedicated library : BHTLIB.lib

- Includes BHT-dedicated barcode reading functions and device driver management functions.
- To use the BHT-dedicated APIs, the BHTLIB.lib should be linked.

(6) BHT-dedicated OCX files : Scanner400.ocx, FileTransfer400.ocx, and FileTransferPC.ocx (for PC)

- Include BHT-dedicated barcode scanning functions and file transfer functions.
- To use the BHT-dedicated OCX, Scanner400.ocx, and FileTransfer400.ocx should be linked.

Chapter 2. Application Development Environment

2.1. Required Hardware (PC to be used for application development)

Item	Specification
OS	Microsoft Windows 2000 Professional with Service Pack 4 or higher, or Microsoft Windows 2000 Server with Service Pack 2 or higher, or Microsoft Windows XP Professional with Service Pack 1 or higher.
PC	With a Pentium-II class processor, 450 MHz or faster
Memory	For Microsoft Windows 2000 Professional or Microsoft Windows XP Professional: 96 MB or more (128 MB or more recommended)
	For Microsoft Windows 2000 Server : 192 MB or more (256 MB or more recommended)
HDD	200 MB or more hard disk space
Display	VGA or higher-resolution monitor. A Super VGA (800 x 600 or larger) monitor is recommended.

2.2. Required Software

Application development tool: Microsoft eMbedded Visual C++ 4.0 (SP4)

You can download Microsoft eMbedded Visual C++ 4.0 and Service Pack 4 from the Microsoft Web site:
(Microsoft eMbedded Visual C++ 4.0)
http://www.microsoft.com/downloads/details.aspx?FamilyID=1dacdb3d-50d1-41b2-a107-fa75ae960856&DisplayLang=en
(Service Pack 4)
http://www.microsoft.com/downloads/details.aspx?FamilyID=4a4ed1f4-91d3-4dbe-986e-a812984318e5&displaylang=en

APIs available for eMbedded Visual C++ are:
(1) Win32API
(2) Microsoft Foundation Class (MFC)
(3) Dedicated APIs (for device control or data entry from the BHT)

Software development kit: BHT400_XXX.msi
This should be embedded into Microsoft eMbedded Visual C++ 4.0 for use.

2.3. Installation

The Microsoft eMbedded Visual C++ 4.0 and BHT-400 software development kit should be installed to an application development PC in this order. To install the development kit, run the BHT400_XXX.msi in the BHT-400 Software Development Kit CD.

Chapter 3. Output to the LCD Screen

3.1. Screen Fonts

The BHT-400 has the following integrated screen fonts:

(1) Arial (ttf)
(2) Courier New (ttf)
(3) Tahoma (ttf)
(4) Time New Roman (ttf)
(5) Wingding (ttf)

If no screen font is specified, Tahoma applies automatically.

Chapter 4. Backlight Control

4.1. Outline

The backlight illumination and power saving modes can be controlled using either of the following methods.
(1) The backlight can be controlled by pressing the backlight control key.
(2) The backlight can be controlled using the backlight control function (**BHT_SetBltStatus**).

The following backlight related setting items are also available.
(1) Backlight control key
(2) Backlight illumination time
(3) Backlight brightness
(4) Backlight power saving mode

4.2. Setting the Backlight Function On/Off Key

You can assign the backlight function on/off key to other keys by the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_KEY...) function or by assigning the backlight control function to the magic key.The table below lists the relationship between the keys that act as a backlight function on/off key and the set values in the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_KEY...) function.
If no key is specified as a backlight function on/off key, the combination of the SF key and M4 key works as a backlight function on/off key by default.

Backlight control key	Set value	Backlight control key	Set value
[SCAN]	0x00000200	[F1]	0x00000101
[M1]	0x00000201	[F2]	0x00000102
[M2]	0x00000202	[F3]	0x00000103
[M3H]	0x00000243	[F4]	0x00000104
[M3]	0x00000203	[F5]	0x00000105
[M4H]	0x00000244	[F6]	0x00000106
[M4]	0x00000204	[F7]	0x00000107
[M5H]	0x00000245	[F8]	0x00000108
[M5]	0x00000205	[F9]	0x00000109
[SF]+[SCAN]	0x00010200	[F10]	0x0000010A
[SF]+[M1]	0x00010201	[F11]	0x0000010B
[SF]+[M2]	0x00010202	[F12]	0x0000010C
[SF]+[M3H]	0x00010243		
[SF]+[M3]	0x00010203		
[SF]+[M4H]	0x00010244		
[SF]+[M4]	0x00010204		
[SF]+[M5H]	0x00010245		
[SF]+[M5]	0x00010205		

NOTE: The "M3H," "M4H," and "M5H" represent M3, M4, and M5 keys halfway depressed, respectively. The "M5" and "M5H" keys are available only to the BHT connected with the grip.

[Ex]

Execute function **BHT_SetSysSettingDW** (BHT_BACKLIGHT_KEY, 0x00010201) when assigning a simultaneous combination of the [SF] and [M1] keys to the backlight control key.

4.3. Setting the Backlight Illumination Time

The backlight illumination time is set and read using the
BHT_SetSysSettingDW (DWORD dwCtrlCode,DWORD dwSysParam) and
BHT_GetSysSettingDW (DWORD dwCtrlCode,DWORD *pdwSysParam) functions.

Parameter	Type	R/W	Control Code (dwCtrlCode)	Parameter Value (dwSysParam)	Default	Validation Timing
Illumination time when powered by battery (sec.)	DW	R/W	BHT_BACKLIGHT _BATT_TIME	0 - 255 0: Backlight OFF 255: Continuously ON	3	When backlight illumination timer is next reset
Illumination time when placed on CU (sec.)	DW	R/W	BHT_BACKLIGHT _AC_TIME	0 - 255 0: Backlight OFF 255: Continuously ON	60	When backlight illumination timer is next reset

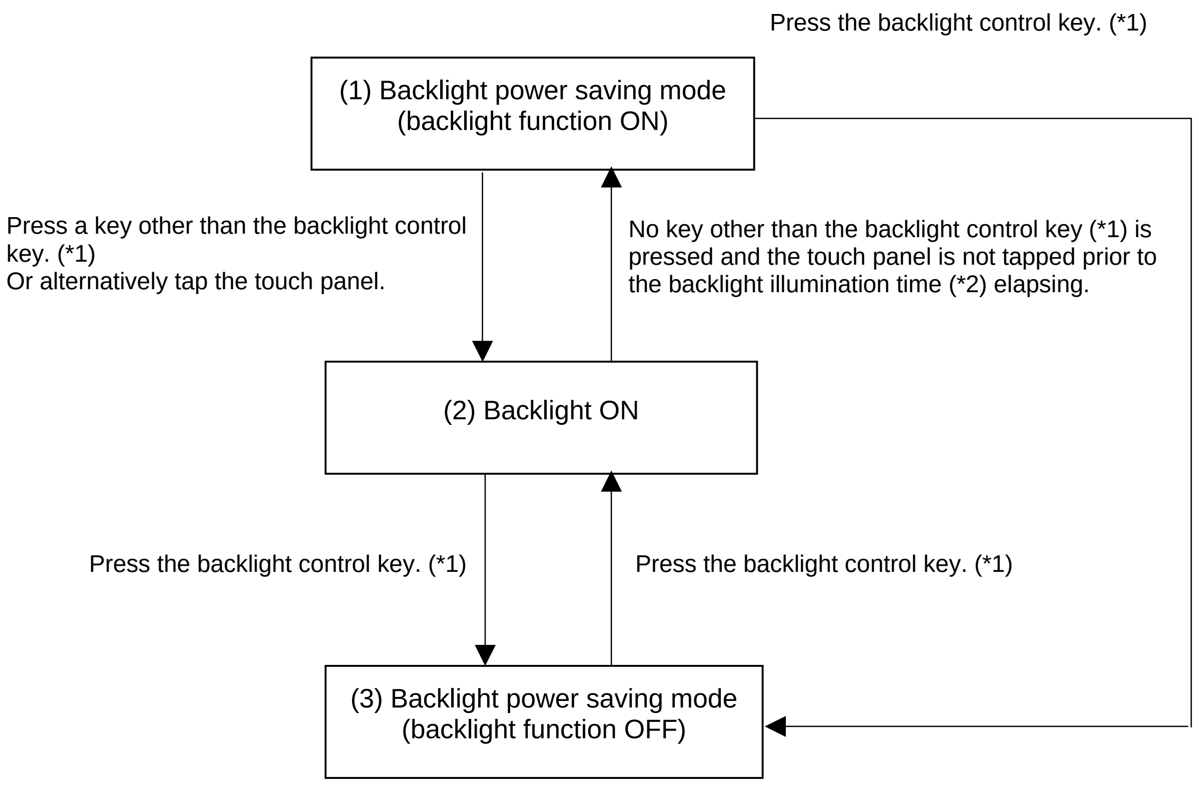
4.4. Setting the Backlight Brightness and Power Saving Mode

The backlight brightness and power saving mode are set and read using the
BHT_SetSysSettingDW (DWORD dwCtrlCode,DWORD dwSysParam) and
BHT_GetSysSettingDW (DWORD dwCtrlCode,DWORD *pdwSysParam) functions.

Parameter	Type	R/W	Control Code (dwCtrlCode)	Parameter Value (dwSysParam)	Default	Validation Timing
Backlight brightness	DW	R/W	BHT_BACKLIGHT _BRIGHTNESS	0: OFF 1: Dark 2: Bright (low) 3: Bright (high)	3	When the backlight is next turned ON
Backlight power saving mode	DW	R/W	BHT_BACKLIGHT _POWERSAVE	0: OFF 1: Dim	1	When power saving mode is next enabled

4.5. Controlling the Backlight with the Backlight Control Key

The backlight function can be enabled/disabled by pressing the backlight function control key (Default: Hold down [SF] key and press [M4].).
The illumination time is specified using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_BATT_TIME/BHT_BACKLIGHT_AC_TIME, …) function. The default value is 3 seconds when powered by the battery, and 60 seconds when placed on the CU. Backlight control is performed as shown in the flow diagram below.


Press the backlight control key. (*1)
(1) Backlight power saving mode (backlight function ON)
Press a key other than the backlight control key. (*1) Or alternatively tap the touch panel.
No key other than the backlight control key (*1) is pressed and the touch panel is not tapped prior to the backlight illumination time (*2) elapsing.
(2) Backlight ON
Press the backlight control key. (*1)
Press the backlight control key. (*1)
(3) Backlight power saving mode (backlight function OFF)

(*1)
Default: Hold down [SF] key and press [M4].
Setting is possible using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_KEY,…) function.

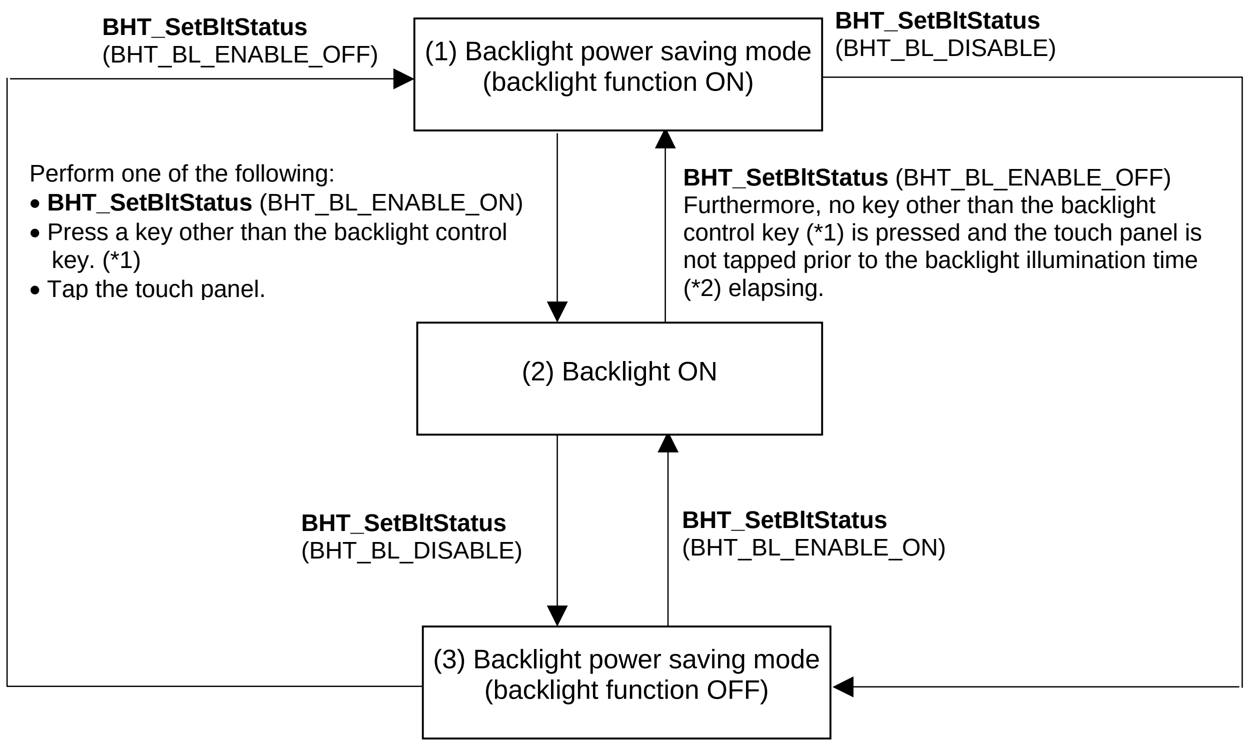
(*2)
The backlight illumination time is set using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_BATT_TIME/ BHT_BACKLIGHT_AC_TIME,…) function. Power saving mode is enabled if no key other than the backlight control key is pressed, or if the touch panel is not tapped within this time.
This time is measured from the point all keys are released or the touch panel is last pressed.

(*3)
Cold booting is performed from the status at (1) above.
However, cold booting is performed from the status at (1) when the registry is saved with the status at (1) or (2), and is performed from the status at (3) when the registry is saved with the status at (3).

(*4)
When performing warm booting or when resuming from the suspend status, the process is performed from (1) if the status prior to warm boot/suspend is (1) or (2), and is performed from (3) if the status prior to warm boot/suspend is (3).

4.6. Controlling the Backlight with the Backlight Control Function

The backlight function can be controlled using the **BHT_SetBltStatus** function.
The **BHT_SetBltStatus** (BHT_BL_ENABLE_ON) function is used to enable the backlight function and turn the backlight ON.
The backlight power saving mode is enabled if no keys are pressed, or the touch panel tapped from the point the backlight is turned ON using the **BHT_SetBltStatus** (BHT_BL_ENABLE_ON) function until the time set using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_BATT_TIME/BHT_BACKLIGHT_AC_TIME,…) function (Default: 3 seconds when powered by battery, 60 seconds when placed on CU) elapses, or if the **BHT_SetBltStatus** (BHT_BL_ENABLE_OFF) function is executed. (The backlight function remains ON at this time.)
If the **BHT_SetBltStatus** (BHT_BL_DISABLE) function is executed, the backlight function is disabled, and the backlight power saving mode is enabled.
Backlight control is performed as shown in the flow diagram below.


BHT_SetBltStatus
(BHT_BL_ENABLE_OFF)
(1) Backlight power saving mode
(backlight function ON)
BHT_SetBltStatus
(BHT_BL_DISABLE)
Perform one of the following:
• BHT_SetBltStatus (BHT_BL_ENABLE_ON)
• Press a key other than the backlight control key. (*1)
• Tap the touch panel.
BHT_SetBltStatus (BHT_BL_ENABLE_OFF)
Furthermore, no key other than the backlight control key (*1) is pressed and the touch panel is not tapped prior to the backlight illumination time (*2) elapsing.
(2) Backlight ON
BHT_SetBltStatus
(BHT_BL_DISABLE)
BHT_SetBltStatus
(BHT_BL_ENABLE_ON)
(3) Backlight power saving mode
(backlight function OFF)

(*1)
Default: Hold down [SF] key and press [M4].
Setting is possible using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_KEY,…) function.

(*2)
The backlight illumination time is set using the **BHT_SetSysSettingDW** (BHT_BACKLIGHT_BATT_TIME/ BHT_BACKLIGHT_AC_TIME,…) function. Power saving mode is enabled if no key other than the backlight control key is pressed, or if the touch panel is not tapped within this time.
This time is measured from the point all keys are released or the touch panel is last pressed.

(*3)
Cold booting is performed from the status at (1) above.
However, cold booting is performed from the status at (1) when the registry is saved with the status at (1) or (2), and is performed from the status at (3) when the registry is saved with the status at (3).

(*4)
When performing warm booting or when resuming from the suspend status, the process is performed from (1) if the status prior to warm boot/suspend is (1) or (2), and is performed from (3) if the status prior to warm boot/suspend is (3).

Chapter 5. Beeper and Vibrator Control

5.1. Outline

The beeper and vibrator are controlled by:

(1) the beeper/vibrator setting functions
(that allow you to choose beeper and/or vibrator and set the beeper volume. Refer to Section 5.2.)

(2) the beeper/vibrator start/stop functions
(that allow you to set the beeping or vibration interval, the number of repetitions, and frequency. Refer to Section 5.3.)

5.2. Setting the Beeper/Vibrator

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam)
and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the beeper/vibrator parameters as specified below.

Parameter name	Type	R/W	Control code (dwCtrlCode)	Parameter value (dwSysParam)	Default	Validating timing
Rumble device	DW	R/W	BHT_BEEP_VIB _SELECT	BEEP_SELECT : Beeper VIB_SELECT : Vibrator BEEP_SELECT | VIB_SELECT : Beeper and vibrator	BEEP_SELECT	Immediately after setting
Beeper volume (*1)	DW	R/W	BHT_BEEP_VIB _VOLUME	0: OFF 1 (Lowest) to 5 (Highest)	5	Immediately after setting
Key clicks (*2)	DW	R/W	BHT_BEEP_VIB _KEY	0: OFF 1 (Soft) 2 (Loud)	2	Immediately after setting
Screen taps	DW	R/W	BHT_BEEP_VIB _TAP	0: OFF 1 (Soft) 2 (Loud)	2	Immediately after setting
Half-pressed key clicks (*3)	DW	R/W	BHT_BEEP_VIB _HALFKEY	0: OFF 1 (Soft) 2 (Loud)	0	Immediately after setting
Trigger switch clicks (*4)	DW	R/W	BHT_BEEP_VIB _TRGKEY	CLICK_SOUND_OFF: Prohibit CLICK_SOUND_ON: Permit	CLICK_SOUND_OFF	Immediately after setting
Laser lighting key clicks	DW	R/W	BHT_BEEP_VIB _LASERKEY	CLICK_SOUND_OFF: Prohibit CLICK_SOUND_ON: Permit	CLICK_SOUND_OFF	Immediately after setting

(*1) This setting is effective only when the value 0, 1, or 2 is specified to the frequency in the beeper start/stop functions (**BHT_StartBeep** or **BHT_StartBeeperOnly**).
(*2) When “trigger switch click sound” is OFF, this setting is not applicable to the fully-pressed magic key which is assigned the trigger switch or halfway-pressed keys.
(*3) When “trigger switch click sound” is OFF, this setting is not applicable to the halfway-pressed magic key which is assigned the trigger switch.
(*4) This setting is effective only for fully- or halfway-pressed magic key which is assigned the trigger switch.

The rumble device specification above takes effect when the beeper/vibrator is driven:
(1) by the **BHT_StartBeep** function.
(2) due to low battery warning, in conjunction with the "Battery voltage has lowered." or "Charge the Battery!" message.
(3) upon completion of barcode reading.
(4) by the MessageBox, MessageBeep, PlaySound of the Windows CE compliant API.

The sound pattern of the key clicks, screen taps, and trigger switch clicks is as follows:
ON-duration: 10 ms
Frequency: 1396 Hz
Volume : Loud, Soft

5.3. Starting/Stopping the Beeper/Vibrator

The beeper/vibrator is activated or deactivated by the following functions:

Function	Used to:
BHT_StartBeep	Activate the selected device (beeper or vibrator).
BHT_StartBeeperOnly	Activate the beeper.
BHT_StartVibratorOnly	Activate the vibrator.

The functions listed above start the beeper/vibrator control and immediately pass control to the subsequent statement or function. The actual device operation is carried out in background processing. The functions listed above do not suspend execution of the subsequent same functions until the device(s) completes the specified operation. Instead, the execution of the subsequent functions proceed immediately.
Calling a second function when the target device is still operating by a first function stops the device and operates it under the newly specified conditions after checking the parameter values.

Specifying the frequency with value 0, 1, or 2 sounds the beeper with the frequency listed below. If any other value is specified, the beeper sounds at the maximum volume.

Parameter value	Frequency (Hz)
0	698
1	1396
2	2793

If the suspend or critical power states are turned OFF while the beeper is sounding or the vibrator is vibrating, the BHT resumes with both the beeper and vibrator stopped when the unit is next resumed.

5.4. Priority Orders between Events that Activate the Beeper/Vibrator

There are priority orders between events that activate the beeper/vibrator as listed below.

Priority	Event that activate the beeper/vibrator
Higher	System error
↕	Completion of bar code reading
	Setting in applications
Lower	Key clicks or screen taps

When the beeper or vibrator is being driven by any event, the lower priority event (if happens) activates no beeper or vibrator but the same or higher priority event (if happens) overrides the currently operating beeper or vibrator and newly activates the beeper or vibrator.

5.5. Beeper Volume Patterns

The beeper is activated according to the beeper volume as listed below.

Beeper volume	Volume
1 (lowest)	Soft
2	
3	Mid
4	
5 (highest)	Loud

Chapter 6. Keys and Trigger Switch Control

6.1. Outline

In addition to the processing for depressed or released keys and trigger switch, the BHT OS controls the following functions assigned to them.

(1) Specifying the shift key operation mode
(2) Assigning special key functions to the magic keys (M1 to M5) and Scan key.
(3) Supporting the alphabet entry mode (in addition to the numeric entry mode)
(4) Function mode
(5) Key click sound
(6) Keyboard type acquisition

Furthermore, both the 31-key pad and 50-key pad keyboard types are supported.

6.2. Setting the Keys and Trigger Switch

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the keys and trigger switch parameters.

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Shift key operation mode	DW	R/W	BHT_KEY _SHIFT_MODE	KEY_NON_LOCK : Non-lock mode KEY_ONE_TIME : Onetime lock mode	KEY_NON_LOCK	Immediately after setting
Assignment to M1 key	DW	R/W	BHT_KEY _M1_MODE	MAGIC_FUNC_NONE : Ignore the depressed key MAGIC_FUNC_ENTER : Treat as ENT key MAGIC_FUNC_TRG : Treat as trigger switch MAGIC_FUNC_SHIFT : Treat as SF key MAGIC_FUNC_ALT : Treat as ALT key MAGIC_FUNC_CTRL : Treat as CTRL key MAGIC_FUNC_BLT : Treat as backlight function on/off key MAGIC_FUNC_TAB : Treat as TAB key MAGIC_FUNC_LASER : Treat as laser lighting key MAGIC_FUNC_CLEAR : Treat as CLEAR key	MAGIC_FUNC_TAB	Immediately after setting
Assignment to M2 key	DW	R/W	BHT_KEY _M2_MODE		MAGIC_FUNC_NONE	Immediately after setting
Assignment to M3H key (M3 half-pressed)	DW	R/W	BHT_KEY _M3H_MODE		MAGIC_FUNC_LASER / MAGIC_FUNC_TRG (*1)	Immediately after setting
Assignment to M3 key	DW	R/W	BHT_KEY _M3_MODE		MAGIC_FUNC_TRG	Immediately after setting
Assignment to M4H key (M4 half-pressed)	DW	R/W	BHT_KEY _M4H_MODE		MAGIC_FUNC_LASER / MAGIC_FUNC_TRG (*1)	Immediately after setting
Assignment to M4 key	DW	R/W	BHT_KEY _M4_MODE		MAGIC_FUNC_TRG	Immediately after setting
Assignment to M5H key (M5 half-pressed)	DW	R/W	BHT_KEY _M5H_MODE		MAGIC_FUNC_LASER / MAGIC_FUNC_TRG (*1)	Immediately after setting
Assignment to M5 key	DW	R/W	BHT_KEY _M5_MODE		MAGIC_FUNC_TRG	Immediately after setting
Assignment to SCAN key	DW	R/W	BHT_KEY _SCAN_MODE		MAGIC_FUNC_TRG	Immediately after setting
Entry mode	DW	R/W	BHT_KEY _INPUT_METHOD	INPUT_METHOD _NUMERIC : Numeric entry mode INPUT_METHOD _ALPHABET : Alphabet entry mode	INPUT_METHOD _NUMERIC	Immediately after setting
Enable/disable alphabet entry switching key	DW	R/W	BHT_DISABLE _KEYMODE _CHANGE_KEY	ENABLE_KEY _TOCHANGE _ALPHABET : Enable alphabet entry DISABLE_KEY _TOCHANGE_ALPHABET : Disable alphabet entry	ENABLE_KEY _TOCHANGE _ALPHABET	Immediately after setting
Function mode	DW	R/W	BHT_KEY _FUNCTION	KEY_FUNCTION_ON : Function mode KEY_FUNCTION_OFF : Non-function mode	KEY_FUNCTION_OFF	Immediately after setting

(*1) The default value for the model without marker is "MAGIC_FUNC_TRG".

6.3. Shift Key Operation Mode

The shift key operation mode works as follows:

Shift key operation mode	Description
Non-lock mode	- The keypad is shifted when the Shift key is held down.
Onetime lock mode	- The shift status is cleared immediately after releasing a key when in the shift status from the time the key is pressed until it is released while the shift key is held down and after it is released.

6.4. Magic Key Control

The table below lists the virtual key codes and character codes of the magic keys (M1 to M5) or Scan key fully or half-depressed.

Parameter value	Virtual key code			Character code	
	Constant		Value	When not shifted	Shifted
MAGIC_FUNC_NONE	[M1] key	VK_M1	C1	-	-
	[M2] key	VK_M2	C2	-	-
	[M3] key	VK_M3	C3	-	-
	[M3H] key	VK_M3H	C8	-	-
	[M4] key	VK_M4	C4	-	-
	[M4H] key	VK_M4H	C9	-	-
	[M5] key	VK_M5	C5	-	-
	[M5H] key	VK_M5H	CA	-	-
	[SCAN] key	VK_SCAN	D1	-	-
MAGIC_FUNC_ENTER	VK_RETURN		0D	0D(CR)	0D(CR)
MAGIC_FUNC_TRG	(*1)			-	-
MAGIC_FUNC_SHIFT	VK_SHIFT		10	-	-
MAGIC_FUNC_CTRL	VK_CONTROL		11	-	-
MAGIC_FUNC_ALT	VK_MENU		12	-	-
MAGIC_FUNC_BLT	(*1)			-	-
MAGIC_FUNC_TAB	VK_TAB		09	09 (tab)	09 (tab)
MAGIC_FUNC_LASER	(*1)			-	-
MAGIC_FUNC_CLEAR	VK_CLEAR		0C	-	-

(*1) The same virtual key code as the one assigned with “MAGIC_FUNC_NONE” is returned.

6.5. Assigning a User-Defined Key Code to the Magic Keys

Apart from the previously mentioned functions, optional keys can be applied to the magic keys following the method below.
With this function it is possible to assign keys to the magic keys that do not exist in the BHT-400, or to execute the equivalent of a multi-key function by pressing a magic key once.

6.5.1. Assignment Method

<u>The steps for setting user-defined key codes for the magic keys are as follows:</u>

(1) Save a user-defined code settings file with the filename “MKeyDef.txt” in the FLASH folder of the BHT.
(2) Choose the key you wish to set from the key definition menu in the BHTShell (for further details refer to the “BHT-400B/400BW-CE User’s Manual”).
(3) Backup files can be created with a backup registry.

6.5.2. User-Defined Code Settings File (MKeyDef.txt)

(1) File name
“MKeyDef.txt” (fixed)
(2) Format
<Character string inside the combo box>,<Defined code number>,<Defined code 1>,…,<Defined code 4>

Item	Display Method	Setting Content
Character string inside the combo box	Character string	A character string containing up to 64 characters. Extra characters will be ignored.
Defined code number	decimal number	A user-defined code specified as a number between 1 and 4.
Defined code 1 through 4	hexadecimal number	The virtual key code you wish to assign.

[Ex] Setting a user-defined key code of “Alt + X” and “Alt + Y” to be added to the combo box list.

```
ALT+X, 2, 0x12, 0x58
ALT+Y, 2, 0x12, 0x59
```

(*) If there is a mistake in the format of a line in the MKeyDef.txt file, that line will be ignored and removed from the BHTShell key definition menu.
(*) Even if the MKeyDef.txt file is deleted, key code settings will be retained (the BHTShell will display "None"). When a different function is designated in the BHTShell, the previous key code settings will be replaced.

6.6. Key Input Modes

The BHT 31-key pad has the following two key entry modes.

(1) Numeric entry mode

This mode allows you to type in numeric data with the numeric keys.

(2) Alphabet entry mode

Use the numeric keys to type in alphabet letters in the same way as he/she uses a cellular phone.

The 50-key pad has no key entry mode, and permits entry of both numeric and alphabet characters at all times.
Alphabet characters are entered in upper case by default, and can be entered in lower case by holding down the Shift key.

6.6.1. Numeric Entry Mode

This mode is the default when the BHT-400 is turned on.

The numeric entry mode starts by:

(1) calling the **BHT_SetSysSettingDW** (BHT_KEY_INPUT_METHOD, INPUT_METHOD_NUMERIC) function.

(2) pressing the ALP key*1 in the alphabet entry mode.

*1 The key takes effect only when it is not disabled by the BHT_DISABLE_KEYMODE-CHANGE_KEY.

Pressing keys in this mode returns virtual key codes and character codes specified in Appendix A.

6.6.2. Alphabet Entry Mode

The alphabet entry mode starts by:

(1) calling the **BHT_SetSysSettingDW** (BHT_KEY_INPUT_METHOD, INPUT_METHOD_ALPHABET) function.

(2) pressing the ALP key*2 in the numeric entry mode.

The alphabet entry mode terminates by:

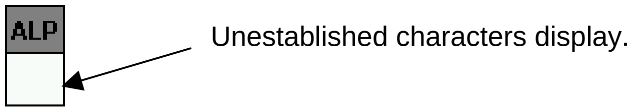
(1) switching to any other entry mode with the **BHT_SetSysSettingDW** function.

(2) pressing the ALP key*2 in the numeric entry mode.

*2 The key takes effect only when it is not disabled by the BHT_DISABLE_KEYMODE-CHANGE_KEY.

In the 31-key pad alphabet entry mode, alphabet characters can be entered using an alphabet character similar to that used on a cellular phones.

(1) When changing to alphabet entry mode, an unestablished character display window similar to that shown below displays.



The unestablished character display window has the following features.

- This window can be moved by using the stylus.
- When the unestablished character is a space, “SP” displays in order to distinguish between those times when there are no unestablished characters.
- The focus is not transferred to the unestablished character display window.
- The unestablished character display window always displays in the foreground.

Furthermore, the following icon displays in the task bar when in alphabet entry mode.

(2) If keys [0] to [9] or the [.] key is pressed, the pressed key becomes an unestablished character and displays in the unestablished character display window. The character then reverts to a character code when any of these keys becomes established.
Press any of the following keys below to establish unestablished characters.

- Keys [0] to [9] or [.] that differ from the key pressed at the unestablished character
- [ENT] key
- “MAGIC_FUNC_ENTER” assigned to the magic/scan keys
- Keys [F1] to [F12]

(3) Keys used for alphabet entry
The table below lists keys whose operations are different from those in the numeric entry mode.

Use this key	To do this
0 to 9 and period (.) keys	Enter alphabets. For alphabets assigned to these keys, refer to “Appendix A. Keyboard Arrangement, Virtual Key Codes and Character Codes” – “A.1.3. Character Codes in Alphabet Entry Mode.”
ENT key	Establish an unestablished key if any. If there is no unestablished key, the same character code as in the numeric entry mode is returned.
BS key	Clear an unestablished key if any. If there is no unestablished key, the same character code as in the numeric entry mode is returned.
CLR key	
F1 to F12 Key	Establish an unestablished key if any. If there is no unestablished key, the same character code as in the numeric entry mode is returned.
Magic key and SCAN key	Establish an unestablished key if any when the MAGIC_FUNC_ENTER is assigned to these keys. If there is no unestablished key, the same character code as in the numeric entry mode is returned.
ALP key	Clears unestablished keys if any exist and switches to numeric entry mode.

6.7. Function Mode

Use either of the methods below to enable function mode.
(1) Call up the **BHT_SetSysSettingDW** (BHT_KEY_FUNCTION,KEY_FUNCTION_ON) function.
(2) Press the [FUNC] key when in function mode.

Use either of the methods below to disable function mode and return to non-function mode.
(1) Call up the **BHT_SetSysSettingDW** (BHT_KEY_FUNCTION,KEY_FUNCTION_OFF) function.
(2) Press the [FUNC] key when in function mode.

Non-function mode is enabled as the default when the unit is booted up.
The following icon displays in the task bar when in function mode.

F

If a key is pressed when in function mode, a virtual key code or character code is returned as outlined in ”Appendix A. Keyboard Arrangement, Virtual Key Codes, and Character Codes”.

6.8. Key Clicks

When the keys are pressed, the BHT clicks as specified below. Note that pressing the power key does not click.

Parameter name	Type	R/W	Control code (dwCtrlCode)	Parameter value (dwSysParam)	Default	Validating timing
Key click volume	DW	R/W	BHT_BEEP_VIB_KEY	0: OFF 1: Soft 2: Loud	2	Immediately after setting
Half-pressed key click volume	DW	R/W	BHT_BEEP_VIB_HALFKEY	0: OFF 1: Soft 2: Loud	0	Immediately after setting
Trigger switch clicks	DW	R/W	BHT_BEEP_VIB_TRGKEY	CLICK_SOUND_OFF: Prohibit CLICK_SOUND_ON: Allow	CLICK_SOUND_OFF	Immediately after setting
Laser lighting key clicks	DW	R/W	BHT_BEEP_VIB_LASERKEY	CLICK_SOUND_OFF: Prohibit CLICK_SOUND_ON: Allow	CLICK_SOUND_OFF	Immediately after setting

6.9. Acquisition of Keypad Type

The **BHT_GetSysSettingDW** (DWORD dwCtrlCode,DWORD *pdwSysParam) function reads the keypad type.

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Keypad type	DW	R	BHT_KEYBOARD_TYPE	KEYBOARD_TYPE1 : 31-key pad KEYBOARD_TYPE2 / KEYBOARD_TYPE2P : 50-key pad (Phone-type key layout) KEYBOARD_TYPE2C : 50-key pad (Calculator-type key layout)	-	-

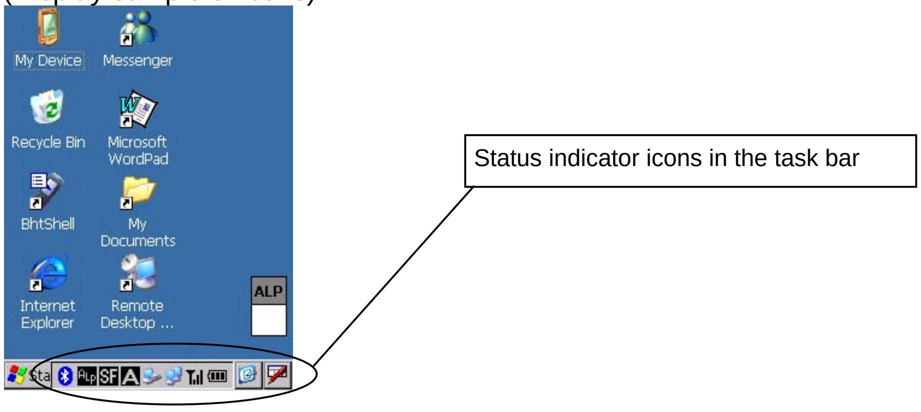
Chapter 7. LCD Status Indication

7.1. Outline

The status of the BHT is displayed on the LCD as specified below.

Status	Description	Icon
Battery voltage level	Displays the battery voltage in five levels.	🔋
Software keyboard state	Shows whether the software keyboard is displayed or hidden. Tapping this icon toggles the software keyboard on and off.	A : The software keyboard is displayed. A : The software keyboard is hidden.
Keypad shift state	Displays the icon when the keypad is shifted.	SF
Function mode state	Displays the icon when in function mode.	F
Alphabet input state	Displays the ALP window when the alphabet input function is activated. An unestablished character appears in this ALP window. (Models with 31-key pad only support this icon.)	ALP
	Displays the icon when the alphabet input function is activated.	ALp
Standby state	Displays this icon when the CPU comes to be on standby.	zzz
Synchronization state	Displays the open state of the wireless device and the radio field intensity.	The wireless device is open. The wireless device is open and the wireless link is established with an access point. : Radio field intensity (Low) : Radio field intensity (Medium) : Radio field intensity (High)
ActiveSync	Displays this icon when the BHT is communicating with the PC via Microsoft ActiveSync (not using wireless).	
Desktop display	Switches the screen between the application execution display and desktop display. Tapping this icon when an application program is running switches the screen to the desktop display. Tapping it again returns to the application execution display.	
Bluetooth power status	Displays the Bluetooth power status. No icons display if the unit is not equipped with a Bluetooth device.	: Power ON : Power OFF

(Display sample of icons)



7.2. Setting the LCD Status Indication

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the LCD status indication as specified below.

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Battery voltage level icon	DW	R/W	BHT_ICON _BATTERY	0: Hide 1: Display	1	Immediately after setting
Software keyboard icon	DW	R/W	BHT_ICON _SIP	0: Hide 1: Display	0	Immediately after setting
Keypad shift icon	DW	R/W	BHT_ICON _SHIFTKEY	0: Hide 1: Display	1	Immediately after setting
Alphabet input icon	DW	R/W	BHT_ICON _IN_ALPHA	0: Hide 1: Display	1	Immediately after setting
Synchronization state icon	DW	R/W	BHT_ICON _RADIO_INTENSE	0: Hide 1: Display	1	Immediately after setting
Standby state icon	DW	R/W	BHT_ICON _STANDBY	0: Hide 1: Display	0	Immediately after setting
Function moe state icon	DW	R/W	BHT_ICON_FUNC	0: Hide 1: Display	1	Immediately after setting
Bluetooth power status	DW	R/W	BHT_ICON _BLUETOOTH	0: Hide 1: Display	0	Immediately after setting

Chapter 8. Power Management

8.1. Outline

The power management functions switch the system powering state.
The following four system power states exist.
(1) Power ON
(2) Standby
(3) Suspned (*1)
(4) Critical OFF (*2)

(*1) Suspend
The BHT will be suspended when the power is turned off with the power key or auto power off feature.
(*2) Critical OFF
The BHT will become critical off when the power is turned off due to battery voltage drop or battery cover unlocked.

Notes
- No processing is performed when the BHT is on standby.
- When the CompactFlash card is used, disable the standby function before accessing the card.

8.2. Standby

8.2.1. Switching to the Standby State

The BHT switches from the power ON state to the standby state when any of the following conditions arises:

(1) When the standby transition timeout occurs after a standby transition prohibited event (listed below) is completed.
(2) When waiting for the event specified by the **BHT_WaitStandbyEvent** function with the standby transition prohibited event completed.
(3) When the standby transition prohibited event is completed while waiting for the event specified by the **BHT_WaitStandbyEvent** function to occur.

8.2.2. Standby Transition Prohibited Events

The following items are standby transition prohibited events.

- Key being pressed
- Touch panel being tapped
- Screen being refreshed
- Beeper/vibrator in operation
- Key click sound/touch panel tap sound in operation
- Backlight being ON (excludes those times when continuously ON)
- Reading bar codes
- IrDA interface port opened
- Connector interface port opened
- USB interface opened
- Wireless device opened
- Flash memory being erased or written
- RTC being accessed
- Indicator LED being ON
- System message being displayed
- Bluetooth device power being ON
- Standby transition time set to "0"

8.2.3. Setting the Standby Transition Timeout

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the standby transition timeout as specified below.

Parameter name	Type	R/W	Control code	Parameter value	Defaults	Validating timing
Standby transition timeout (in units of 100 msec)	DW	R/W	BHT_PM_STBYTIME	0: Disable 1 - 255	10 (1 sec)	Immediately after setting

8.3. Suspend

8.3.1. Setting the Standby Transition Timeout

The BHT switches to the suspend state when any of the following conditions arises:
(1) When the power is on, the power key is held down for the effective held-down time (for switching to the suspend state) or more.
(2) An auto power-off timeout occurs after one of the suspend transition prohibited events (listed below) is completed.
(3) When the power OFF function is called.

8.3.2. Suspend Transition Prohibited Events

The following items are suspend transition prohibited events.
- Key press (other than power key) authentication
- Touch panel tap authentication
- When ActiveSync connection established (IrDA, RS-232C and USB)
- When auto power OFF time is set to "0"
- When the following registry value is set to "0" with a wireless connection established

```
[HKEY_LOCAL_MACHINE\Comm\CXPort]
  "NoIdleTimerReset"=dword : 0
```

Furthermore, the auto power OFF time is reset upon the occurrence of the following events.
- When a serial communication event occurs (IrDA, RS-232C and USB)
- When a PCMCIA IREQ interruption occurs
- When the SystemIdleTimerReset() function is executed
- When an event with event object name "PowerManager, ActivityTimer, or UserActivity" is set

8.3.3. Setting the Auto Power-off Timeout

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the auto power-off timeout as specified below.

Parameter name	Type	R/W	Control code	Parameter value	Defaults	Validating timing
Auto power-off timeout (sec.) (When battery-driven)	DW	R/W	BHT_PM _BATTPOWEROFF	0: Disable 1 - 0xFFFFFFFF	180 (3 min.)	Immediately after setting
Auto power-off timeout (sec.) (When placed on the CU)	DW	R/W	BHT_PM _EXTPOWEROFF	0: Disable 1 - 0xFFFFFFFF	0	Immediately after setting

8.3.4. Setting the Effective Held-down Time of the Power Key for Switching to the Suspend State

The **BHT_SetSysSettingDW** (DWORD dwCtrlCode, DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode, DWORD *pdwSysParam) functions write or read the effective held-down time of the power key for switching to the suspend state as specified below.

Parameter name	Type	R/W	Control code	Parameter value	Defaults	Validating timing
Effective held-down time of the power key for switching to the suspend state (in units of 100 msec)	DW	R/W	BHT_PWRDOWN_KEY _WAIT_TIME	1 - 255	5	Immediately after setting

<u>Saving the Registry</u>
If the BHT is switched to the suspend state by pressing the power key with the SF (*1) key held down, the Registry will be saved into the flash memory.
(*1) Here, this means only the key marked “SF.” The Registry will not be saved even if you press the power key while holding down the magic key to which the SF key function is assigned.

Chapter 9. Battery State

9.1. Outline

The battery status can be ascertained using the following methods.
(1) Battery status acquisition
(2) Battery voltage icon
(3) Low battery voltage warning message display

9.2. Battery Voltage Acquisition

The **BHT_GetPowerStatus** function can be used to ascertain whether the BHT is on the CU, and acquires the battery level, battery voltage, and battery type.

9.3. Battery Voltage Icon

The battery voltage status is indicated with the icons below if the battery voltage status display is authorized.

Battery voltage level		Battey Voltage Icon
Level	Voltage	
High	3.9 V or higher	[▮▮▮]
Medium	3.7 V or higher and less than 3.9 V	[▮▮]
Low	3.6 V or higher and less than 3.7 V	[▮]
Warning	Less than 3.6 V	[▯]

9.4. Battery Voltage Warning

If the output voltage of the battery cartridge drops below the specified lower limit, the BHT displays the Level-1 message "Battery voltage has lowered." on the LCD and beeps three times. After that, it will resume the previous regular operation.

If the battery output voltage drops further, the BHT displays the Level-2 message "Charge the battery!," beeps five times, and then turns itself off automatically.

Chapter 10. Backup Battery

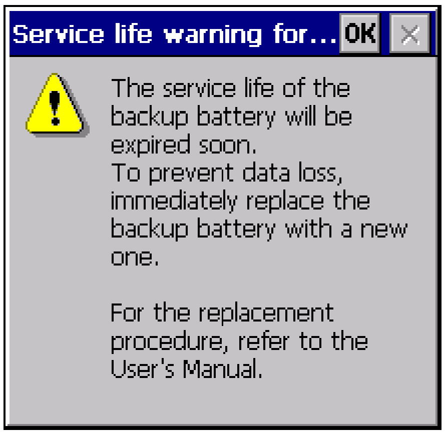
10.1. Outline

The backup battery has a service life determined by the number of full discharge times. To prompt the user to replace it, the BHT OS controls the following:

If the battery is fully discharged:	The BHT:
Less than 200 times	Performs no processing.
200 times or more	Notifies the user with a warning display and window message each time the power is turned ON. (cold-boot/warm-boot, or power on from the suspend or critical OFF state)

10.2. Service Life Warning

When the discharge count reaches 200 times or more, the following warning message displays, the beeper sounds 5 times (each beep sound lasts for 0.1 seconds), and the power then turns ON as normal.



The service life of the backup battery will be expired soon.
To prevent data loss, immediately replace the backup battery with a new one.

For the replacement procedure, refer to the User's Manual.

Warning message

10.3. Window Message Notification

The following window message is posted when the battery is discharged 200 times or more.

Parameter		Type	Detail
Registered message character string		LPCTSTR	MSG_BHT_WARNING (=TEXT("BHTWarning"))
Message value		UINT	**RegisterWindowMessage()** function return value
Additional message information	wParam	DWORD	0 (Indicates a backup battery service life warning.)
	lParam	DWORD	Discharge count

Chapter 11. LED

11.1. Outline

The BHT-400 has two LEDs. The display LED can be controlled from the application.

LED	Color	ON/OFF control from applications
Indicator LED	Red and blue	Possible
Charger LED	Red and green	Impossible

11.2. LED Control

11.2.1. Display LED

(1) Control method

The red and blue display LEDs can be turned ON and OFF using the **BHT_SetNLedStatus**, **BHT_SetNLedOn**, and **BHT_SetNLedOff** functions.
Furthermore, the LED ON/OFF status can be acquired using the **BHT_GetNLedStatus** and **BHT_GetNLedStatusEx** functions

(2) Limited items

- ▪ LEDs cannot be controlled when a barcode device file is open. LEDs can be controlled, however, if LEDs are set not to illuminate when a barcode device file is open.
- ▪ The red color for the display LED and red color for the charge LED use an OR connection. As a result, if the red LED is ON while the BHT-400 is charging, it cannot be turned OFF even if attempting to do so from the application. At such times, remove the BHT-400 from the CU to turn OFF the red LED.
- ▪ If the function mentioned at (1) above is used to turn ON an LED from the application, the LED remains ON even after exiting the application used to turn ON the LED. Use the function mentioned at (1) to turn OFF the LED.

11.2.2. Charge LED

The charge LED cannot be turned ON or OFF from the application.

Chapter 12. Data Communication

12.1. Outline

In communication between the BHT and host computer, the following interfaces are available:
(1) IrDA interface
(2) Connector interface
(3) USB interface

12.2. Programming for Data Communication

(1) IrDA interface
The IrDA interface is assigned to port 4.

Communications parameter	Effective setting	Default
Transmission speed (bps)	115200, 57600, 38400, 19200, 9600	9600

Parameters other than the transmission speed are fixed (Parity = None, Character length = 8 bits, Stop bit length = 1 bit), since the physical layer of the IrDA interface complies with the IrDA-SIR 1.2.

(2) Connector interface
The Connector interface is assigned to port 1.

Communications parameter	Effective setting	Default
Transmission speed (bps)	115200,57600,38400,19200,9600, 4800,2400,1200,600,300	9600
Parity	None, even, or odd	None
Character length	7 or 8 bits	8
Stop bit length	1 or 2 bits	1

(3) USB interface
The USB interface is assigned to port 2.

12.3. ActiveSync

12.3.1. Establishing an ActiveSync Connection

An ActiveSync connection can be established automatically in addition to the manual connection method. The ActiveSync connection method is set to manual by default.
An ActiveSync automatic connection can be established using any of the following three procedures.

(1) By establishing an ActiveSync connection via the IrDA interface when the BHT is placed on the CU with the power ON.

Notes

- When establishing an ActiveSync connection via the IrDA interface, it is only possible to connect to the computer using a USB interface CU.
  It is not possible to connect using an RS-232C interface CU.

(2) By establishing an ActiveSync connection via the RS-232C interface when attaching an RS-232C cable to the BHT with the power ON.

(3) By establishing an ActiveSync connection via the USB interface when attaching a USB cable to the BHT with the power ON.

12.3.2. ActiveSync Auto Connection Setting Method

The ActiveSync auto connection function is set and read using the **BHT_SetSysSettingDW** (DWORD dwCtrlCode,DWORD dwSysParam) and **BHT_GetSysSettingDW** (DWORD dwCtrlCode,DWORD *pdwSysParam) functions.

Parameter	Type	R/W	Control Code	Parameter Value	Default	Validation Timing
ActiveSync auto connection	DW	R/W	BHT_ACTSYNC _AUTOCNCT	ACTSYNC _AUTOCNCT_DISABLE : Prohibited ACTSYNC _AUTOCNCT_INFRARED : Infra-red (IrDA) only permitted ACTSYNC _AUTOCNCT_USB : USB only permitted ACTSYNC _AUTOCNCT_RS232C : RS-232C only permitted	ACTSYNC _AUTOCNCT _DISABLE	After setting

Chapter 13. Wireless Communication

13.1. Outline

13.1.1. Spread Spectrum Communications Method

Through the integrated wireless card, the BHT uses the TCP/IP protocol subset over the spread spectrum communications device.
For details about programming for spread spectrum communication, refer to Section 13.2

13.1.2. Configuration of Spread Spectrum System

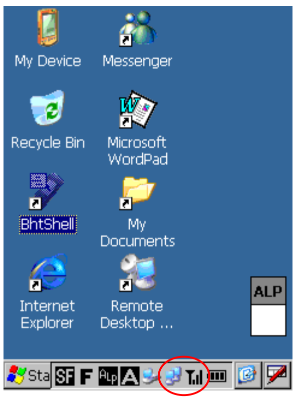
The BHT communicates with the host computer via an access point in wireless communication.

For details, refer to the "BHT-400B/400BW-CE User's Manual."

The table below shows the communications status transition as the state of the spread spectrum communications device built in the BHT-400.

Spread spectrum communications device	Communication
Open (power on)	Impossible
Checking synchronization with access point	Impossible
Synchronization complete	Possible
Roaming	Impossible if the BHT is not synchronized with an access point Possible if synchronization with an access point is kept
End of roaming	Possible
Close (power off)	Impossible

If always being opened, the spread spectrum communications device will consume much power. When the device is not in use, therefore, close it as soon as possible.
However, it will take several seconds to open the spread spectrum communications device and synchronize it with the access point for making communications ready. Frequent opening and closing of the device will require much time, resulting in slow response. Take into account the application purposes of user programs when programming.
When the spread spectrum communications device is synchronized with the access point, the BHT will display a synchronization icon on the LCD as shown below.



13.2. Programming for Wireless Communication

To connect to the wireless communications pathway, specify the following system settings in System Menu or in a user program:
- POWER
- ESSID (Extended Service Set ID)
- ENCRYPTION
- AUTHENTICATION
- EAP TYPE
- WEP KEY

For the procedure in System Menu, refer to the "BHT-400B/400BW-CE User's Manual."
If no system settings are made in a user program, those made in System Menu will apply.

The following procedure is used to perform system settings in the user program.

Step 1: Select the profile to be edited.

Call the following function to edit an existing profile.
BHT_RF_IoControl (RF_UPDATE_PROFILE, NULL, 0, NULL, 0, NULL);
Call the following function to edit or create a new profile.
BHT_RF_IoControl (RF_SET_PROFILE, …);
Please refer to section "13.2.1 Wireless Communication Parameters" for details of the setting method.
Use ESSID and Infrastructure mode to specify the profile.
If no profile corresponding to the specified ESSID and Infrastructure mode combination exists, a new profile will be created.

Step 2: Change parameter 1, parameter 2, ….., parameter N for the profile selected at Step 1.

Please refer to section "13.2.1 Wireless Communication Parameters".

Step 3: Update the set parameters to the driver.

BHT_RF_IoControl (RF_COMMIT_PROFILE, NULL, 0, NULL, 0, NULL);

Use the highest priority profile from among those created to attempt a connection.
If connection fails, attempt to connect automatically using the highest priority profiles sequentially.

The profile with the highest priority will be the one created last.
Up to a maximum of 16 profiles can be created.

13.2.1. Wireless Communication Parameters

Settable Parameters

The BHT can be used with the following security configurations by setting ZeroConfig.
- PEAP (802.1x)
- EAP-TLS (802.1x)
- PEAP (WPA)
- EAP-TLS (WPA)
- PSK (WPA)

Details of the parameters used with the above security configurations are outlined in the table below.

Parameter	Security					
	None	PEAP (802.1x)	EAP-TLS (802.1x)	PEAP (WPA)	EAP-TLS (WPA)	PSK (WPA)
Authentication	OPEN	OPEN	OPEN	WPA	WPA	WPA-PSK
Encryption	Disable WEP (static)	WEP (auto distribution)	WEP (auto distribution)	TKIP	TKIP	TKIP
802.1x	Disable	PEAP	EAP-TLS	PEAP	EAP-TLS	Disable
ESSID	●	●	●	●	●	●
Profile Priority	●	●	●	●	●	●
Pre Shared Key	-	-	-	-	-	●
WEP Key	●	-	-	-	-	-

(●: Setting valid, -: Setting invalid)

▪ POWER
Set the power mode for the wireless module built in the BHT. The following two power modes are available. The default is P_PWRSAVE_PSP.

Power mode	Power consuming state
P_PWRSAVE_CAM	Consumes much power (no power saving effect)
P_PWRSAVE_PSP	Consumes less power (much power saving effect). The BHT may take more time to establish the wireless link or send response messages.

[Ex.] Set the power mode to "Cosumes much power"
```
DWORD dwVal = P_PWRSAVE_CAM;
BHT_RF_SetParamInt (P_INT_POWERSAVE, &dwVal, sizeof(dwVal));
```

▪ ESSID
Specify an ID that identifies the wireless network as a character string. The ESSID of the BHT should be the same as the SSID of the access point. If the ESSID is not set correctly, no communication is possible.

[Ex.] Set the "BHT400" to the ESSID (The infrastructure mode is assumed to be an “Infrastructure.”)
```
ST_RF_PROFILE_KEY  stKey;
wcscpy(&stKey.szESSID[0], TEXT(“BHT400”));           // ESSID
stKey.dwInfraMode = INFRA_INFRASTRUCTURE;      // Infrastructure
BHT_RF_IoControl (RF_SET_PROFILE, (LPVOID)&stKey, sizeof(stKey), NULL, 0, NULL);
```

▪ ENCRYPTION
This is the encryption method setting. A selection can be made from Prohibited, WEP, and TKIP.

▪ AUTHENTICATION
This is the authentication method setting. A selection can be made from Open, Shared, WPA, and WPA-PSK.

▪ EAP TYPE
This is the EAP type setting. A selection can be made from Prohibited, PEAP, and TLS.

▪ WEP KEY
The encryption key (WEP KEY) can be set.

[Ex.] Setting to enable WEP. Set the WEP KEY to "01234567890123456789ABCDEF" (128 bit).

```
DWORD dwVal = P_AUTH_OPEN;
BHT_RF_SetParamInt (P_INT_AUTHENTICATE, &dwVal, sizeof(dwVal));
DWORD dwVal = P_ENCRYPT_WEP;
BHT_RF_SetParamInt (P_INT_ENCRYPTION, &dwVal, sizeof(dwVal));
DWORD dwVal = P_8021X_DISABLE;
BHT_RF_SetParamInt (P_INT_8021X, &dwVal, sizeof(dwVal));
BHT_RF_SetParamStr (P_STR_WEPKEY1,
                    TEXT("01234567890123456789ABCDEF"),26);
```

Parameter List

Parameter	Type	R/W	Parameter value	Default
Power mode	DW	R/W	P_PWRSAVE_CAM : High power consumption P_PWRSAVE_PSP : Low power consumption	P_PWRSAVE_PSP
Authentication method	DW	R/W	P_AUTH_OPEN : Open P_AUTH_SHARED : Shared P_AUTH_WPA : WPA P_AUTH_WPAPSK : WPA PSK	P_AUTH_OPEN
Encryption	DW	R/W	P_ENCRYPT_DISABLE : Prohibited P_ENCRYPT_WEP : WEP P_ENCRYPT_TKIP : TKIP	P_ENCRYPT_DISABLE
802.1x Encryption (EAP type)	DW	R/W	P_8021X_DISABLE : Prohibited P_8021X_PEAP : PEAP P_8021X_TLS : TLS	P_8021X_DISABLE
Profile priority	DW	R/W	1 (high) to 16 (low)	1
Index Key	DW	R/W	1 to 4	1
WEP Key 1	WCS	W	26-character hexadecimal notation character string (128 bit) 10-character hexadecimal notation character string (40 bit)	TEXT("")
Pre Shared Key	WCS	W	8 to 63-character ASCII character string 64-character hexadecimal notation character string	TEXT("")
Version	WCS	R	-	
MAC address	WCS	R	-	TEXT("00.00.00.00.00.00")

Note that if you use **BHT_RF_GetParamInt** function for getting a value, the value preset by the **BHT_RF_SetParamInt** function will be obtained.

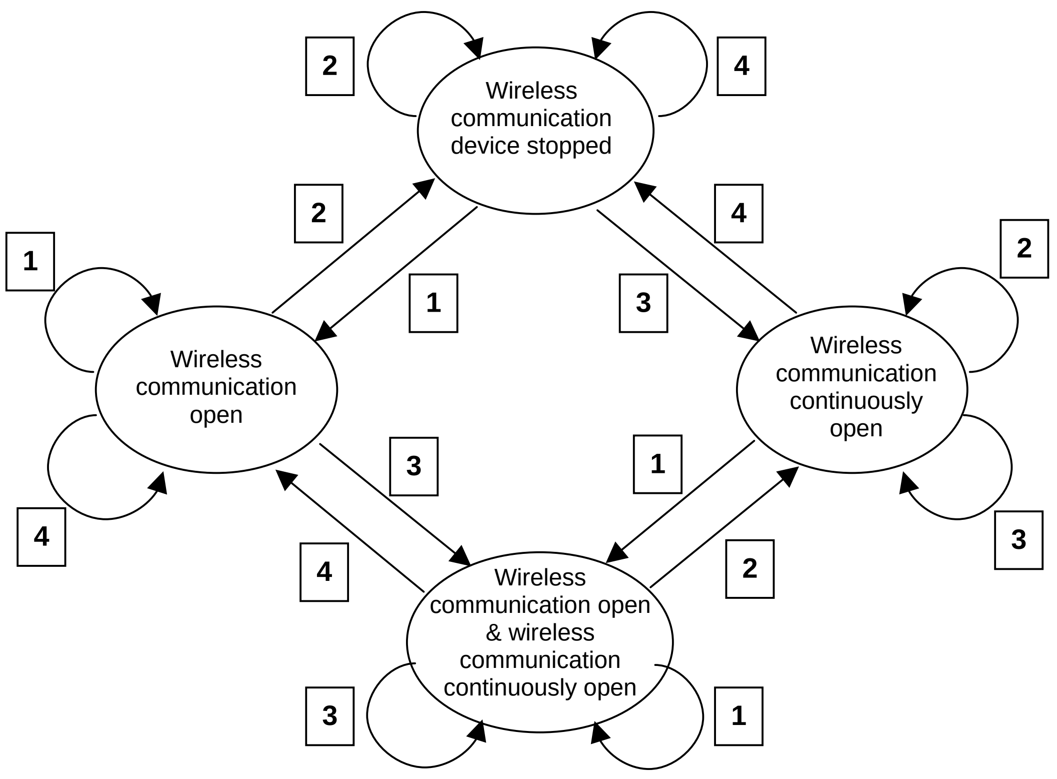
13.2.2. Opening and Closing the Wireless Communications Device

Use the **BHT_RF_Open** and **BHT_RF_OpenEx** functions to start up the wireless communication device and permit wireless communication.
Use the **BHT_RF_Close** and **BHT_RF_CloseEx** functions to stop the wireless communication device and prohibit wireless communication.

Use the **BHT_RF_OpenEx** (DWORD dwOpt) and **BHT_RF_CloseEx** (DWORD dwOpt) functions to perform wireless communication in the following communication formats.

Settable Value	Details
COMM_NORMAL	Wireless communication open
COMM_CONTINUOUS	Wireless communication continuously open

The following diagram illustrates the wireless communication device status transmission.



Wireless communication device stopped
Wireless communication open
Wireless communication continuously open
Wireless communication open & wireless communication continuously open
1
2
3
4

$\boxed{1}$ **BHT_RF_Open**() (*1)

$\boxed{2}$ **BHT_RF_Close**() (*2)

$\boxed{3}$ **BHT_RF_OpenEx**(COMM_CONTINUOUS)

$\boxed{4}$ **BHT_RF_CloseEx**(COMM_CONTINUOUS)

(*1) Includes BHT_RF_OpenEx(COMM_NORMAL)

(*1) Includes BHT_RF_OpenEx(COMM_NORMAL)

13.2.3. Checking Synchronization with the Access Point

When performing data communication with a wireless communication device, use the **BHT_RF_Synchronize** function to check whether synchronization with the access point has been obtained.

The following is a list of possible reasons why it may not be possible to obtain synchronization with the access point.

(1) The wireless communication device is currently open.

Several seconds are required to obtain synchronization with the access point after opening the wireless communication device.
Furthermore, when using DHCP, there are times when several tens of seconds are required to obtain the IP after connecting to the network.

(2) When the wireless device is moved from the current access point to the next access point during roaming

(3) When the wireless device is moved outside the radio-wave area covered by the access point.

(4) When the wireless device is moved to a location where an obstruction prevents wireless communication with the access point.

Chapter 14. Bar Code Reading

14.1. Outline

14.1.1. Enable Reading

The **BHT_EnableBar** function enables the bar code device to read bar codes. In this function, you may specify the following bar code types available in the BHT. The BHT can handle one of them or their combination.

Available Bar Code Type	Default Setting
Universal product codes EAN-13 (*1) (JAN-13 (*1)) EAN-8 (JAN-8) UPC-A (*1), UPC-E	No national flag specified.
Interleaved 2of5 (ITF)	No length of read data specified. No check digit.
Standard 2of5 (STF)	No length of read data specified. No check digit. Short format of the start/stop characters supported.
Codabar (NW-7)	No length of read data specified. No check digit. No start/stop character.
Code 39	No length of read data specified. No check digit.
Code 93	No length of read data specified.
Code 128 (EAN-128) (*2)	No length of read data specified.
Interleaved 2of5 (ITF)	No length of read data specified. No check digit.
RSS (*3)	Nothing specified.

(*1) Reading wide bars
EAN-13 and UPC-A bar codes may be wider than the readable area of the bar-code reading window. Such wider bars can be read by long-distance scanning. Pull the bar-code reading window away from the bar code so that the entire bar code comes into the illumination range.
(*2) Specifying Code 128 makes it possible to read not only Code 128 but also EAN-128.
(*3) Cannot be read using the BHT-400B for the Japanese market.

14.1.2. Specify Options in the **BHT_EnableBar** Function

You may also specify several options as listed below for each of the bar code types in the **BHT_EnableBar** function.

Barcode type	Options
Universal product code	Initial (country flag) add-on code
Interleaved 2of5 (ITF)	Length of read data Check digit
CODABAR (NW-7)	Length of read data Start/stop character Check digit
Code 39	Length of read data Check digit
Code 93	Length of read data
Code 128	Length of read data
Standard 2of5(STF)	Length of read data Start/stop character Check digit
MSI	1-digit check digit
RSS	Nothing specified.

14.1.3. Barcode Buffer

The barcode buffer stores the inputted barcode data.
The barcode buffer will be occupied by one operator entry job and can contain up to 99 characters.

You can check whether the barcode buffer stores code data, by using the **BHT_GetBarNum** function. To read barcode data stored in the barcode buffer, use the **BHT_ReadBar/BHT_ReadBarEx** function.

14.2. Programming

14.2.1. Code Mark

The **BHT_GetBarType** function allows you to check the code mark (denoting the code type) and the length of the inputted barcode data.

14.2.2. Multiple Code Reading

You may activate the multiple code reading feature which reads more than one code type while automatically identifying them. To do it, you should designate desired code types in the read code parameter of the **BHT_EnableBar** function.

14.2.3. Read Mode of the Trigger Switch

The trigger switch function is assigned to the magic keys M3 and M4 by default. You may assign the trigger switch function to other keys by using the **BHT_SysSettingDW** function.
You may select the read mode of the trigger switch by using the **BHT_EnableBar** function as listed below.

Read Mode	**BHT_EnableBar** Function
Auto-off Mode (Default)	**BHT_EnableBar** (TEXT ("F…
Momentary Switching Mode	**BHT_EnableBar** (TEXT ("M…
Alternate Switching Mode	**BHT_EnableBar** (TEXT ("A…
Continuous Reading Mode	**BHT_EnableBar** (TEXT ("C…

To check whether the trigger switch is pressed or not, use the **BHT_WaitEvent** function as shown below.

```
BHT_WaitEvent (1, BHT_EVT_MASK_TRGDOWN, 0, &dwSignaledEvent);
    if ( (dwSignaledEvent & BHT_EVT_MASK_TRGDOWN) != 0 ) {
        printf("Trigger switch pressed ");
    }
```

14.2.4. Generating a Check Digit of Barcode Data

Specifying a check digit in the **BHT_EnableBar** function makes the Interpreter automatically check bar codes. If necessary, you may use the **BHT_GetBarChkdgt** function for generating a check digit of barcode data.

14.2.5. Controlling the Indicator LED and Beeper/Vibrator as a Confirmation of Successful Reading

By using the **BHT_EnableBar** function, you can control:

- whether the indicator LED should light in blue or not (Default: Light in blue)
- whether the beeper should beep or not (Default: No beep)

when a bar code is read successfully. For detailed specifications, refer to the description for the **BHT_EnableBar** function.

It is also possible to operate the vibrator as a confirmation of successful reading instead, by using the **BHT_SetSysSettingDW** (BHT_BEEP_VIB_SELECT, VIB_SELECT) function.

(1) Controlling the indicator LED

If you have activated the indicator LED (blue) in the **BHT_EnableBar** function, the **BHT_SetNLedStatus** function cannot control the LED.

If you have deactivated the indicator LED (blue) in the **BHT_EnableBar** function, the **BHT_SetNLedStatus** function can control the LED even when the barcode device file is opened.

This way, you can control the indicator LED, enabling that:

- a user program can check the value of a scanned bar code and turn on the indicator LED in blue when the bar code has been read successfully.

  (For example, you can make the user program interpret barcode data valued from 0 to 100 as correct data.)

- a user program can turn on the indicator LED in red the moment the bar code has been read.

(2) Controlling the beeper and vibrator

If you have activated the beeper in the **BHT_EnableBar** function, the BHT will beep when it reads a bar code successfully.

You may select beeping only, vibrating only, or beeping & vibrating by setting on the system menu (BHTSHELL.exe) or by setting the output port in the **BHT_SetSysSettingDW**.

This feature is used to sound the beeper or operate the vibrator the moment the BHT reads a bar code successfully.

14.3. Barcode Reading Using the Virtual COM Port

14.3.1. Outline

Barcode reading using the virtual COM port is supported on the BHT-400 series (see the DENSOWAVE QBNet website for updated support information).
For greater convenience, this function is available for use in conjunction with kbifCE. For more information on kbifCE, see the kbifCE user's guide (available for download on the DENSOWAVE QBNet website).
Using this function it is possible to obtain reading data as if it were being received through a COM port. For applications, it is equivalent to a reader being connected to the communication port (COMx). Using COM, barcode reading data can be used by multiple applications.

14.3.2. Programming

Port number 5 is allocated to the virtual COM port used for barcode reading.
Barcode reading mode and the types of barcodes that are allowed to be read are designated by the kbifCE.
A comparison of the functions of Win32 API when using a general COM port and a virtual COM port for barcode use is as follows:

Win32 API	General COM	Virtual COM used for reading
CreateFile	Open COM port	←
CloseHandle	Close COM port	←
ReadFile	Read received data	Read data
GetCommMask	Obtain type of wait event	←
SetCommMask	Set type of wait event	← Treat completed reading event as receiving event.Non-reading events invalid.(*1)
GetCommTimeouts	Obtain timeout value	←
SetCommTimeouts	Set timeout value	← Non-receiver side timeouts invalid.(*1)
WaitCommEvent	Wait for event	← Non-receiving events invalid.

(*1) An error will not occur.

The following functions are not supported. If operation is attempted, no function will be executed.

List of functions not yet supported		
WriteFile	GetCommModemStatus	SetCommBreak
ClearCommBreak	GetCommProperties	SetCommState
ClearCommError	GetCommState	SetupComm
EscapeCommFunction	PurgeComm	TransmitComm

14.3.3. How to Use

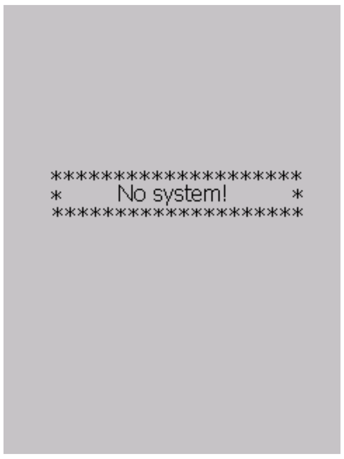
Start up kbifCE and set the destination for the virtual COM port (for further details see the kbifCE user's guide).

```
hVCom = CreateFile(TEXT(“COM5:”), GENERIC_READ, 0, NULL, OPEN_EXISTING, 0, NULL);
… …
… …
…
SetCommMask(hVCom, EV_RXCHAR);
while (TRUE) {
        bRtn = WaitCommEvent(hVCom, &dwSignaledEvent, NULL);
        if ( (TRUE == bRtn) && ((dwSignaledEvent & EV_RXCHAR) != 0) ) {
                ReadFile(hVCom, &buffer[0], 100, &dwRead);
        }
}
… …
CloseHandle(hVCom);
```

Chapter 15. System Messages

When the BHT is turned on or during program execution, the following system messages can be displayed.

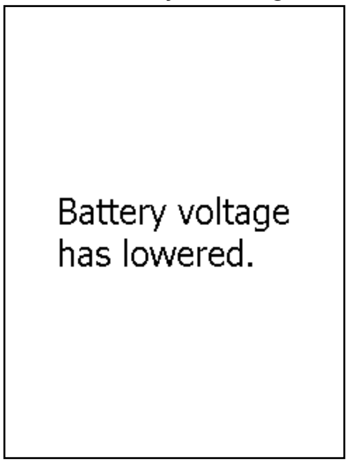
■ System program error



```
********************
*    No system!    *
********************
```

When System Program is not set up correctly, the BHT OS displays this error message, sounds the beeper five times (for 0.2 second per beep), and turns the power off.

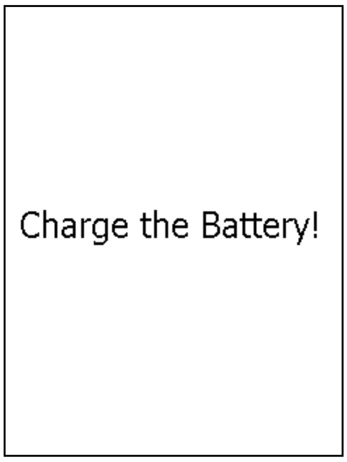
■ Low battery warning



Battery voltage
has lowered.

If the BHT switches from the suspend or critical OFF state to the power ON state, the OS measures the battery voltage level at the specified intervals. Only when you press a key or tap the touch panel first after the battery voltage level drops below 3.6 V, the OS displays this warning message for approx. 2 seconds and beeps three times (for 0.1 second per beep). After that, the BHT resumes previous regular operation.

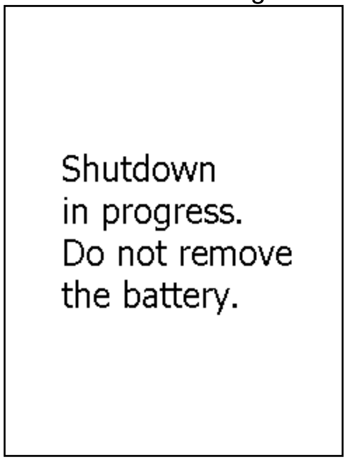
■ Shutdown due to low battery



Charge the Battery!

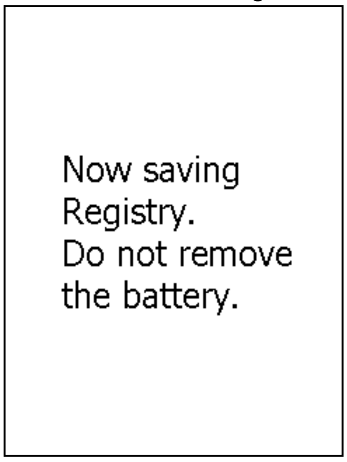
When the BHT is turned on, the BHT OS measures the battery voltage level at the specified intervals. If the battery voltage level drops below the specified level, the OS displays this error message for approx. 2 seconds, beeps five times (for 0.1 second per beep), switches to the critical OFF state.

■ Power-off message--without backing up the Registry


Shutdown
in progress.
Do not remove
the battery.

If the BHT is turned off by pressing the power key or by auto power-off feature, the BHT OS displays this error message and then switches to the suspend state.
comes to be on suspend.

■ Power-off message--with backing up the Registry


Now saving
Registry.
Do not remove
the battery.

If the BHT power is turned OFF by pressing power key while holding down the [SF] key, the registry is saved before the power turns OFF.
The message on the left displays while the registry is being saved.

Chapter 16. Updating OS

The OS can be updated (version update) using the following method when running Windows CE.

When using the BHT-400 RAM:

(1) Execute the **BHT_ShutdownSystem** (BHT_PWR_SYSMODIFY) function to secure an area for the OS file to be stored.
(2) The user should then copy the OS file to the "SysModify" directory.
(3) Execute the **BHT_SystemModify** function.
For the 1st argument, specify the absolute path to the folder (SysModify) in which the OS file was stored, and for the 2nd argument, specify whether to turn OFF the power or perform a cold boot after updating the OS.
(4) After the OS has been successfully updated, the BHT-400 power will either be turned OFF or will cold boot depending on the setting made for the 2nd argument.

When using the CF memory card:

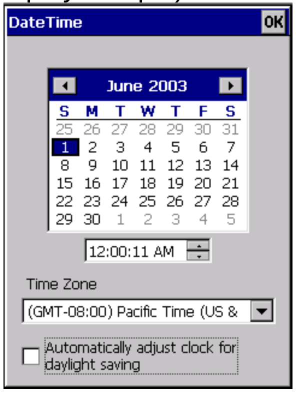
(1) The OS file is stored in the CF memory card, and the card then inserted into the BHT-400 CF slot.
(2) Execute the **BHT_SystemModify** function.
For the 1st argument, specify the absolute path to the CF card where the OS file was stored, and for the 2nd argument, specify whether to turn OFF the power or perform a cold boot after updating the OS.
(3) After the OS has been successfully updated, the BHT-400 power will either be turned OFF or will cold boot depending on the setting made for the 2nd argument.

Chapter 17. Starting the BHT

17.1. Setting up the BHT

(1) The touch panel adjustment screen will display when the BHT is booted up (when cold booted) if the touch panel adjustment value is not stored in the registry.
The touch panel adjustment screen is compliant with the Windows CE standard windows screen and input method.
(2) If the RTC is stopped when the BHT is booted up, a menu displays allowing the user to set the date and time.

(Display sample)



S	M	T	W	T	F	S
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

12:00:11 AM

Time Zone: (GMT-08:00) Pacific Time (US &

☐ Automatically adjust clock for daylight saving

After completion of setting of date, time, and time zone, tap the OK button.

17.2. Warm Boot / Cold Boot

(1) Warm boot / Cold boot conditions
The Warm Boot / Cold Boot conditions are as follows.

Boot Method	Conditions
Cold boot	- When the BHT-400 is booted up by pressing the Power key and Reset buttons simultaneously - When the BHT-400 is booted up after updating the OS - When the BHT-400 is booted up when the RAM is volatile - When cold boot is specified using the **BHT_ShutdpwnSystem** function
Warm boot	- When the Reset button is pressed, regardless of whether the power is ON or OFF - When warm boot is specified using the **BHT_ShutdpwnSystem** function

(2) Memory contents after Cold boot / Warm boot

	Warm Boot	Cold Boot
Data in flash folder	●	●
Data in other folders	●	-
Registry	●	- [Note]
Data currently being edited	-	-

●: Data prior to reset saved, -: Data lost

Notes
If the registry has been saved then the saved registry is used.

17.3. Specifying the Reboot Modes in Application Programs

The **BHT_ShutdownSystem** function turns off the BHT to boot it in any of the following modes. In the case of (2) through (4), the BHT automatically boots as specified.

(1) Suspend
(2) Warm boot
(3) Cold boot with Registry initialization
(4) Cold boot without Registry initialization
(5) Cold boot (securing a contiguous area in RAM; for updating OS)

Chapter 18. System Functions

The system functions are used to write or read the BHT system parameters.
They are classified into two groups (DWORD/character string) according to values to be handled.

Function	Used to:
BHT_SetSysSettingDW	Write system parameter values (DWORD).
BHT_GetSysSettingDW	Read system parameter values (DWORD).
BHT_SetSysSettingWCS	Write system parameter values (character string).
BHT_GetSysSettingWCS	Read system parameter values (character string).

18.1. If a System Parameter Value is DWORD

BHT_SetSysSettingDW

Description
Write system parameter values.

Syntax

```
DWORD BHT_SetSysSettingDW (
DWORD dwCtrlCode ,
DWORD dwSysParam )
```

Parameters

dwCtrlCode
[in] Control code

dwSysParam
[in] Parameter value

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Invalid parameter
ERROR_GEN_FAILURE	Not supported

BHT_GetSysSettingDW

Description
Read system parameter values.

Syntax

```
DWORD BHT_GetSysSettingDW (
DWORD dwCtrlCode ,
DWORD* pdwSysParam )
```

Parameters

dwCtrlCode
[in] Control code

pdwSysParam
[out] Address for storing the parameter value

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_GEN_FAILURE	Not supported

18.2. If a System Parameter Value is a Character String

BHT_SetSysSettingWCS

Description

Write system parameter values.

Syntax

```
DWORD BHT_SetSysSettingWCS (
DWORD dwCtrlCode ,
TCHAR* pwchSysParam ,
DWORD dwLen )
```

Parameters

dwCtrlCode
[in] Control code

pwchSysParam
[in] Heading address of the storage buffer for a string written

dwLen
[in] String length

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Invalid parameter
ERROR_GEN_FAILURE	Not supported

BHT_GetSysSettingWCS

Description

Read system parameter values.

Syntax

```
DWORD BHT_GetSysSettingWCS (
DWORD dwCtrlCode ,
TCHAR* pwchSysParam ,
DWORD dwLen ,
DWORD* pdwLenReturned )
```

Parameters

dwCtrlCode
[in] Control code

pwchSysParam
[out] Heading address of the storage buffer for a string read

dwLen
[in] String length

pdwLenReturned
[out] Length of the string read out

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_GEN_FAILURE	Not supported

18.3. System Parameter Values That Can be Set/Obtained

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
System information related						
System version (4 characters)	WCS	R	BHT_SYS_OS _VERSION	-	-	-
Total RAM size (bytes)(*1)	DW	R	BHT_SYS _RAMSIZE	-	-	-
Total ROM size (bytes) (*1)	DW	R	BHT_SYS _ROMSIZE	-	-	-
Model name (8 characters)	WCS	R	BHT_SYS _MACHINE_NAME	-	-	-
Product number (16 characters)	WCS	R	BHT_SYS _MACHINE_NUMBER	-	-	-
Serial number (6 characters)	WCS	R/W	BHT_SYS _SERIAL_NUMBER	6-digit number	Lower 6 characters in the code printed on the back of the BHT	Immediately after setting
Power management related						
Waiting time to switch to standby mode (in units of 100 ms)	DW	R/W	BHT_PM_STBYTIME	0: Disable 1 to 255	10 (1 sec)	Immediately after setting
Waiting time to auto power OFF when powered by battery (sec.)	DW	R/W	BHT_PM _BATTPOWEROFF	0: Disable 1 to 0xFFFFFFFF	180 (3 min)	Immediately after setting
Waiting time to auto power OFF when placed on CU (sec.)	DW	R/W	BHT_PM _EXTPOWEROFF	0: Disable 1 to 0xFFFFFFFF	0	Immediately after setting
CPU clock (*2)	DW	R/W	BHT_PM _CPU_CLOCK	CPU_CLK_NORMAL : Regular speed CPU_CLK_FAST : High speed	CPU_CLK_NORMAL	When warm-booting after setting
Auto power OFF permitted/prohibited for CF slot 0 currently in use	DW	R/W	BHT_PM_SUSPEND _SLOT0	SUSPEND_ENABLE : Suspend permitted SUSPEND_DISABLE : Suspend prohibited	SUSPEND_DISABLE	Immediately after setting
Auto power OFF permitted/prohibited for CF slot 1 currently in use	DW	R/W	BHT_PM_SUSPEND _SLOT1	SUSPEND_ENABLE : Suspend permitted SUSPEND_DISABLE : Suspend prohibited	SUSPEND_ENABLE	Immediately after setting
Beeper and vibrator related						
Rumble device	DW	R/W	BHT_BEEP_VIB _SELECT	BEEP_SELECT : Beeper VIB_SELECT : Vibrator (BEEP_SELECT | VIB_SELECT) : Beeper and vibrator	BEEP_SELECT	Immediately after setting
Beeper volume	DW	R/W	BHT_BEEP_VIB _VOLUME	0:OFF 1 (lowest) to 5 (highest)	5	Immediately after setting
Key click volume	DW	R/W	BHT_BEEP_VIB_KEY	0: OFF 1: Soft 2: Loud	2	Immediately after setting
Screen tap volume	DW	R/W	BHT_BEEP_VIB_TAP	0: OFF 1: Soft 2: Loud	2	Immediately after setting
Half-pressed key click volume(*3)	DW	R/W	BHT_BEEP_VIB_KEY	0: OFF 1: Soft 2: Loud	0	Immediately after setting
Trigger switch clicks(*4)	DW	R/W	BHT_BEEP_VIB _TRGKEY	CLICK_SOUND_OFF : Prohibit CLICK_SOUND_ON : Allow	CLICK_SOUND_OFF	Immediately after setting
Laser lighting key clicks(*5)	DW	R/W	BHT_BEEP_VIB _LASERKEY	CLICK_SOUND_OFF : Prohibit CLICK_SOUND_ON : Allow	CLICK_SOUND_OFF	Immediately after setting

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Backlight related						
Backlight ON-duration (sec.) (When battery- driven)	DW	R/W	BHT_BACKLIGHT _BATT_TIME	0 - 255 0: Backlight OFF 255: Backlight continuously ON	3	Immediately after setting
Backlight ON-duration (sec.) (When placed on the CU)	DW	R/W	BHT_BACKLIGHT _AC_TIME	0 - 255 0: Backlight OFF 255: Backlight continuously ON	60	Immediately after setting
Control key	DW	R/W	BHT_BACKLIGHT _KEY	Key number	0x10204 ([SF]+[M4])	Immediately after setting
Backlight brightness level	DW	R/W	BHT_BACKLIGHT _BRIGHTNESS	0: OFF 1: Dark – 3: Bright	3	Immediately after setting
Backlight power saving mode	DW	R/W	BHT_BACKLIGHT _POWERSAVE	0: OFF 1: Dim	1	Immediately after setting
Barcode reading related						
Re-read prevention enabled time (in units of 100 ms)	DW	R/W	BHT_BAR_CRTIME	0 to 255 (*6)	10	Immediately after setting
Black-and-white inverted label reading function	DW	R/W	BHT_BAR_INVERT	0: Prohibit 1: Allow (automatic)	0	Immediately after setting
Decode level	DW	R/W	BHT_BAR_DCD _LEVEL	1 to 9	4	When the bar code device is opened first after setting
Min. number of digits to be read for ITF	DW	R/W	BHT_BAR_MINDGT _ITF	2 to 20	4	When the bar code device is opened first after setting
Min. number of digits to be read for STF	DW	R/W	BHT_BAR_MINDGT _STF	1 to 20	3	When the bar code device is opened first after setting
Min. number of digits to be read for Codabar (CODABAR)	DW	R/W	BHT_BAR_MINDGT _NW7	3 to 20	4	When the bar code device is opened first after setting
Scanning range marker	DW	R/W	BHT_BAR_MARKER	MARKER_NORMAL : Normal mode MARKER_AHEAD : Always ON (*7) MARKER_DISABLE : Fixed to OFF	MARKER _NORMAL / MARKER _DISABLE (*8)	Immediately after setting

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Keyboard related						
Shift key mode	DW	R/W	BHT_KEY_SHIFT _MODE	KEY_NON_LOCK : Non-lock KEY_ONE_TIME : Onetime lock	KEY_NON_LOCK	Immediately after setting
Assignment to M1 key	DW	R/W	BHT_KEY _M1_MODE	MAGIC_FUNC_NONE : Ignore the depressed key MAGIC_FUNC_ENTER : Treat as ENT key MAGIC_FUNC_TRG : Treat as trigger switch MAGIC_FUNC_SHIFT : Treat as SF key MAGIC_FUNC_ALT : Treat as ALT key MAGIC_FUNC_CTRL : Treat as CTRL key MAGIC_FUNC_BLT : Treat as bacjlight function on/off key MAGIC_FUNC_TAB : Treat as TAB key MAGIC_FUNC_LASER : Treat as laser lighting key MAGIC_FUNC_CLEAR : Treat as CLEAR key	MAGIC_FUNC_TAB	Immediately after setting
Assignment to M2 key	DW	R/W	BHT_KEY _M2_MODE		MAGIC_FUNC_NONE	Immediately after setting
Assignment to M3H key (M3 half-pressed)	DW	R/W	BHT_KEY _M3H_MODE		MAGIC_FUNC_LASER / MAGIC_FUNC_TRG (*9)	Immediately after setting
Assignment to M3 key	DW	R/W	BHT_KEY _M3_MODE		MAGIC_FUNC_TRG	Immediately after setting
Assignment to M4H key (M4 half-pressed)	DW	R/W	BHT_KEY _M4H_MODE		MAGIC_FUNC_LASER / MAGIC_FUNC_TRG (*9)	Immediately after setting
Assignment to M4 key	DW	R/W	BHT_KEY _M4_MODE		MAGIC_FUNC_TRG	Immediately after setting
Assignment to M5H key (M5 half-pressed)	DW	R/W	BHT_KEY _M5H_MODE		MAGIC_FUNC_LASER / MAGIC_FUNC_TRG (*9)	Immediately after setting
Assignment to M5 key	DW	R/W	BHT_KEY _M5_MODE		MAGIC_FUNC_TRG	Immediately after setting
Scan key mode	DW	R/W	BHT_KEY _SCAN_MODE		MAGIC_FUNC_TRG	Immediately after setting
Key entry mode	DW	R/W	BHT_KEY _INPUT_METHOD	INPUT_METHOD _NUMERIC : Numeric entry mode INPUT_METHOD _ALPHABET : Alphabet entry mode	INPUT_METHOD _NUMERIC	Immediately after setting
Enable/disable alphabet entry switching key	DW	R/W	BHT_DISABLE _KEYMODE _CHANGE_KEY	ENABLE_KEY _TOCHANGE _ALPHABET : Enable alphabet entry DISABLE_KEY _TOCHANGE _ALPHABET : Disable alphabet entry	ENABLE_KEY _TOCHANGE _ALPHABET	Immediately after setting
Function mode	DW	R/W	BHT_KEY _FUNCTION	KEY_FUNCTION_ON : Function mode KEY_FUNCTION_OFF : Non-function mode	KEY_FUNCTION_ OFF	Immediately after setting
Effective held-down time of power key for suspending (in units of 100 ms)	DW	R/W	BHT_PWRDOWN _KEY_WAIT_TIME	1 - 255	5	Immediately after setting
Keypad type	DW	R	BHT_KEYBOARD _TYPE	KEYBOARD_TYPE1 : 31-key pad KEYBOARD_TYPE2 / KEYBOARD_TYPE2P : 50-key pad (Phone-type key layout) KEYBOARD_TYPE2C : 50-key pad (Calculator-type key layout)	-	-

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Status indicator related						
Battery voltage level icon	DW	R/W	BHT_ICON _BATTERY	0: Hide 1: Display	1	Immediately after setting
Software keyboard icon	DW	R/W	BHT_ICON_SIP	0: Hide 1: Display	0	Immediately after setting
Keypad shift icon	DW	R/W	BHT_ICON _SHIFTKEY	0: Hide 1: Display	1	The icon appears when the keypad is shifted first after this parameter is set to "1." (If the keypad has been shifted, the icon appears immediately.) It disappears when the shift is released first after this parameter is set to "0."
Alphabet input icon	DW	R/W	BHT_ICON _IN_ALPHA	0: Hide 1: Display	1	The icon appears when the alphabet input function is activated first after this parameter is set to "1." It disappears when the alphabet input function is deactivated first after this parameter is set to "0."
Wireless communication state icon	DW	R/W	BHT_ICON _RADIO_INTEN SE	0: Hide 1: Display	1	The icon appears when the wireless device is opened first after this parameter is set to "1." (If the wireless device has been opened, the icon appears immediately.) It disappears immediately after this parameter is set to "0."
Standby state icon	DW	R/W	BHT_ICON _STANDBY	0: Hide 1: Display	0	The icon appears when the CPU comes to be on standby first after this parameter is set to "1." It disappears immediately after this parameter is set to "0."
Function mode state icon	DW	R/W	BHT_ICON_FU NC	0: Hide 1: Display	0	The icon appears when the function mode is activated first after this parameter is set to "1." It disappears when the function mode is deactivated first after this parameter is set to "0."
Bluetooth power status	DW	R/W	BHT_ICON _BLUETOOTH	0: Hide 1: Display	0	Immediately after setting
Communication related						
ActiveSync automatic connection	DW	R/W	BHT_ACTSYNC _AUTOCNCT	ACTSYNC_AUTOCNCT _DISABLE : Prohibited ACTSYNC_AUTOCNCT _INFRARED : Only IrDA allowed (*10) ACTSYNC_AUTOCNCT _USB : Only USB allowed ACTSYNC_AUTOCNCT _RS232C : Only RS232C allowed	ACTSYNC _AUTOCNCT _DISABLE	After setting, when the USB cable or RS232C cable is first inserted, or when the CU421 is installed.

Parameter name	Type	R/W	Control code	Parameter value	Default	Validating timing
Touch screen related						
Touch screen disabling	DW	R/W	BHT_TOUCH _DEVICE	TOUCH_ENABLE : Enable TOUCH_DISABLE : Disable	TOUCH_ENABLE	Immediately after setting
Bluetooth related						
Bluetooth device initial power status	DW	R/W	BHT_BT_INITIAL_ POWER_STATUS	BHT_BT_POWER_OFF : Power OFF BHT_BT_POWER_ON : Power ON	BHT_BT _POWER_OFF	Immediately after setting
Others						
Grip connection	DW	R	BHT_HANDLE _STATUS	HANDLE_STATUS _LOADED : Grip connected HANDLE_STATUS _NO_HANDLE : No grip connected	-	-

(*1) The RAM or ROM size obtained indicates the capacity of the memory mounted on the BHT. To obtain t he size of the memory area allowed for the user to use, use GetDiskFreeSpaceEx.
(*2) If the CPU clock is set to high speed, the processing speed becomes higher but the power consumption Increases.
(*3) This parameter controls the click volume of the M3, M4, and M5 keys half-pressed.
(*4) This parameter controls the on/off of the click sound of the magic key which the trigger switch is assigned to. If it is set to ON, pressing the magic key clicks at the volume specified by the "Key clock volume"/"Half-pressed key click volume."
(*5) The parameter controls the on/off of the click sound of the magic key which the laser lighting key is assigned to. If it is set to ON, pressing the magic key clicks at the volume specified by the "Key clock volume"/"Half-pressed key click volume."
(*6) If this parameter is set to "0," the BHT no longer reads the same bar code in succession.
(*7) Marker ahead mode is supported only on those models intended for the domestic Japanese market.
(*8) The default value for the model without marker is "MARKER_DISABLE".
(*9) The default value for the model without marker is "MAGIC_FUNC_TRG".
(*10) The CU-421 is necessary to enable the ActiveSync automatic connection function used by the IrDA.

Chapter 19. Device Control Functions

The device control functions listed below control the devices (barcode reading device, backlight, battery, indicator LED, etc.) dedicated to the BHT.

Function	Used to:
BHT_EnableBar	Open the bar code device file to enable bar code reading. This function specifies the read mode and readable bar code types.
BHT_DisableBar	Close the barcode device file to disable bar code reading.
BHT_ReadBar	Read out data read from the barcode buffer.
BHT_ReadBarEx	Read out data from the barcode buffer and encodes it into the specified data format.
BHT_GetBarType	Read the bar code type and the number of digits of a bar code read most recently.
BHT_GetBarNum	Read the number of digits of the bar code remaining in the barcode buffer.
BHT_GetBarInfo	Read the information on the code read most recently.
BHT_GetBarChkDgt	Calculate a check digit (CD) of the barcode data according to the calculation method specified by dwCDType.
BHT_BAR_SetDecodeOptions	Sets the editing function setting value for the decoded result.
BHT_BAR_GetDecodeOptions	Acquires the editing function setting value for the decoded result.
BHT_SetBltStatus	Control the backlight.
BHT_GetBltStatus	Read the backlight status.
BHT_GetPowerStatus	Read information about the battery loaded in the BHT body.
BHT_GetPowerStatus2nd	Read information about the battery loaded in the grip.
BHT_GetNLedStatus	Read the status of the indicator LED.
BHT_SetNLedStatus	Control the indicator LED.
BHT_GetNLedStatusEx	Read the status of the indicator LED and synchronization LED.
BHT_SetNLedOn	Turn on the indicator LED and/or synchronization LED.
BHT_SetNLedOff	Turn off the indicator LED and/or synchronization LED.
BHT_StartBeep	Drive the beeper/vibrator.
BHT_StartBeeperOnly	Drive the beeper.
BHT_StartVibrationOnly	Drive the vibrator.

Function	Used to:
BHT_RF_Open	Open the wireless LAN device and enable wireless communication.
BHT_RF_OpenEx	Set the communication format, open the wireless LAN device and enable wireless communication.
BHT_RF_Close	Close the wireless LAN device and disable wireless communication.
BHT_RF_CloseEx	Close the wireless LAN device for the set format and disable wireless communication.
BHT_RF_Synchronize	Get the association status.
BHT_RF_GetParamInt	Read integer from the wireless communications parameter.
BHT_RF_SetParamInt	Write integer to the wireless communications parameter.
BHT_RF_GetParamStr	Read string from the wireless communications parameter.
BHT_RF_SetParamStr	Write string to the wireless communications parameter.
BHT_RF_GetInfoInt	Read integer from the communications parameter.
BHT_RF_GetInfoStr	Read string to the communications parameter.
BHT_RF_IoControl	Perform operation for the profile and certificate etc.
BHT_RF_GetSiteSurvey	Get quality of the communications link.
BHT_SystemModify	Update the BHT OS.
BHT_WaitEvent	Make the system wait until the specified event or timeout occurs.
BHT_WaitStandbyEvent	Make the system wait until the specified event occurs.
BHT_ShutdownSystem	Turn off the BHT and boot it according to the specified mode.
BHT_RegStore	Turn off the BHT and boot it according to the specified mode.

19.1. Barcode API

BHT_EnableBar

Description

Open the bar code device file to enable bar code reading.
This function specifies the read mode and readable bar code types. Up to eight bar code types can be specified.

Syntax

DWORD BHT_EnableBar (
TCHAR* *pwchRdMode* **,**
TCHAR* *pwchCdParam* **)**

Parameters

pwchRdMode
[in] Heading address of the storage buffer for a character string specifying the read mode, beeper/vibrator on/off, and LED on/off

pwchCdParam
[in] Heading address of the storage buffer for a character string specifying bar code types to be read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_TOO_MANY_OPEN_FILES	Barcode device file already opened.
ERROR_INVALID_PARAMETER	Parameter error. More than 24 bar code types are specified.

Comment:

Up to 24 bar code types can be specified.

■ readmode

The BHT supports four read modes--the momentary switching mode, the auto-off mode, the alternate switching mode, and the continuous reading mode, which can be selected by specifying M, F, A, and C to readmode, respectively.

□ Momentary switching mode (M)

Only when you hold down the trigger switch, the illumination LED lights and the BHT can read a bar code.
Until the entered barcode data is read out from the barcode buffer, pressing the trigger switch cannot turn on the illumination LED so that the BHT cannot read the next bar code.

[Ex]
BHT_EnableBar (TEXT ("M"), TEXT ("A, I:4-99, M:1-99, N:3-99, L:1-99, K:1-99, H:1-99"))

□ Auto-off mode (F)

If you press the trigger switch, the illumination LED comes on. When you release the switch or when the BHT completes bar code reading, then the illumination LED will go off. Holding down the trigger switch lights the illumination LED for a maximum of 5 seconds.
While the illumination LED is on, the BHT can read a bar code until a bar code is read successfully or the bar code devices file becomes closed.
If the illumination LED goes off after 5 seconds from when you press the trigger switch, it is necessary to press the trigger switch again for reading a bar code.
Once a bar code is read successfully, pressing the trigger switch cannot turn on the illumination LED and the BHT cannot read the next bar code as long as the entered barcode data is not read out from the barcode buffer.

[Ex]
BHT_EnableBar (TEXT ("F"), TEXT ("A, I:4-99, M:1-99, N:3-99, L:1-99, K:1-99, H:1-99"))

□ Alternate switching mode (A)

If you press the trigger switch, the illumination LED comes on. Even if you release the switch, the illumination LED remains on until the bar code device file becomes closed or you press that switch again. While the illumination LED is on, the BHT can read a bar code.
Pressing the trigger switch toggles the illumination LED on and off.
Once a bar code is read successfully, pressing the trigger switch turns on the illumination LED but the BHT cannot read the next bar code as long as the entered barcode data is not read out from the barcode buffer.

[Ex]
BHT_EnableBar (TEXT("A"), TEXT("A,I:4-99,M:1-99,N:3-99,L:1-99,K:1-99,H:1-99"))

□ Continuous reading mode (C)

If this mode is specified, the BHT turns on the illumination LED and keeps it on until the bar code device file becomes closed, irrespective of the trigger switch.
While the illumination LED is on, the BHT can read a bar code.
Once a bar code is read successfully, the BHT cannot read the next bar code as long as the entered barcode data is not read out from the barcode buffer.

[Ex]
BHT_EnableBar (TEXT("C"), TEXT("A,I:4-99,M:1-99,N:3-99,L:1-99,K:1-99,H:1-99"))

If readmode is omitted, the BHT defaults to the auto-off mode.
In the momentary switching mode, alternate switching mode, or continuous reading mode, after you read a low-quality bar code which needs more than one second to be read, keeping applying the barcode reading window to that bar code may re-read the same bar code in succession at intervals of one second or more.

■ beepercontrol and LEDcontrol

This function can control the beeper and the indicator LED to activate or deactivate each of them when a bar code is read successfully. This function may also control the vibrator with beepercontrol.

- You should describe parameters of readmode, beepercontrol, and LEDcontrol without any space inbetween.
- You should describe readmode, beepercontrol, and LEDcontrol in this sequence.
- Specifying B to beepercontrol allows you to select beeping only, vibrating only, or beeping & vibrating according to the setting made on the BEEP/VIBRATOR menu in System Menu or the setting made with the system function.
- Specifying L to LEDcontrol will not turn on the indicator LED.

[Ex] To sound the beeper (or operate the vibrator) when a bar code is read successfully:
BHT_EnableBar (TEXT("FB"), TEXT("A,I:4-99,M:1-99,N:3-99, L:1-99,K:1-99,H:1-99"))

[Ex] To deactivate the indicator LED when a bar code is read successfully:
BHT_EnableBar (TEXT ("FL"), TEXT ("A, I:4-99, M:1-99, N:3-99, L:1-99, K:1-99, H:1-99"))

■ readcode

The BHT supports the universal product codes, Interleaved 2of5 (ITF), Standard 2of5 (STF), Codabar (NW-7), Code 39, Code 93, and Code 128, MSI, Plessey, and Anker. The BHT can read also EAN-128 if Code 128 is specified.

□ Universal product codes (A)

Syntax

A [;[code][1st character [2nd character]][supplemental]]

where code is A, B, or C specifying the following:

code	Bar code type
A	EAN-13 (JAN-13), UPC-A
B	EAN-8 (JAN-8)
C	UPC-E

If code is omitted, the default is all of the universal product codes.
1stchara and 2ndchara are flag characters representing a country code and should be numerals from 0 to 9. If a question mark (?) is specified to 1stchara or 2ndchara, it acts as a wild card.

“supplemental” refers to the reading of an add-on code. Specifying an S for add-on enables the BHT to read also bar codes with an add-on code.

[Ex] To enable the BHT to scan EAN-13 with 1stchara "4," 2ndchara "9" and add-on code
BHT_EnableBar(TEXT("FL"), TEXT("A:49S"))

[Ex] To enable the BHT to scan EAN-13 and EAN-8 only
BHT_EnableBar(TEXT("FL"), TEXT("A:A,A:B"))

□ Interleaved 2 of 5 (ITF) (I)

Syntax

I[:[mini.no.digits[-max.no.digits]][CD]]

where
mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for bar codes to be read by the BHT, respectively.They should be a numeral from 2 to 99 and satisfy the following conditions:

mini.no.digits ≤ max.no.digits

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is from the minimum number of digits specified in the system menu (BHTSHELL.exe) up to 99 digits.
If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

CD is a check digit. Specifying a C to CD makes the Interpreter check bar codes with MOD-10. The check digit is included in the number of digits.

[Ex] To enable the BHT to scan ITF with min.no.digits 6, max.no.digits 10, and MOD-10
BHT_EnableBar(TEXT("FL"), TEXT("I:6-10C"))

[Ex] To enable the BHT to scan ITF with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40
BHT_EnableBar(TEXT("FL"),TEXT("I:6-10,I:20-40"))

□ CODABAR (NW-7) (N)

Syntax

N[:[mini.no.digits[-max.no.digits]][startstop][CD]]

Where
mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for bar codes to be read by the BHT, respectively.They should be a numeral from 3 to 99 and satisfy the following condition:

mini.no.digits ≤ max.no.digits

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is from the minimum number of digits specified in the system menu (BHTSHELL.exe) up to 99 digits.
If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

start and stop are the start and stop characters, respectively. Each of them should be an A, B, C, or D. If a question mark (?) is specified, it acts as a wild card. The start and stop characters are included in the number of digits. The A through D will be stored in the barcode buffer as a through d.

CD is a check digit. Specifying a C to CD makes the Interpreter check bar codes with MOD-16. The check digit is included in the number of digits.

[Ex] To enable the BHT to scan NW-7 with min.no.digits 8, start character A and stop character A, and MOD-16
BHT_EnableBar(TEXT("FL"), TEXT("N:8AAC"))

[Ex] To enable the BHT to scan NW-7 with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40
BHT_EnableBar(TEXT("FL"),TEXT("N:6-10,N:20-40"))

□ CODE-39 (M)

Syntax

M[:[mini.no.digits[-max.no.digits]][CD]]

Where
mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for bar codes to be read by the BHT, respectively.They should be a numeral from 1 to 99, excluding start/stop characters. They should satisfy the following condition:

mini.no.digits ≤ max.no.digits

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

CD is a check digit. Specifying a C to CD makes the Interpreter check bar codes with MOD-43. The check digit is included in the number of digits.

[Ex] To enable the BHT to scan Code 39 with min.no.digits 8, max.no.digits 12, and MOD-43
BHT_EnableBar(TEXT("FL"), TEXT("M:8-12C"))

[Ex] To enable the BHT to scan Code 39 with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40
BHT_EnableBar(TEXT("FL"),TEXT("M:6-10,M:20-40"))

□ CODE-93 (L)

Syntax

L[:[mini.no.digits[-max.no.digits]]]

Where
mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for bar codes to be read by the BHT, respectively.They should be a numeral from 1 to 99, excluding start/stop characters and check digits. They should satisfy the following condition:

mini.no.digits ≤ max.no.digits

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

[Ex] To enable the BHT to scan Code 93 with min.no.digits 6 and max.no.digits 12
BHT_EnableBar(TEXT("FL"), TEXT("L:6-12"))

[Ex] To enable the BHT to scan Code 93 with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40
BHT_EnableBar(TEXT("FL"),TEXT("L:6-10,L:20-40"))

NOTE: Neither start/stop characters nor check digits will be transferred to the barcode buffer.

□ CODE-128 (K)

Syntax

K[:[mini.no.digits[-max.no.digits]]]

Where
mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for bar codes to be read by the BHT, respectively.They should be a numeral from 1 to 99, excluding start/stop characters and check digit. They should satisfy the following condition:

mini.no.digits ≤ max.no.digits

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

[Ex] To enable the BHT to scan Code 128 with min.no.digits 6 and max.no.digits 12
BHT_EnableBar(TEXT("FL"), TEXT("K:6-12"))

[Ex] To enable the BHT to scan Code 128 with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40
BHT_EnableBar(TEXT("FL"),TEXT("K:6-10,K:20-40"))

NOTE: Neither start/stop characters nor check digits will be transferred to the barcode buffer.

Handling special characters

If the BHT reads any bar code consisting of special characters only (such as FNC, CODEA, CODEB, CODEC and SHIFT characters), it will not transfer the data to the barcode buffer. The beeper sounds only if it is enabled.

Details about FNC characters

(1) FNC1

The BHT will not transfer an FNC1 character placed at the first or second character position immediately following the start character, to the barcode buffer. FNC1 characters in any other positions will be converted to GS characters (1Dh) and then transferred to the barcode buffer like normal data.

If an FNC1 immediately follows the start character, the bar code will be recognized as EAN-128 and marked with W instead of K.

(2) FNC2

If the BHT reads a bar code containing an FNC2 character(s), it will not buffer such data but transfer it excluding the FNC2 character(s).

(3) FNC3

If the BHT reads a bar code containing an FNC3 character(s), it will regard the data as invalid and transfer no data to the barcode buffer, while it may drive the indicator LED and beeper (vibrator) if activated this **BHT_EnableBar** function.

(4) FNC4

An FNC4 converts data encoded by the code set A or B into a set of extended ASCII-encoded data (128 added to each official ASCII code value).
1 A single FN4 character converts only the subsequent data character into the extended ASCII-encoded data.

A pair of FNC4 characters placed in successive positions converts all of the subsequent data characters preceding the next pair of FNC4 characters or the stop character, into the extended ASCII-encoded data. If a single FNC4 character is inserted in those data characters, however, it does not convert the subsequent data character only.

An FNC4 character does not convert any of GS characters converted by an FNC1 character, into the extended ASCII-encoded data.

□ Standard 2 of 5 (STF) (H)

Syntax

H[:[mini.no.digits[-max.no.digits]][CD][startstop]]

Where
mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for bar codes to be read by the BHT, respectively.They should be a numeral from 1 to 99, excluding start/stop characters. They should satisfy the following condition:

mini.no.digits ≤ max.no.digits

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is from the minimum number of digits specified in the system menu (BHTSHELL.exe) up to 99 digits.
If only max.no.digits is omitted, only the number of digits specified by mini.no.digits can be read.

CD is a check digit. Specifying a C to CD makes the Interpreter check bar codes with MOD-10. The check digit is included in the number of digits.

startstop specifies the normal or short format of the start/stop characters.
Specify N for the normal format; specify S for the short format. If startstop is omitted, start/stop characters can be read in either format.

[Ex] To enable the BHT to scan STF with min.no.digits 6 and max.no.digits 12
BHT_EnableBar(TEXT("FL"), TEXT("H:6-12"))

[Ex] To enable the BHT to scan STF with min.no.digits 6 and max.no.digits 10 or with min.no.digits 20 and max.no.digits 40
BHT_EnableBar(TEXT("FL"),TEXT("H:6-10,H:20-40"))

□ MSI (P)

Syntax

P[:[mini.no.digits[-max.no.digits]][CD]]

Where
mini.no.digits and max.no.digits are the minimum and maximum numbers of digits for bar codes to be read by the BHT, respectively.They should be a numeral from 1 to 99, excluding start/stop characters. They should satisfy the following condition:

mini.no.digits ≤ max.no.digits

If both of mini.no.digits and max.no.digits are omitted, then the default reading range is 1 to 99 digits. If only max.no.digits is omitted, the BHT can only read the number of digits specified by mini.no.digits.

CD is a check digit. Specifying a C1 or C2 to CD makes the Interpreter check bar codes with a single-digit or two-digit CD, respectively. If no CD is specified, the Interpreter checks bar codes with a single-digit CD. The check digit is included in the number of digits.

[Ex] To enable the BHT to scan MSI with min.no.digits 6, max.no.digits 12, and a single CD check
BHT_EnableBar(TEXT("FL"), TEXT("P:6-12C1"))

[Ex] To enable the BHT to scan MSI with min.no.digits 6, max.no.digits 10 and a single CD check or with min.no.digits 20, max.no.digits 40 and a two-digit CD check
BHT_EnableBar(TEXT("FL"),TEXT("P:6-10,P:20-40C2"))

BHT_DisableBar

Description:
Close the barcode device file to disable bar code reading.

Syntax:

```
DWORD BHT_DisableBar (void)
```

Parameters
None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_HANDLE	Barcode device file not opened

BHT_ReadBar

Description

Read out data read from the barcode buffer.
If the string length longer than that of the read barcode is specified to dwBarLen, the remaining area following the read barcode will be filled with NULL codes.
If barcode reading is not enabled, an error (ERROR_INVALID_HANDLE) will result.

Syntax:

```
DWORD BHT_ReadBar (
TCHAR* pwchBuffer ,
DWORD dwBarLen ,
DWORD* pdwActualBarLen )
```

Parameters

pwchBuffer
[out] Heading address of the storage buffer storing the read data

dwBarLen
[in] Maximum length of data to be read

pdwActualBarLen
[out] Length of data read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_HANDLE	Barcode device file not opened.
ERROR_INVALID_PARAMETER	No storage address specified.

BHT_ReadBarEx

Description

Read out data from the barcode buffer and encodes it into the specified data format.
If the length of the read data is shorter than the specified maximum data length (dwBarLen), the excess part will be filled with 0s.
If barcode reading is disabled, an error (ERROR_INVALID_HANDLE) will be caused.

Syntax:

```
DWORD BHT_ReadBarEx (
DWORD dwDataType,
LPVOID lpBuffer,
DWORD dwBarLen,
DWORD* pdwActualBarLen )
```

Parameters

dwDataType
[in] Encoding format
READ_CODE_BINARY : binary data (no encoding)
READ_CODE_UNICODE : unicode data

lpBuffer
[in] Starting address of the read data in the storage buffer

dwBarLen
[in] Maximum read data length (maximum length of data to be read out)

pdwActualBarLen
[out] Length of data read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_HANDLE	Barcode device file not opened.
ERROR_INVALID_PARAMETER	The specified encoding is wrong. No storage address specified.

BHT_GetBarType

Description

Read the bar code type and the number of digits of a bar code read most recently.
If no bar code has been read after the BHT was turned on, the function gets "0."

Syntax

```
DWORD BHT_GetBarType (
DWORD* pdwBarMark ,
DWORD* pdwBarlen )
```

Parameters

pdwBarMark
[out] Address for storing the bar code type

pdwBarlen
[out] Address for storing the bar code length

The pdwBarMark contains one of the following letters representing code types:

Bar code type	*pdwBarMark*
(No code read)	0
EAN-13 (JAN-13), UPC-A	'A'
EAN-8 (JAN-8)	'B'
UPC-E	'C'
ITF	'I'
STF	'H'
CODABAR (NW-7)	'N'
CODE-39	'M'
CODE-93	'L'
CODE-128	'K'
EAN-128	'W'
MSI	'P'

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

BHT_ GetBarInfo

Description

Read the information on the code read most recently, including the code type and the number of digits in the code.
If no barcode has been read after the BHT was turned on, the function gets "0" for both the code type and the number of digits.

Syntax

```
DWORD BHT_GetBarInfo (
ST_CODE_INFO* pstInfo ,
DWORD* pdwCodeNum )
```

Parameters

pstInfo
[out] Destination address into which the code information is to be stored

pdwCodeNum
[in] No. of codes to be obtained

[out] Destination address into which the number of codes is to be stored. This is set to “1” when a code other than a multiple-row code or an EAN･UCC composite code has been read.

Shown below is the format of the structure containing code information. For the relationship between dwType and code type, refer to BHT_GetBarType.

```
struct ST_CODE_INFO {
    DWORD dwType;  // code type
    DWORD dwLen;  // no. of digits
) ;
```

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

If you specify NULL in pstCodeInfo, the number of elements of ST_CODE_INFO necessary to store the read code will be stored into pdwCodeNum.
An error occurs if a value greater than **MAX_NUM_CODE_1D_SCANNER** is specified for pdwCodeNum.

BHT_GetBarNum

Description

Read the number of digits of the bar code remaining in the barcode buffer.
If barcode reading is not enabled, an error (ERROR_INVALID_HANDLE) will result.

Syntax

```
DWORD BHT_GetBarNum (
DWORD* pdwCodeNum )
```

Parameters

pdwCodeNum
[out] Address for storing the bar code length

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_HANDLE	Barcode device file not opened
ERROR_INVALID_PARAMETER	Storage address not specified

BHT_GetBarChkDgt

Description

Calculate a check digit (CD) of the barcode data according to the calculation method specified by dwCDType.

Syntax

DWORD BHT_GetBarChkDgt (
TCHAR* *pwchBarbuf* **,**
DWORD *dwCDType* **,**
DWORD* *pdwChkdgt* **)**

Parameters

pwchBarbuf
[in] Heading address of barcode data storage buffer

dwCDType
[in] Check digit type

Bar code type and the corresponding calculation method

Bar Code Type	dwCDType	Calculation Method
EAN(JAN), UPC	'A'	MOD10 (Modulo arithmetic-10)
ITF	'I'	MOD10 (Modulo arithmetic-10)
STF	'H'	MOD10 (Modulo arithmetic-10)
CODABAR (NW-7)	'N'	MOD16 (Modulo arithmetic-16)
CODE-39	'M'	MOD43 (Modulo arithmetic-43)
MSI	'P'	MOD10 (Modulo arithmetic-10)

pdwChkdgt
(out) Address for storing the check digit calculated

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Invalid check digit type. Invalid barcode data. Storage address not specified.

Comment:

If barcode data contains a character(s) out of the specification of the bar code type specified by dwCDType, then this function sets "0" and returns an error code. However, if only the CD position character in barcode data is out of the specification, this function calculates the correct CD and returns it as one-character string.

[Ex 1] **BHT_GetBarChkDgt**(TEXT("494AB4458"), 'A', &dwChkDgt);
"A" and "B" are out of the specification of EAN or UPC, so dwChkDgt is "0" and the function returns an error code.

[Ex 2] **BHT_GetBarChkDgt**(TEXT("4940045X"), 'A', &dwChkDgt);
"X" is out of the specification but it is a CD position character, so this function calculates the correct CD and dwChkDgt is "8."

[Ex 3] **BHT_GetBarChkDgt**(TEXT("a0ef3-a"), 'N', &dwChkDgt);
"e" and "f" are out of the specification of Codabar (NW-7), so dwChkDgt is "0" and the function returns an error code.

[Ex 4] **BHT_GetBarChkDgt**(TEXT("a123Qa"), 'N', &dwChkDgt)
"Q" is out of the specification but it is a CD position character, so this function calculates the correct CD and dwChkDgt is "-."

When dwCDType is TAT (EAN or UPC), Tthis function identifies the EAN or UPC depending upon the data length (number of digits) as listed below. If the data length is a value other than 13, 8, and 7, this function gets "0" and returns an error code.

Data length of barcode data	Bar code type
13	EAN-13 (JAN-13), UPC-A
8	EAN-8 (JAN-8)
7	UPC-E

To check whether the CD is correct: Pass a CD-suffixed barcode data to the BHT_GetBarChkDgt function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("49400458"), 'A', &dwChkDgt);
if ( dwChkDgt == '8' ) {
    printf("CD OK");
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("4940045"));
wcscpy(wchBarData1, wchBarData);
wcscat(wchBarData1, TEXT("0"));
BHT_GetBarChkDgt(wchBarData1, 'A', &dwChkDgt);
wprintf(TEXT("CD = %s%c"), wchBarData, dwChkDgt);
```

Result

```
> CD = 49400458
```

When dwCDType is I (ITF), the length of barcode data must be an even number of two or more digits. If not, this function gets "0" and returns an error code.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("123457"), 'I', &dwChkDgt);
if ( dwChkDgt == '7' ) {
    printf("CD OK");
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("12345"));
wcscpy(wchBarData1, wchBarData);
wcscat(wchBarData1, TEXT("0"));
BHT_GetBarChkDgt(wchBarData1, 'I', &dwChkDgt);
wprintf(TEXT("%s%c"), wchBarData, dwChkDgt);
```

Result

```
> CD = 123457
```

When dwCDType is H (STF), the length of barcode data must be two or more digits. If not, this function gets "0" and returns an error code.
To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("12345678905"), 'H', &dwChkDgt);
if ( dwChkDgt == '5' ) {
    printf("CD OK");
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("1234567890"));
wcscpy(wchBarData1, wchBarData);
wcscat(wchBarData1, TEXT("5"));
BHT_GetBarChkDgt(wchBarData1, 'H', &dwChkDgt);
wprintf(TEXT("%s%c"), wchBarData, dwChkDgt);
```

Result

```
> CD = 12345678905
```

When dwCDType is N (Codabar), the length of barcode data must be three digits or more including start and stop characters. If not, this function gets "0" and returns an error code.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
 BHT_GetBarChkDgt(TEXT("a0123-a"), 'M', &dwChkDgt);
 if ( dwChkDgt == '-' ) {
        printf("CD OK");
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
 wcscpy(wchBarData, TEXT("a0123a"));
 len = wcslen(wchBarData);
 wcsncpy(wchTmp1BarData, wchBarData, len – 1);
 wcscpy(wchTmp2BarData, wchTmp1BarData);
 wcscat(wchTmp2BarData, TEXT("0"));
 wcscat(wchTmp2BarData, &(wchBarData[len – 1]));
 BHT_GetBarChkDgt(wchTmp2BarData) 'M', &dwChkDgt);
 wprintf(TEXT("%s%c%s"), wchTmp1BarData, dwChkDgt, &wchTmp2BarData[len-1]));
```

```
 Result
 > CD = a0123-a
```

When dwCDType is M (Code 39), the length of barcode data must be two or more digits except for start and stop characters. If not, this function gets "0" and returns an error code.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("CODE39W"), 'M', &dwChkDgt);
if ( dwChkDgt == 'W' ) {
        printf("CD OK");
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("CODE39"));
wcscpy(wchBarData1, wchBarData);
wcscat(wchBarData1, TEXT("0"));
BHT_GetBarChkDgt(wchBarData1, 'M', &dwChkDgt);
wprintf(TEXT("%s%c"), wchBarData, dwChkDgt);
```

Result

```
> CD = CODE39W
```

When dwCDType is P (MSI), the length of barcode data must be two or more digits. If not, this function gets "0" and returns an error code. To calculate a two-digit CD, call this function twice.

To check whether the CD is correct: Pass a CD-suffixed barcode data to the **BHT_GetBarChkDgt** function as shown below. If the returned value is equal to the CD, the CD data is suitable for the barcode data.

[Ex]

```
BHT_GetBarChkDgt(TEXT("123456782"), 'P', &dwChkDgt);
if (dwChkDgt == '2' ) {
    printf("CD OK");
}
```

To add a CD to barcode data: Pass barcode data followed by a dummy character to the **BHT_GetBarChkDgt** function as shown below. The returned value will become the CD to be replaced with the dummy character.

[Ex]

```
wcscpy(wchBarData, TEXT("12345678"));
wcscpy(wchBarData1, wchBarData);
wcscat(wchBarData1, TEXT("0"));
BHT_GetBarChkDgt(wchBarData1, 'P', &dwChkDgt);
wprintf(TEXT("%s%c"), wchBarData, dwChkDgt);
```

Result

```
> CD = 123456782
```

BHT_BAR_SetDecodeOptions

Description

Sets the editing function setting value for the decoded result.
This is supported only by the BHT-400B and is not supported by the BHT-400B for the Japanese market.

Syntax

```
DWORD BHT_BAR_SetDecodeOptions (
EN_DCD_OPTIONS_CODE_TYPE enCodeType,
LPVOID pOptions,
DWORD dwLen)
```

Parameters

enCodeType
[in] Code type to be edited

pOptions
[out] Editing function setting value. The addresses for the structure are listed below.

Code type	*EnCodeType*	*pOptions*
UPC-E	EnOptionsUPCE	ST_DCD_UPCE_OPTIONS
UPC-A	EnOptionsUPCA	ST_DCD_UPCA_OPTIONS
EAN-8	enOptionsEAN8	ST_DCD_EAN8_OPTIONS

dwLen
[in] pOptions size (bytes). Sets the value calculated at Sizeof.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	The storage address has not been set. Invalid size specified.

Construction?

The member construction for **EN_DCD_OPTIONS_CODE_TYPE** used to specify the code type is as follows.

```
typedef enum _EN_DCD_OPTIONS_CODE_TYPE {
    enOptionsUPCE,
    enOptionsUPCA,
    enOptionsEAN8,
} EN_DCD_OPTIONS_CODE_TYPE;
```

The member construction for **ST_DCD_UPCE_OPTIONS**, **ST_DCD_UPCA_OPTIONS**, and **ST_DCD_EAN8_OPTIONS** is shown below.

```
typedef struct _ST_DCD_UPCE_OPTIONS {
    BOOL    bConvertToUPCA;
    BOOL    bReportNumsys;
    BOOL    bReportChk;
} ST_DCD_UPCE_OPTIONS,  *PST_DCD_UPCE_OPTIONS;
```

Member name	Default	Details
bConvertToUPCA	FALSE	Used to convert (TRUE) or not convert (FALSE) to UPC-A.
bReportNumsys	FALSE	Used to add (TRUE) or not add (FALSE) a “0” at the beginning.
bReportChk	TRUE	Used to add (TRUE) or not add (FALSE) a C/D.

```
typedef struct _ST_DCD_UPCA_OPTIONS {
   BOOL    bReportNumsys;
   BOOL    bReportChk;
} ST_DCD_UPCA_OPTIONS,  *PST_DCD_UPCA_OPTIONS;
```

Member name	Default	Details
bReportNumsys	TRUE	Used to add (TRUE) or not add (FALSE) a “0” at the beginning.
bReportChk	TRUE	Used to add (TRUE) or not add (FALSE) a C/D.

```
typedef struct _ST_DCD_EAN8_OPTIONS {
   BOOL    bConvertToEAN13;
} ST_DCD_EAN8_OPTIONS,  *PST_DCD_EAN8_OPTIONS;
```

Member name	Default	Details
bConvertToEAN13	FALSE	Used to convert (TRUE) or not convert (FALSE) to EAN-13.

Notes

Authorize reading of the code type prior to conversion when authorizing code reading with **BHT_EnableBar**.
The value acquired with **BHT_ReadBar**, **BHT_GetBarType**, **BHT_GetBarNum**, and **BHT_GetBarInfo** will be the value after conversion.
The set value will only be valid within the application in which it is set. Settings are not updated to other applications.
(Ex.) The following settings are used in order to convert UPC-E codes to UPC-A codes.

```
ST_DCD_UPCE_OPTIONS    stOptions;
DWORD dwLen = sizeof(stOptions);
BHT_EnableBar(TEXT(“FB”), TEXT(“A:C”);
…………
…
/* Acquires current setting */
BHT_BAR_GetDecodeOptions(enOptionsUPCE, (LPVOID)&stOptions, &dwLen);
/* Authorizes conversion to UPC-A */
stOptions.bConvertToUPCA = TRUE;
BHT_BAR_SetDecodeOptions(enOptionsUPCE, (LPVOID)&stOptions, dwLen);
```

BHT_BAR_GetDecodeOptions

Description

Sets the editing function setting value for the decoded result.
This is supported only by the BHT-400B and is not supported by the BHT-400B for the Japanese market.

Syntax

```
DWORD BHT_BAR_GetDecodeOptions (
EN_DCD_OPTIONS_CODE_TYPE enCodeType,
LPVOID pOptions,
DWORD* pdwLen)
```

Parameters

enCodeType
[in] Code type to be edited

pOptions
[out] Editing function setting value. The addresses for the structure are listed below.

Code type	*enCodeType*	*pOptions*
UPC-E	enOptionsUPCE	ST_DCD_UPCE_OPTIONS
UPC-A	enOptionsUPCA	ST_DCD_UPCA_OPTIONS
EAN-8	enOptionsEAN8	ST_DCD_EAN8_OPTIONS

dwLen
[in] pOptions size (bytes). Sets the value calculated at Sizeof.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	The storage address has not been set. Invalid size specified.

Construction

The member construction for **EN_DCD_OPTIONS_CODE_TYPE** used to specify the code type is as follows.

```
typedef enum _EN_DCD_OPTIONS_CODE_TYPE {
    enOptionsUPCE,
    enOptionsUPCA,
    enOptionsEAN8,
} EN_DCD_OPTIONS_CODE_TYPE;
```

The member construction for **ST_DCD_UPCE_OPTIONS**, **ST_DCD_UPCA_OPTIONS**, and **ST_DCD_EAN8_OPTIONS** is shown below.

```
typedef struct _ST_DCD_UPCE_OPTIONS {
    BOOL    bConvertToUPCA;
    BOOL    bReportNumsys;
    BOOL    bReportChk;
} ST_DCD_UPCE_OPTIONS,  *PST_DCD_UPCE_OPTIONS;
```

Member name	Default	Details
bConvertToUPCA	FALSE	Used to convert (TRUE) or not convert (FALSE) to UPC-A.
bReportNumsys	FALSE	Used to add (TRUE) or not add (FALSE) a “0” at the beginning.
bReportChk	TRUE	Used to add (TRUE) or not add (FALSE) a C/D.

```
typedef struct _ST_DCD_UPCA_OPTIONS {
    BOOL    bReportNumsys;
    BOOL    bReportChk;
} ST_DCD_UPCA_OPTIONS,  *PST_DCD_UPCA_OPTIONS;
```

Member name	Default	Details
bReportNumsys	TRUE	Used to add (TRUE) or not add (FALSE) a “0” at the beginning.
bReportChk	TRUE	Used to add (TRUE) or not add (FALSE) a C/D.

```
typedef struct _ST_DCD_EAN8_OPTIONS {
    BOOL    bConvertToEAN13;
} ST_DCD_EAN8_OPTIONS,  *PST_DCD_EAN8_OPTIONS;
```

Member name	Default	Details
bConvertToEAN13	FALSE	Used to convert (TRUE) or not convert (FALSE) to EAN-13.

Notes

The acquired value will be the value set at that application.

(Ex.) The following settings are used in order to convert UPC-E codes to UPC-A codes.

```
    ST_DCD_UPCE_OPTIONS stOptions;
    DWORD dwLen = sizeof(stOptions);
    BHT_EnableBar(TEXT(“FB”), TEXT(“A:C”);
    …………
    …
    /* Acquires current setting */
    BHT_BAR_GetDecodeOptions(enOptionsUPCE, (LPVOID)&stOptions, &dwLen);
    /* Authorizes conversion to UPC-A */
    stOptions.bConvertToUPCA = TRUE;
    BHT_BAR_SetDecodeOptions(enOptionsUPCE, (LPVOID)&stOptions, dwLen);
```

19.2. Backlight API

BHT_SetBltStatus

Description

Control the backlight.

Syntax

```
DWORD BHT_SetBltStatus (
DWORD dwStatus )
```

Parameters

dwStatus

[in] Backlight status

dwStatus	Specification
BHT_BL_ENABLE_ON	Turn on the backlight.
BHT_BL_ENABLE_OFF	Turn off the backlight.
BHT_BL_DISABLE	Disable the backlight.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

BHT_GetBltStatus

Description

Read the backlight status.

Syntax

DWORD BHT_GetBltStatus (
DWORD* *pdwStatus* **)**

Parameters

pdwStatus

[out] Current backlight status

pdwStatus	Specification
BHT_BL_ENABLE_ON	Backlight ON
BHT_BL_ENABLE_OFF	Backlight OFF
BHT_BL_DISABLE	Backlight enabled

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

19.3. Battery API

BHT_GetPowerStatus

Description

Read information about the battery loaded in the BHT body.

Syntax

```
DWORD BHT_GetPowerStatus (
WORD* pwCuOnLine ,
WORD* pwBatteryFlag ,
WORD* pwBatteryVoltage ,
WORD* pwBatteryChemistry )
```

Parameters

pwCuOnLine

[out] Read the BHT state on/off the CU

pwCuOnLine	Specification
AC_LINE_ONLINE	Placed on the CU
AC_LINE_OFFLINE	Not placed on the CU

pwBatteryFlag

[out] Read battery voltage level

pwBatteryFlag	Specification
BHT_BATTERY_FLAG_HIGH	High level (3.9 V ≤ Voltage)
BHT_BATTERY_FLAG_MID	Medium level (3.7 V ≤ Voltage < 3.9 V)
BHT_BATTERY_FLAG_LOW	Low level (3.6 V ≤ Voltage < 3.7 V)
BHT_BATTERY_FLAG_WARNING	Warning level (Voltage < 3.6 V)
BHT_BATTERY_FLAG_CRITICAL	Critical level (Voltage < 3.4 V)
BHT_BATTERY_FLAG_NO_BATTERY	No battery loaded

pwBatteryVoltage

[out] Battery output voltage (mV)

pwBatteryChemistry

[out] Battery type

pwBatteryChemistry	Specification
BATTERY_CHEMISTRY_LION	Lithium ion battery
BATTERY_CHEMISTRY_UNKNOWN	Unknown

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

Comments

(1) The "BHT_BATTERY_FLAG_CRITICAL" or "BHT_BATTERY_FLAG_NO_BATTERY" can be returned only when the grip is connected and loaded with the battery cartridge.

(2) If this function is called when the grip is loaded with the battery cartridge but the BHT body is not, it returns the following:
- Battery voltage level: BHT_BATTERY_FLAG_NO_BATTERY (No battery loaded)
- Battery output voltage: 0 mV
- Battery type: BATTERY_CHEMISTRY_UNKNOWN (Unknown)

BHT_GetPowerStatus2nd

Description

Read information about the battery loaded in the grip.

Syntax

DWORD BHT_GetPowerStatus2nd (
WORD* *pwCuOnLine* **,**
WORD* *pwBatteryFlag* **,**
WORD* *pwBatteryVoltage* **,**
WORD* *pwBatteryChemistry* **)**

Parameters

pwCuOnLine
[out] Read the BHT state on/off the CU

pwCuOnLine	Specification
AC_LINE_ONLINE	Placed on the CU
AC_LINE_OFFLINE	Not placed on the CU

pwBatteryFlag
[out] Read battery voltage level

pwBatteryFlag	Specification
BHT_BATTERY_FLAG_NO_BATTERY	No battery loaded or no grip connected

pwBatteryVoltage
[out] Battery output voltage (mV)
"0" is always returned.

pwBatteryChemistry
[out] Battery type

pwBatteryChemistry	Specification
BATTERY_CHEMISTRY_UNKNOWN	Unknown

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

Comments

With the BHT-400, the battery is not stored in the grip, and therefore the following information is returned if this function is called.

- Battery voltage level	: BHT_BATTERY_FLAG_NO_BATTERY (No battery loaded)
- Battery output voltage	: 0 mV
- Battery type	: BATTERY_CHEMISTRY_UNKNOWN (Unknown)

19.4. LED API

BHT_GetNLedStatus

Description

Read the status of the indicator LED (red/blue).

Syntax

```
DWORD BHT_GetNLedStatus (
DWORD* pdwInfo )
```

Parameters

pdwInfo

[out] Address for storing the LED status

pdwInfo	Specification
LED_OFF	Both red and blue LEDs OFF
RED_LED_ON	Red LED ON
GREEN_LED_ON	Blue LED ON
RED_LED_ON | GREEN_LED_ON	Both red and blue LEDs ON

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Storage address not specified.

BHT_SetNLedStatus

Description

Control the indicator LED (red/blue).

Syntax

```
DWORD BHT_SetNLedStatus (
DWORD dwStatus )
```

Parameters

dwStatus

[in] Controls the LED ON/OFF

dwStatus	Specification
LED_OFF	Turn off both red and blue LEDs
RED_LED_ON	Turn on red LED only
GREEN_LED_ON	Turn on blue LED only
RED_LED_ON | GREEN_LED_ON	Turn on both red and blue LEDs

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Notes:

- When the barcode device file is opened by the **BHT_EnableBar** function, the indicator LED cannot be controlled. Note that if the LED has been specified to be kept off by the **BHT_EnableBar**, the LED can be controlled.
- If the LED is turned on by this function in a user program, it will be kept on until this function turns off the LED even if the user program is terminated.

BHT_GetNLedStatusEx

Description

Read the status of the indicator LED and synchronization LED.

Syntax

```
DWORD BHT_GetNLedStatusEx (
DWORD dwLedDevice ,
DWORD* pdwStatus )
```

Parameters

dwLedDevice

[in] LED device

dwLedDevice	Specification
LED_BAR	Indicator LED

pdwStatus

[out] Address for storing the LED status

pdwStatus	Specification
	If dwLedDevice = LED_BAR
RED_LED_ON	Red LED ON (Blue LED OFF)
GREEN_LED_ON	Blue LED ON (Red LED OFF)
RED_LED_ON | GREEN_LED_ON	Both red and blue LEDs ON

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error. Storage address not specified.

BHT_SetNLedOn

Description

Turn on the indicator LED and/or synchronization LED.

Syntax

```
DWORD BHT_SetNLedOn (
DWORD dwLedDevice ,
DWORD dwLedNum )
```

Parameters

dwLedDevice

[in] LED device

dwLedDevice	Specification
LED_BAR	Indicator LED

dwLedNum

[in] LEDs to be turned on

dwLedNum	Specification
	If dwLedDevice = LED_BAR
RED_LED	Red LED
GREEN_LED	Blue LED
RED_LED | GREEN_LED	Red and blue LEDs

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Notes:

- When the barcode device file is opened by the **BHT_EnableBar** function, the indicator LED cannot be controlled. Note that if the LED has been specified to be kept off by the **BHT_EnableBar**, the LED can be controlled.

- If the LED is turned on by this function in a user program, it will be kept on until this function turns off the LED even if the user program is terminated.

BHT_SetNLedOff

Description

Turn off the indicator LED and/or synchronization LED.

Syntax

```
DWORD BHT_SetNLedOff (
DWORD dwLedDevice ,
DWORD dwLedNum )
```

Parameters

dwLedDevice
[in] LED device

dwLedDevice	Specification
LED_BAR	Indicator LED

dwLedNum
[in] LEDs to be turned off

dwLedNum	Specification
	If dwLedDevice = LED_BAR
RED_LED	Red LED
GREEN_LED	Blue LED
RED_LED | GREEN_LED	Red and blue LEDs

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Notes:

- When the barcode device file is opened by the **BHT_EnableBar** function, the indicator LED cannot be controlled. Note that if the LED has been specified to be kept off by the **BHT_DisableBar**, the LED can be controlled.

19.5. Beeper/Vibrator API

BHT_StartBeep

Description

Drive the beeper or vibrator.

Syntax

```
DWORD BHT_StartBeep (
DWORD dwOnTime ,
DWORD dwOffTime ,
WORD wRepCnt ,
WORD wFreq )
```

Parameters

dwOnTime
[in] ON-duration (in units of 100 ms), Entry range: 0 to 255

dwOffTime
[in] OFF-duration (in units of 100 ms), Entry range: 0 to 255

wRepCnt
[in] Number of repetitions, Entry range: 0 to 255

wFreq
[in] Frequency (Hz) , Entry range: 0 to 32767

Specification of 0, 1 or 2 to wFeq produces the special beeper effects as listed below.

wFreq	Tone	Frequency (Hz)
0	Low-pitched	698
1	Medium-pitched	1396
2	High-pitched	2793

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Comment:

- The system functions allow the beeper volume to be changed. (Refer to Section 5.2.)
- Specification of any of 3 through 667 to wFreq deactivates the beeper or vibrator.
- Specification of zero to dwOnTime deactivates the beeper or vibrator.
- Specification of a value except zero to dwOnTime and wRepCnt and specification of zero to dwOffTime keep the beeper sounding.
- For your reference, the relationship between the frequencies and the musical scale is listed below.

	Scale 1	Scale 2	Scale 3	Scale 4
do (C)	-	1046	2093	4186
do# (C#)	-	1108	2217	
re (D)	-	1174	2349	
re# (D#)	-	1244	2489	
mi (E)	-	1318	2637	
fa (F)	698	1396	2793	
fa# (F#)	739	1479	2959	
sol (G)	783	1567	3135	
sol# (G#)	830	1760	3520	
la (A)	880	1760	3520	
la (A#)	932	1864	3729	
si (B)	987	1975	3951	

BHT_StartBeeperOnly

Description

Drive the beeper.

Syntax

```
DWORD BHT_StartBeeperOnly (
DWORD dwOnTime ,
DWORD dwOffTime,
WORD wRepCnt ,
WORD wFreq )
```

Parameters

dwOnTime
[in] ON-duration (in units of 100 ms), Entry range: 0 to 255

dwOffTime
[in] OFF-duration (in units of 100 ms), Entry range: 0 to 255

wRepCnt
[in] Number of repetitions, Entry range: 0 to 255

wFreq
[in] Frequency (Hz) , Entry range: 0 to 32767

Specification of 0, 1 or 2 to wFeq produces the special beeper effects as listed below.

wFreq	Tone	Frequency (Hz)
0	Low-pitched	698
1	Medium-pitched	1396
2	High-pitched	2793

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Comment:

- The system functions allow the beeper volume to be changed. (Refer to Section 5.2.)
- Specification of any of 3 through 667 to wFreq deactivates the beeper or vibrator.
- Specification of zero to dwOnTime deactivates the beeper or vibrator.
- Specification of a value except zero to dwOnTime and wRepCnt and specification of zero to dwOffTime keep the beeper sounding.
- For your reference, the relationship between the frequencies and the musical scale is listed below.

	Scale 1	Scale 2	Scale 3	Scale 4
do (C)	-	1046	2093	4186
do# (C#)	-	1108	2217	
re (D)	-	1174	2349	
re# (D#)	-	1244	2489	
mi (E)	-	1318	2637	
fa (F)	698	1396	2793	
fa# (F#)	739	1479	2959	
sol (G)	783	1567	3135	
sol# (G#)	830	1760	3520	
la (A)	880	1760	3520	
la (A#)	932	1864	3729	
si (B)	987	1975	3951	

BHT_StartVibrationOnly

Description

Drive the vibrator.

Syntax

```
DWORD BHT_StartVibrationOnly (
DWORD dwOnTime ,
DWORD dwOffTime ,
WORD wRepCnt )
```

Parameters

dwOnTime
[in] ON-duration (in units of 100 ms), Entry range: 0 to 255

dwOffTime
[in] OFF-duration (in units of 100 ms), Entry range: 0 to 255

wRepCnt
[in] Number of repetitions, Entry range: 0 to 255

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

19.6. Wireless Communication API

BHT_RF_Open

Description

Open the wireless LAN device and enable wireless communication.

Syntax

```
DWORD BHT_RF_Open ( void )
```

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.

BHT_RF_OpenEx

Description

Sets the communication format, opens the wireless LAN device and enables wireless communication.

Syntax

DWORD BHT_RF_OpenEx (
DWORD *dwOpt* **)**

Parameters

dwOpt

[in] Communication format

dwOpt	Specification
COMM_NORMAL	Wireless communication open
COMM_CONTINUOUS	Wireless communication continuously open

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_INVALID_PARAMETER	Parameter error

BHT_RF_Close

Description

Close the wireless LAN device and disable wireless communication.

Syntax

```
DWORD BHT_RF_Close ( void )
```

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion

BHT_RF_CloseEX

Description

Closes the wireless LAN device for the set format and disables wireless communication.

Syntax

```
DWORD BHT_RF_CloseEx (
DWORD dwOpt )
```

Parameters

dwOpt

[in] Communication format

dwOpt	Specification
COMM_NORMAL	Wireless communication open
COMM_CONTINUOUS	Wireless communication continuously open

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error

BHT_RF_IoControl

Description

Sends a control command to the driver and performs an operation corresponding to that command.

Syntax

DWORD BHT_RF_IoControl (
DWORD *Oid* **,**
LPVOID *lpInBuf* **,**
DWORD *nInBufSize* **,**
LPVOID *lpOutBuf* **,**
DWORD *nOutBufSize* **,**
LPVOID *lpBytesReturned* **)**

Parameters

Oid
[in] Control command ID

Oid	Specification
RF_UPDATE_PROFILE	Updates the profile settings for the BHT wireless registry. (*1)
RF_COMMIT_PROFILE	Updates the changed parameter value to the driver. (*2)
RF_SET_PROFILE	Selects the profile to be edited.
RF_REMOVE_PROFILE	Deletes the profile.
RF_GET_PROFILE_COUNT	Acquires the number of completed profiles.
RF_GET_PROFILE_KEY	Acquires the profile key.

(*1) Copies values set at the ZeroConfig GUI to the BHT wireless registry referenced by the wireless driver.
(*2) Updates values set at this API to ZeroConfig.

lpInBuf
[in] Header address for buffer in which input data is stored

nInBufSize
[in] Size of buffer in which input data is stored (Bytes)

lpOutBuf
[out] Header address for buffer in which output data is stored

nOutBufSize
[out] Size of buffer in which output data is stored (Bytes)

lpBytesReturned
[out] Size of actually acquired output data (Bytes)

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error Storage header address unset
ERROR_NOT_READY	Not in ZeroConfig mode
ERROR_NOT_ENOUGH_MEMORY	The number of profiles has exceeded the maximum (16).
ERROR_NOT_FOUND	The relevant profile cannot be found.
ERROR_FILE_NOT_FOUND	The relevant file cannot be found.

The details set for each argument differ for each command.

Oid	lpInBuf	nInBufSize	lpOutBuf	nOutBufSize
RF_UPDATE_PROFILE	–	–	–	–
RF_COMMIT_PROFILE	–	–	–	–
RF_SET_PROFILE	ST_RF _PROFILE_KEY (*3)	ST_RF_PROFILE _KEY size	–	–
RF_REMOVE_PROFILE	ST_RF _PROFILE_KEY	ST_RF_PROFILE _KEY size	–	–
RF_GET_PROFILE_COUNT	–	–	Profile count storage variable	sizeof(DWORD)
RF_GET_PROFILE_KEY	Profile index to be acquired	sizeof(DWORD)	ST_RF _PROFILE_KEY	ST_RF_PROFILE _KEY size

(*3) Use ESSID and Infrastructure mode to specify the profile. Create a new profile if no profile can be found corresponding to the specified ESSID and Infrastructure mode.

The ST_RF_PROFILE_KEY configuration is as follows.

Construction

```
Typedef struct _ST_RF_PROFILE_KEY {
TCHAR szESSID [SSID_MAX+1];         // ESSID
UCHAR ucReserved [2];               // reserved
DWORD dwInfraMode;                  // Infrastructure
mode
} ST_RF_PROFILE_KEY;
```

Members

szESSID
SSID specified character string

dwInfraMode
Infrastructure mode

dwInfraMode	Specification
INFRA_INFRASTRUCTURE	Infrastructure
INFRA_ADHOC	Ad-hoc

BHT_RF_Synchronize

Description

Get the association status.

Syntax

DWORD BHT_RF_Synchronize (
long *lTimeout* **,**
long* *plSync* **)**

Parameters

lTimeout
[in] Timeout (in units of 100 ms)

lTimeout	Specification
> 0	Confirm the synchronization status until timeout
0	Check the synchronization status immediately and return the result
-1	Try to synchronize with the access point until synchronized

plSync
[out] Address for storing the synchronization result

plSync	Specification
0	Successfully synchronized
-1	Synchronization incomplete (timed out)

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_NOT_READY	Device not ready.
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.

Notes

This function should be executed after execution of the **BHT_RF_Open** function. Otherwise, the called function returns "ERROR_NOT_READY."

BHT_RF_GetParamInt

Description

Read integer from the wireless communications parameter.

Syntax

```
DWORD BHT_RF_GetParamInt (
DWORD dwParam ,
DWORD* pdwData ,
DWORD* pdwLen )
```

Parameters

dwParam
[in] Parameter number

dwParam	Specification
P_INT_POWERSAVE	Power mode dwData = P_PWRSAVE_CAM = P_PWRSAVE_PSP
P_INT_AUTHENTICATE	Authentication method dwData = P_AUTH_OPEN = P_AUTH_SHARED = P_AUTH_WPA = P_AUTH_WPAPSK
P_INT_ENCRYPTION	Encryption dwData = P_ENCRYPT_DISABLE = P_ENCRYPT_WEP = P_ENCRYPT_TKIP
P_INT_8021X	802.1x authentication (EAP type) dwData = P_8021X_DISABLE = P_8021X_PEAP = P_8021X_TLS
P_INT_PRIORITY	Profile priority dwData = 1 (high) to 16 (low)
P_INT_INDEXKEY	Index key dwData = 1 to 4

pdwData
[out] Address for storing data obtained

pdwLen
[out] Address for storing the length of data obtained
If the function succeeds in getting data, the length of data obtained is always 4.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error Address for storing data obtained not specified.
ERROR_NOT_SUPPORTED	Not supported

BHT_RF_SetParamInt

Description

Write integer to the wireless communications parameter.

Syntax

```
DWORD BHT_RF_SetParamInt (
DWORD dwParam ,
const DWORD* pdwData ,
DWORD dwLen )
```

Parameters

dwParam

[in] Parameter number

dwParam	Specification
P_INT_POWERSAVE	Power mode dwData = P_PWRSAVE_CAM = P_PWRSAVE_PSP
P_INT_AUTHENTICATE	Authentication method dwData = P_AUTH_OPEN = P_AUTH_SHARED = P_AUTH_WPA = P_AUTH_WPAPSK
P_INT_ENCRYPTION	Encryption dwData = P_ENCRYPT_DISABLE = P_ENCRYPT_WEP = P_ENCRYPT_TKIP
P_INT_8021X	802.1x authentication (EAP type) dwData = P_8021X_DISABLE = P_8021X_PEAP = P_8021X_TLS
P_INT_PRIORITY	Profile priority dwData = 1 (high) to 16 (low)
P_INT_INDEXKEY	Index key dwData = 1 to 4

pdwData

[in] Destination address where the set data is to be stored

dwLen

[in] Length of data

The data length is always 4.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error Address for storing data obtained not specified.
ERROR_NOT_SUPPORTED	Not supported

BHT_RF_GetParamStr

Description

Read string from the wireless communications parameter.

Syntax

```
DWORD BHT_RF_GetParamStr (
DWORD dwParam ,
TCHAR* pwchData ,
DWORD* pdwLen )
```

Parameters

dwParam

[in] Parameter number

dwParam	Specification
P_STR_VERSION	Driver version
P_STR_FW_VERSION	Firmware version
P_STR_MACADDRESS	MAC address

pwchData

[out] Heading address of the storage buffer for data obtained

pdwLen

[out] Length of data obtained

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.
ERROR_NOT_SUPPORTED	Not supported

BHT_RF_SetParamStr

Description

Write character string to the wireless communications parameter.

Syntax

```
DWORD BHT_RF_SetParamStr (
DWORD dwParam ,
TCHAR* pwchData ,
DWORD dwLen )
```

Parameters

dwParam

[in] Parameter number

dwParam	Specification
P_STR_WEPKEY1	WEP Key 1
P_STR_PRESHAREDKEY	Pre Shared Key

pwchData

[in] Heading address of the storage buffer for data specified

dwLen

[in] Length of data specified

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error
ERROR_NOT_SUPPORTED	Not supported.

BHT_RF_GetInfoInt

Description

Read integer from the communications parameter.

Syntax

```
DWORD BHT_RF_GetInfoInt (
DWORD dwType ,
DWORD* pdwInfo )
```

Parameters

dwType

[in] Type of information to be read out

dwType	Specification
P_RATE_INFO	Current communication speeds: No link → P_RATE_NOT_LINK 1Mbps → P_RATE_1MBPS 2Mbps → P_RATE_2MBPS 5.5Mbps → P_RATE_5_5MBPS 11Mbps → P_RATE_11MBPS Above 11Mbps → P_RATE_OVER11MBPS
P_RATE_INFO2	Current communication speeds (Units: KHz 100bps): [Ex.] 5.5Mbps → 55,000 11Mbps → 110,000 54Mbps → 540,000
P_CHANNEL_INFO	Frequency channel currently used

pdwInfo

[out] Address for storing info read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_NOT_READY	Device not ready.
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.

BHT_RF_GetInfoStr

Description

Read string from the communications parameter.

Syntax

```
DWORD BHT_RF_GetInfoStr (
DWORD dwType ,
TCHAR* pwchInfo )
```

Parameters

dwType

[in] Type of information to be read out

dwType	Specification
P_APMAC_INFO	MAC address of AP being linked

pwchInfo

[out] Heading address of the storage buffer for info read

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_NOT_READY	Device not ready.
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.

BHT_RF_GetSiteSurvey

Description

Get the quality of the communications link.

Syntax

```
DWORD BHT_RF_GetSiteSurvey (
DWORD* pdwStrength ,
DWORD* pdwBeacon ,
DWORD* pdwLink )
```

Parameters

pdwStrength

[out] Current signal strength, 0 to 100 (%)

pdwBeacon

[out] The same value as *pdwStrength*, 0 to 100 (%)

pdwLink

[out] Current link quality

pdwLink	Specification
LQ_UNSYNC	Not associated
LQ_POOR	Poor communications link (less than 20%)
LQ_FAIR	Fair communications link (20% or more and less than 40%)
LQ_GOOD	Good communications link (40% or more and less than 75%)
LQ_EXCELLENT	Excellent communications link (75% or more for send and receive)

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
No NIC device found.	No NIC device found.
ERROR_NOT_READY	Device not ready.
ERROR_INVALID_PARAMETER	Parameter error Storage address not specified.

19.7. Flash File System API

You can use Microsoft Win32 API by accessing the flash memory file in applications. To access it, specify the "FLASH" folder (on which the flash memory file is mounted) to the pathname parameter.

[Ex] Create a directory named "test" on the root directory of the flash memory file.

CreateDirectory (TEXT("\\FLASH\\test"), NULL);

API implementation for the flash memory system

Win32 API	Implementation
CloseHandle	Fully
CreateDirectory	Fully
CreateFile	Fully
DeleteAndRenameFile	Partially
DeleteFile	Partially
DeviceIOControl	Fully
FindClose	Fully
FindFirstFile	Partially
FindNextFile	Partially
FlushFileBuffers	Fully
GetDiskFreeSpace	Fully
GetFileAttributes	Fully
GetFileInformationByHandle	Fully
GetFileSize	Partially
GetFileTime	Partially
MoveFile	Partially
ReadFile	Fully
RemoveDirectory	Partially
SetEndOfFile	Fully
SetFileAttributes	Fully
SetFilePointer	Partially
SetFileTime	Partially
WriteFile	Fully

Fully: Windows CE API is fully implemented.
Partially: Windows CE API is partially implemented. Refer to the next page.

Restrictions on the use of API

If a filepath specified to access any interface in Win 32 API exceeds the length specified by MAX_PATH, the BHT cannot operate normally. Specify the filepath within the range defined by MAX_PATH.

Other restrictions are listed below.

API	Content	
DeleteAndRenameFile	Restriction	If the power to the BHT is shut down during transfer of a data file, the original file may be lost.
	Solution	None
FindFirstFile	Restriction	At the normal end of this API, any file existing in the same directory and matching the pattern for the next search cannot be deleted or moved. Furthermore, any parent directory cannot be changed or deleted.
	Solution	Close the handle by using **CloseHandle** before change or deletion.
FindNextFile	Restriction	Same as **FindFirstFile**
	Solution	Same as **FindFirstFile**
GetFileTime	Restriction	Can obtain only the day and time for the created file.
	Solution	None
MoveFile	Restriction	Same as **DeleteAndRenameFile**
	Solution	None
SetFileTime	Restriction	If these APIs are called together with other APIs, there are times when processing will fail.

Initialization

You can initialize the FLASH folder in System Menu. For details, refer to the "BHT-400B/400BW-CE User's Manual."

Scandisk

If the power to the BHT is shut down when the BHT is writing data to the flash file, some broken file fragments may remain on the flash file clusters. To remove or clear those fragments, run Scandisk on the flash file. For details, refer to the "BHT-400B/400BW-CE User's Manual."

19.8. OS Updating API

BHT_SystemModify

Description

Update the BHT OS.

Syntax

```
DWORD BHT_SystemModify (
TCHAR* pwszFileName ,
DWORD dwMode )
```

Parameters

pwszFileName

[in] Pointer filename that points a NULL-appended character string containing the OS reconfiguration filename. Either "\\SysModify\\" or "/SysModify/" must be specified as the path name.

dwMode

[in] Reboot mode after turning the power off

dwMode	Specification
SYSMDFY_POWEROFF	Turn the power off. (Cold-boot the BHT at the next power on)
SYSMDFY_REBOOT	Perform a cold boot.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_FILE_NOT_FOUND	Specified file or device not found. (OS reconfiguration file not found.)
ERROR_INVALID_PARAMETER	Parameter error.
ERROR_BAD_FORMAT	The OS update file is incorrect.

Comment:

It is necessary to execute the **BHT_ShutdownSystem** (BHT_PWR_SYSMODIFY) function in order to secure an area for the OS update file to be stored prior to executing these functions.

19.9. Bluetooth API

BHT_BT_PowerOn

Description

Turns ON the Bluetooth device power supply and enables Bluetooth.

Syntax

```
DWORD BHT_BT_PowerOn ( void )
```

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	The unit is not equipped with a Bluetooth device.

BHT_BT_PowerOff

Description

Turns OFF the Bluetooth device power supply and disables Bluetooth.

Syntax

DWORD BHT_BT_PowerOff (void)

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	The unit is not equipped with a Bluetooth device.

BHT_BT_GetPowerStatus

Description

Acquires the Bluetooth device power status.

Syntax

```
DWORD BHT_BT_GetPowerStatus (
DWORD *pdwStatus )
```

Parameters

pdwStatus

［in］ Device status storage location address
The following values are returned for the device status.

pdwStatus	Specification
BHT_BT_POWER_ON	The Bluetooth device power is ON.
BHT_BT_POWER_OFF	The Bluetooth device power is OFF.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_DEV_NOT_EXIST	The unit is not equipped with a Bluetooth device.
ERROR_INVALID_PARAMETER	Storage location address unset

19.10. Other APIs

BHT_WaitEvent

Description

Make the system wait until the specified event or timeout occurs.

Syntax

```
DWORD BHT_WaitEvent (
DWORD dwEvtNum ,
DWORD dwEvtMask ,
DWORD dwTimeOut ,
DWORD* pdwSignalEvent )
```

Parameters

dwEvtNum

[in] Number of events to wait

dwEvtMask

[in] Waiting event mask

dwEvtMask	Specification
EVT_MASK_KEYDOWN	Key depressed
EVT_MASK_TRGDOWN	Trigger switch depressed
EVT_MASK_TCHUP	Stylus released
EVT_MASK_DECODE	Decoding completed
EVT_MASK_RECEIVE EVT_MASK_RECEIVE_IRDA	Data reception (IrDA interface)
EVT_MASK_RECEIVE_RS232C	Data reception(Serial interface)
EVT_MASK_RECEIVE_USB	Data reception(USB interface)
EVT_MASK_LASERKEYDOWN	Laser lighting key depressed

NOTE: ORing these events enables the BHT to wait for the two or more events.

dwTimeOut

[in] Timeout period (ms)

pdwSignalEvent

[out] Address for storing an event mask that occurred

pdwSignalEvent	Specification
EVT_MASK_KEYDOWN	Key depression
EVT_MASK_TRGDOWN	Trigger switch depression
EVT_MASK_TCHUP	Stylus release
EVT_MASK_DECODE	Decoding complete
EVT_MASK_RECEIVE EVT_MASK_RECEIVE_IRDA	Data reception(IrDA interface)
EVT_MASK_RECEIVE_RS232C	Data reception(Serial interface)
EVT_MASK_RECEIVE_USB	Data reception(USB interface)
EVT_MASK_LASERKEYDOWN	Laser lighting key depression
EVT_MASK_TIMEOUT	Timeout

NOTE: When two or more events except WAIT_TIMEOUT occur concurrently, an ORed value of these events is stored in the address.

To make the system wait for occurrence of any event infinitely, specify INFINITE in dwTimeOut.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error. Storage address not specified

Comment:

The following six types of events can be specified:
- Depression of any key
- Depression of the trigger switch
- Stylus release
- Decoding completion
- Data reception (in IrDA interface, Serial interface, USB interface)
- Depression of the laser lighting key

Specifying two or more events concurrently using this function allows the system to wait for occurrence of any of these events. To wait for other events in addition to events listed above, add desired events using macros with the event names defined by the BHTLIB.h library.

[Ex] Wait for occurrence of entry by any key depression or decoding completion for 10 seconds

```
BHT_WaitEvent (2, EVT_MASK_KEYDOWN | EVT_MASK_DECODE,
  10 * 1000, &dwSignalEvent);
```

BHT_WaitStandbyEvent

Description

Make the system wait until the specified event occurs.

Syntax

```
BHT_WaitStandbyEvent (
DWORD dwEvtNum ,
DWORD dwEvtMask ,
DWORD* pdwSignalEvent )
```

Parameters

dwEvtNum

[in] Number of events to wait

dwEvtMask

[in] Events to wait

dwEvtMask	Specification
EVT_MASK_KEYDOWN	Key depression
EVT_MASK_TRGDOWN	Trigger switch depression
EVT_MASK_TCHUP	Stylus release
EVT_MASK_DECODE	Decoding complete
EVT_MASK_RECEIVE EVT_MASK_RECEIVE_IRDA	Data reception(IrDA interface)
EVT_MASK_RECEIVE_RS232C	Data reception(Serial interface)
EVT_MASK_RECEIVE_USB	Data reception(USB interface)
EVT_MASK_LASERKEYDOWN	Laser lighting key depression

pdwSignalEvent

[out] Address for storing events that occurred

pdwSignalEvent	Specification
EVT_MASK_KEYDOWN	Key depression
EVT_MASK_TRGDOWN	Trigger switch depression
EVT_MASK_TCHUP	Stylus release
EVT_MASK_DECODE	Decoding complete
EVT_MASK_RECEIVE EVT_MASK_RECEIVE_IRDA	Data reception(IrDA interface)
EVT_MASK_RECEIVE_RS232C	Data reception(Serial interface)
EVT_MASK_RECEIVE_USB	Data reception(USB interface)
EVT_MASK_LASERKEYDOWN	Laser lighting key depression

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error. Storage address not specified

Comment:

The following six types of events can be specified:

- Depression of any key
- Depression of the trigger switch
- Stylus release
- Decoding completion
- Data reception (in IrDA interface, Serial interface, USB interface)
- Depression of the laser lighting key

Unlike **BHT_WaitEvent**, this function lets the CPU enter the standby mode when making the system wait, reducing power consumption. Note that execution of any other active thread will be suspended during execution of this function.

BHT_ShutdownSystem

Description
Turn off the BHT and boot the BHT according to the mode specified by the parameter.

Syntax
```
DWORD BHT_ShutdownSystem (
DWORD dwMode )
```

Parameters
dwMode
[in] Power-off mode

dwMode	Specifications
BHT_PWR_WARM	Turn off and warm-boot the BHT. No power-off action is required. The contents in the RAM can be retained.
BHT_PWR_SUSPEND	Transfer control to the suspended mode. Pressing the power key starts the BHT. The contents in the RAM will be retained as long as the sub-battery is charged.
BHT_PWR_COLD_REGINIT	Turn off and cold-boot the BHT. Pressing the power key starts the BHT. The contents in the RAM will be lost and the system registry will be initialized.
BHT_PWR_COLD_REGREMAIN	Turn off and cold-boot the BHT. Pressing the power key starts the BHT. The contents of the system registry will be saved into the non-volatile memory in powering-off sequence and restored at the cold boot.
BHT_PWR_SYSMODIFY	A cold boot is performed automatically after turning OFF the power. An area is secured in order to store the OS.
BHT_PWR_COLD	A cold boot is performed automatically after turning OFF the power. If the registry has been saved, the BHT is booted based on the values for that registry, however, if it has not been saved, the BHT is booted based on the values for the default registry value.

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	Parameter error.

Comment:
Any of the following five modes can be specified:

- Warm boot*
- Suspend
- Cold boot* with Registry initialization (The Registry backup will also be lost.)
- Cold boot* without Registry initialization
- Cold boot* with securing of area to store OS
- Cold boot*

*Contents of the memory after warm-/cold-booting the BHT

	After warm booting	After cold booting
Files in the FLASH folder	Retained	Retained
Files in the RAM	Retained	Erased
Contents of the Registry	Retained	Retained (Note)
Data being edited	Erased	Erased

(Note) If the Registry has been backed up, the backup will apply.

BHT_RegStore

Description

Save the registry.

Syntax

```
DWORD BHT_RegStore ( void )
```

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_WRITE_FAULT	Failed to save registry.

Chapter 20. Programming Using OCX (OLE Customer Control)

The BHT-400 Software Development Kit (BHT-400 SKD) provides ActiveX Control that can be used for programming applications for barcode reading and file transfer.
This chapter gives information for using the ActiveX control.

20.1. System Requirements

(1) BHT-400 Software Development Kit
(2) Control files .ocx for the desktop
- Scanner400.ocx: For barcode reading
- FileTransfer400.ocx: For file transmission
- FileTransferPC.ocx: For file transmission(for PC)

20.2. Installation

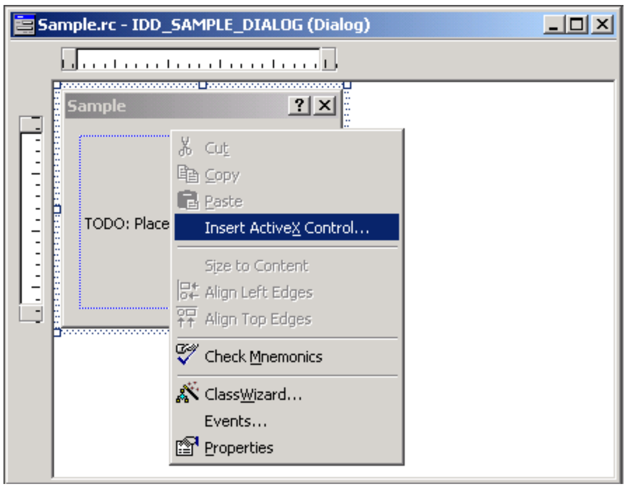
(1) Copy the .ocx files in the BHT-400 Software Development Kit CD onto the appropriate folder of your PC.
(2) Open the DOS command prompt and change the directory to the folder including the .ocx files.
(3) Run the following two commands on the command line (>):

```
> regsvr32 Scanner400.ocx
> regsvr32 FileTransfer400.ocx
> regsvr32 FileTrrnaferPC.ocx
```

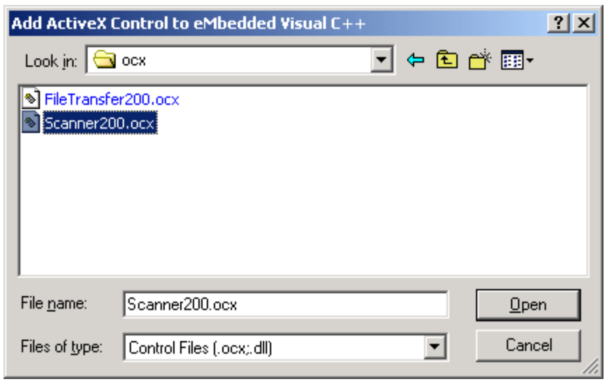
20.3. Using OCX

In Microsoft Foundation Class (MFC)

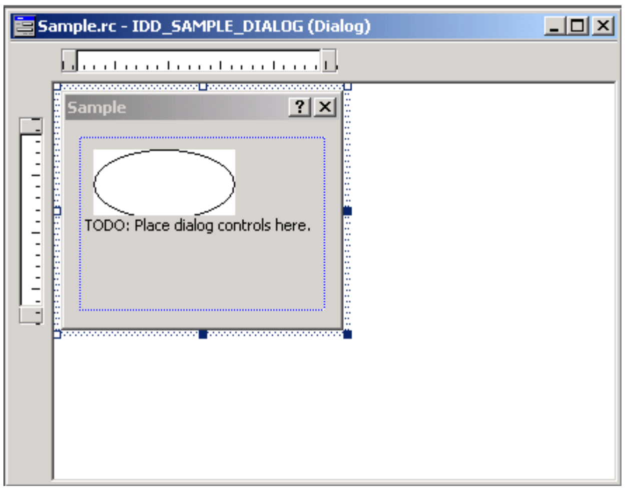
(1) Open an existing project or create a new project in eMbedded Visual C++.
(2) Insert the newly installed ActiveX control into eMbedded Visual C++. (This step is required only when the ActiveX control is first used after installation.)
(3) -1 Point and right-click the active window or dialog, then choose "Insert ActiveX Control" command on the dropdown menu.



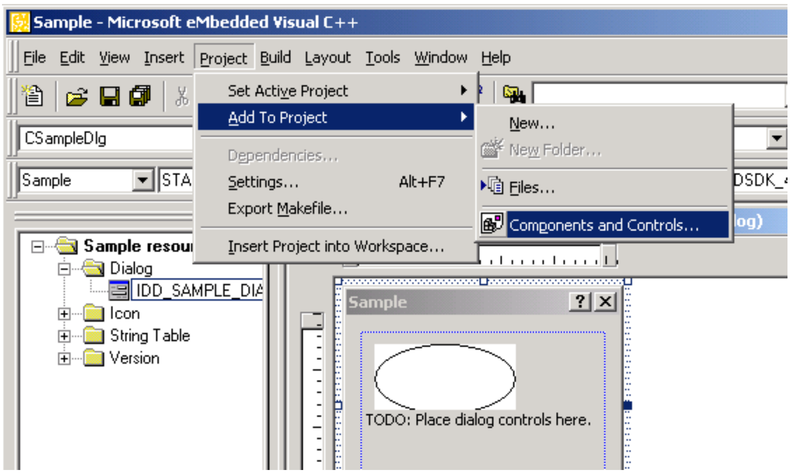
(2)-2 Click **Add Control** and choose the newly installed OCX by clicking **Open**.


Add ActiveX Control to eMbedded Visual C++
Look in: ocx
FileTransfer200.ocx
Scanner200.ocx
File name: Scanner200.ocx
Files of type: Control Files (.ocx;.dll)
Open
Cancel

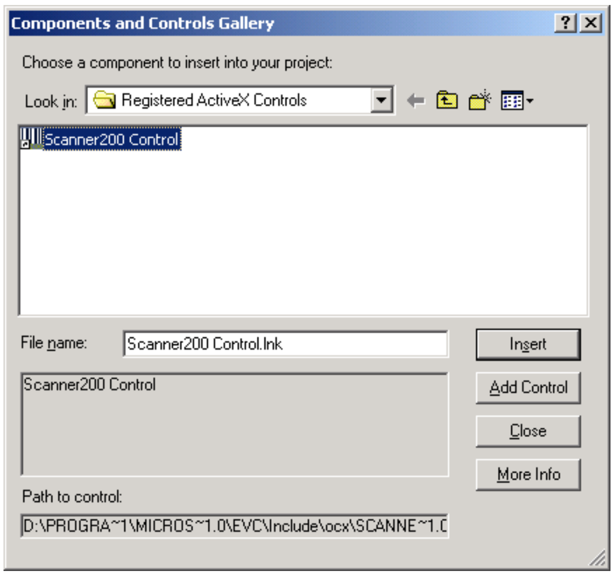
(2)-3 Click **OK**, and the control is pasted as shown below.


Sample.rc - IDD_SAMPLE_DIALOG (Dialog)
Sample
TODO: Place dialog controls here.

(3) Add the control to the project.
(3)-1 Click **Project**–**Add to Project**–**Components and Controls** on the menu bar as shown below.



(3)-2 Select the installed .OCX file.


Components and Controls Gallery

Choose a component to insert into your project:

Look in: Registered ActiveX Controls

Scanner200 Control

File name: Scanner200 Control.lnk

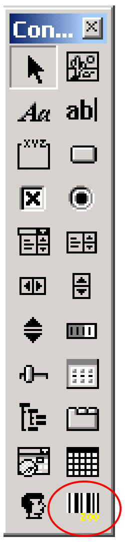
Scanner200 Control

Path to control:

D:\PROGRA~1\MICROS~1.0\EVC\Include\ocx\SCANNE~1.C

(3)-3 Click **Insert**, and the message “Do you insert component?” pops up. Click **OK,** and specify an appropriate class name, header filename and implement filename.

(3)-4 If **OK** is clicked, an icon of the added control will be added to the dialog as shown below (red-circled).



(4) Following ClassWizard, assign a member variable to the inserted control.

20.4. Scanner Control

20.4.1. Properties

Name and type		R/W	Value	Default value	Description
eVC++					
GetPortOpen **SetPortOpen**	BOOL	R/W	TRUE or FALSE	FALSE	Enable/disable flag for barcode reading TRUE: Enable FALSE: Disable
GetReadMode **SetReadMode**	CString	R/W	(*1)	"FB"	Character string for specifying the read mode (*1), (*2)
GetReadType **SetReadType**	CString	R/W	(*1)	"A,I:4-99, M:1-99, N:3-99, L:1-99, K:1-99, H:3-99"	Character string for specifying the enable read code (*1), (*2)
GetBufferData **SetBufferData**	CString	R	-	""	Data stored in the barcode buffer (*1)
GetBufferCount **SetBufferCount**	long	R	-	0	Number of digits stored in the barcode buffer (*1)
GetBufferType **SetBufferType**	long	R	-	0	Barcode type stored in the barcode buffer (*1)
GetLastCount **SetLastCount**	long	R	-	0	Number of digits in the barcode read last
GetLastType **SetLastType**	long	R	-	0	Barcode type read last
GetErrorStatus **SetErrorStatus**	long	R/W	(*3)	ERROR_SUCCESS	Error code that occurred last (*4)
GetWaitStby **SetWaitStby**	BOOL	R/W	TRUE or FALSE	FALSE	Whether or not the control transfers to the standby mode before decoding completes TRUE: Transfer FALSE: Not transfer

(*1) Refer to **BHT_EnableBar** function.
(*2) Even if a value out of the range is specified, no error occurs. If TRUE is set to the portOpen property with the value being out of the range, an error occurs.
(*3) For details about error codes, refer to Section 20.4.4 Error Codes."
(*4) A new error code overwrites the old one whenever an error occurs. The ERROR_SUCCESS does not overwrite.

20.4.2. Methods

GetChkDigit

Description

Calculate a check digit (CD) of the barcode data according to the specified calculation method. (Refer to the **BHT_GetBarChkDgt** function.)

Syntax

```
long GetChkDigit (
TCHAR* BarData ,
short ChkDgtType )
```

Parameters

BarData
[in] Character string of the barcode

ChkDgtType
[in] Check digit type

(For details, refer to the **BHT_GetBarChkDgt** function.)

Return value

Value of the check digit calculated

20.4.3. Event Callback Function

DecodeDone

Description

This function is called when decoding is successfully completed. It reads out the bufferData property to get data decoded.

Syntax

```
void OnDecodeDone ( void )
```

Parameters

None

Return value

None

20.4.4. Error Codes

If an error occurs during access to properties or during calling to methods, the error code will be stored into the errorStatus variable.

Error Code Table

Propertie or Method	Name	Content
portOpen	ERROR_TOO_MANY_OPEN_FILES	Barcode reading enabled (when flag is TRUE).
	ERROR_INVALID_PARAMETER	readMode or readType out of the range (when flag is TRUE)
	ERROR_INVALID_HANDLE	Barcode reading disabled (when flag is FALSE)
BufferData	ERROR_INVALID_HANDLE	Barcode reading disabled
GetChkDigit	ERROR_INVALID_PARAMETER	Check digit type out of the range or invalid barcode data

20.4.5. Coding Sample

```
/* Initialize main dialog */
BOOL CBarOCXDlg::OnInitDialog()
{
   CDialog::OnInitDialog();

   …………………………..….
   …………………………..….
   /* Enable barcode reading */
   m_ScanCtrl.SetPortOpen(TRUE);

   return TRUE;
}

/* Initialize main dialog */
void CBarOCXDlg::OnDestroy()
{
   /* Disable barcode reading */
   m_ScanCtrl.SetPortOpen(FALSE);

   CDialog::OnDestroy();
}

/* Callback for decoding completion */
void CBarOCXDlg::OnDecodeDoneScannerctrl()
{
   CString BarData; /* Read data */

   /* Read data from buffer */
   BarData = m_ScanCtrl.GetBufferData();
   /* Display */
   …………………………..….
  …………………………..….
}
```

20.5. File Transfer Control

20.5.1. Properties

Name		R/W	Value	Default value	Content
eVC++					
GetPort **SetPort**	short	R/W	COM1 COM4	COM4	COM port
GetBaud **SetBaud**	long	R/W	CBR_300 (*1) CBR_600 (*1) CBR_1200 (*1) CBR_2400 CBR_4800 (*1) CBR_9600 CBR_19200 CBR_38400 CBR_57600 CBR_115200	CBR_9600	Transmission rate
GetParity **SetParity**	short	R/W	NOPARITY ODDPARITY (*1) EVENPARITY (*1)	NOPARITY	Parity
GetStopBit **SetStopBit**	short	R/W	ONESTOPBIT TWOSTOPBITS (*1)	ONESTOPBIT	Stop bit
GetPath **SetPath**	CString LPCTSTR	R/W	Absolute path starting with \ sign	"\"	Folder to store send files Folder to store receive files
GetTransferring **EventInterval** **SetTransferring** **EventInterval**	long	R/W	0 to 2147483647	0	Transferring Event interval during transmission (in units of 100 ms) 0 for no event
GetLinkTimeout **SetLinkTimeout**	long	R/W	0 to 65535	30 (30sec.)	Time required from commencement of transmission to timeout (in seconds) No timeout occurs when set to 0.
GetRetransmissionInterval **SetRetransmissionInterval**	long	R/W	1 to 65535	30 (30sec.)	Retransmission interval (in units of 100 ms)
GetTransmissionTimeout **SetTransmissionTimeout**	long	R/W	1 to 65535	30 (30sec.)	Time required for transmission timeout (in seconds)

(*1) Only for COM1

20.5.2. Methods

Function	Description
AddFile	Add a file to be transmitted.
ClearFile	Clear a file added by AddFile.
GetFileCount	Return the number of files transmitted including a file being transmitted.
Send	Transmit a file specified by AddFile.
Receive	Receive a file.
Abort	Abort the current file transmission process.
GetState	Get the current file transmission status.
GetError	Return the error information about the transaction processed last.

AddFile

Description

Add a file to be transmitted. Specify the filename excluding its pathname. The length of the filename is within 90 characters.

Syntax

long AddFile (
LPCTSTR *FileName* **)**

Parameters

FileName
[in] Filename excluding pathname

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_INVALID_PARAMETER	NULL set to the parameter. Filename length is 0.
ERROR_FILENAME_EXCED_RANGE	Filename too long

ClearFile

Description
Clears a file added by AddFile.

Syntax
```
void ClearFile ( void )
```

Parameters
None

Return value
None

GetFileCount

Description
Return the number of files transmitted including a file being transmitted.

Syntax
```
short GetFileCount ( void )
```

Parameters
None

Return value
Number of files transmitted (including a file being transmitted)

Send

Description

Transmit a file specified by AddFile.

Syntax

```
Long Send ( void )
```

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_ACCESS_DENIED	Access to COM port denied (e.g., occupied by other tasks)
ERROR_FILE_NOT_FOUND	Specified file or device not found
ERROR_NO_MORE_FILES	No send file found (No file added by **AddFile**.)
ERROR_BAD_PATHNAME	Path too long (Path + filename > 260 characters)

Receive

Description

Receive a file.

Syntax

```
long Receive (void)
```

Parameters

None

Return value

Error code	Meaning
ERROR_SUCCESS	Successful completion
ERROR_ACCESS_DENIED	Access to COM port denied (e.g., occupied by other tasks)
ERROR_FILE_NOT_FOUND	Specified file or device not found

Abort

Description

Abort the current file transmission process. After aborting, the *Done* event will occur.

Syntax

```
Void Abort ( void )
```

Parameters

None

Return value

None

GetState

Description

Get the current file transmission status.

Syntax

```
short GetState ( void )
```

Parameters

None

Return value

Error code	Meaning
TRANSFER_READY	On standby
TRANSFER_SEND	Transmitting
TRANSFER_RECEIVE	Receiving

GetError

Description

Return the error information for the transaction processed last.

Syntax

```
long GetError ( void )
```

Parameters

None

Return value

Code of an error that occurred during access to properties or processing of methods.

20.5.3. Event Callback Functions

Function	Description
Done	This function is called when the transmission ends as specified.
Transferring	Get the information about a file being transmitted.

Done

Description

This function is called when the transmission ends as specified.

Syntax

```
void OnDone (
long Result )
```

Parameters

Result

[out] End code listed in the table below

Result	Meaning
RROR_SUCCESS	Succeeded.
ERROR_TIMEOUT	Timeout.
ERROR_OPERATION_ABORTED	Process is aborted.
ERROR_OPEN_FAILED	Failed to open a file.
ERROR_INVALID_DATA	Invalid data received.
ERROR_DISK_FULL	Sufficient storage area not reserved.
ERROR_BAD_PATHNAME	Path too long (Path + filename > 260 characters)

Return value

None

Transferring

Description

Get the information about a file being transmitted.

Syntax

```
void OnTransferring (
LPCTSTR FileName ,
long Total ,
long Transferred )
```

Parameters

FileName
[out] Name of file being transmitted

Total
[out] Size of file being transmitted

Transferred
[out] Size of file already transmitted

Return value

None

20.5.4. Coding Sample

```
void CSerialTransferDlg::DoDataExchange(CDataExchange* pDX)
{
   CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CSerialTransferDlg)
   DDX_Control(pDX, IDC_FILETRANSFERCTRL1, m_clFileTransfer);
   //}}AFX_DATA_MAP
}

BEGIN_EVENTSINK_MAP(CSerialTransferDlg, CDialog)
   //{{AFX_EVENTSINK_MAP(CSerialTransferDlg)
   ON_EVENT(CSerialTransferDlg, IDC_FILETRANSFERCTRL1, 1 /* Done */, OnDoneFiletransferctrl, VTS_I4)
   ON_EVENT(CSerialTransferDlg, IDC_FILETRANSFERCTRL1, 2 /* Transferring */,
OnTransferringFiletransferctrl, VTS_BSTR VTS_I4 VTS_I4)
   //}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

/* Start download */
void CSerialTransferDlg::OnDownload()
{
   m_clFileTransfer.SetPath(TEXT("\\My Documents"));          // Set a filepath for the work file
   m_clFileTransfer.SetTransferringEventInterval(10);         // File transmission event (1s)
   m_clFileTransfer.Receive();                                // Start transmission
}

/* Start upload */
void CSerialTransferDlg::OnUpload()
{
   m_clFileTransfer.SetPath(TEXT("\\My Documents"));          // Set a filepath for the work file
   m_clFileTransfer.AddFiles(TEXT("File1.dat"));              // Transmission file 1
   m_clFileTransfer.AddFiles(TEXT("File2.dat"));              // Transmission file 2
   m_clFileTransfer.AddFiles(TEXT("File3.dat"));              // Transmission file 3
   m_clFileTransfer.SetTransferringEventInterval(10);         // File transmission event (1s)
   m_clFileTransfer.Send();                                   // Start transmission
}

/* Abort */
void CSerialTransferDlg::OnAbort()
{
   m_clFileTransfer.Abort();                                  // Abort
}

/* Send/receive complete */
void CSerialTransferDlg::OnDoneFiletransferctrl(long Result)
{
   CString clMsg;
   clMsg.Format(TEXT("Done:%d"), Result);
   AfxMessgeBox(clMsg, MB_ICONINFORMATION);
}

/* Display the info about file being transmitted */
void CSerialTransferDlg::OnTransferringFiletransferctrl(LPCTSTR FileName, long Total,
long Transferred)
{
   if(0 < Total)
   {
      TCHAR szProgress[MAX_PATH];
      wsprintf(szProgress, TEXT("%s %d%%"), FileName, (int)(Transferred*100/Total));
      SetWindowText(szProgress);                              // Display on the title bar
   }
}
```

Chapter 21. Error Codes

Error code table

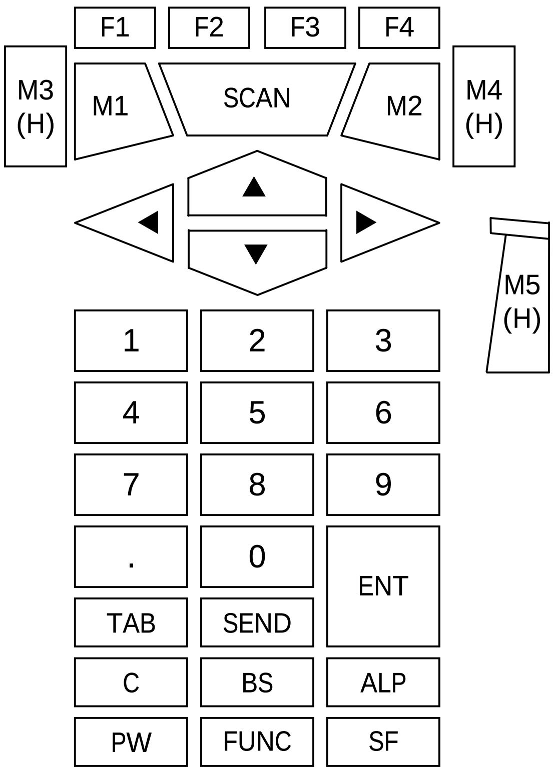
Error code	Content
ERROR_ACCESS_DENIED	Access to COM port denied. (e.g., occupied by other tasks)
ERROR_BAD_PATHNAME	Path too long. (Path + filename > 260 characters)
ERROR_DEV_NOT_EXIST	No NIC device found.
ERROR_DISK_FULL	Sufficient storage area not reserved.
ERROR_FILENAME_EXCED_RANGE	Filename too long.
ERROR_FILE_NOT_FOUND	Specified file or device not found.
ERROR_GEN_FAILURE	Not supported.
ERROR_INVALID_DATA	Invalid data received.
ERROR_INVALID_HANDLE	Barcode device file not opened.
ERROR_INVALID_PARAMETER	Parameter error. Address for storing data obtained not specified.
ERROR_NOT_READY	Attempt to open a device not ready.
ERROR_NOT_SUPPORTED	Invalid device.
ERROR_NO_MORE_FILES	No send file found. (No file added by **AddFile**.)
ERROR_OPEN_FAILED	Failed to open a file.
ERROR_OPERATION_ABORTED	Process is aborted.
ERROR_SUCCESS	Normal end.
ERROR_TIMEOUT	Timeout.
ERROR_TOO_MANY_OPEN_FILES	Barcode device file already opened.

Appendix A. Keyboard Arrangement, Virtual Key Codes and Character Codes

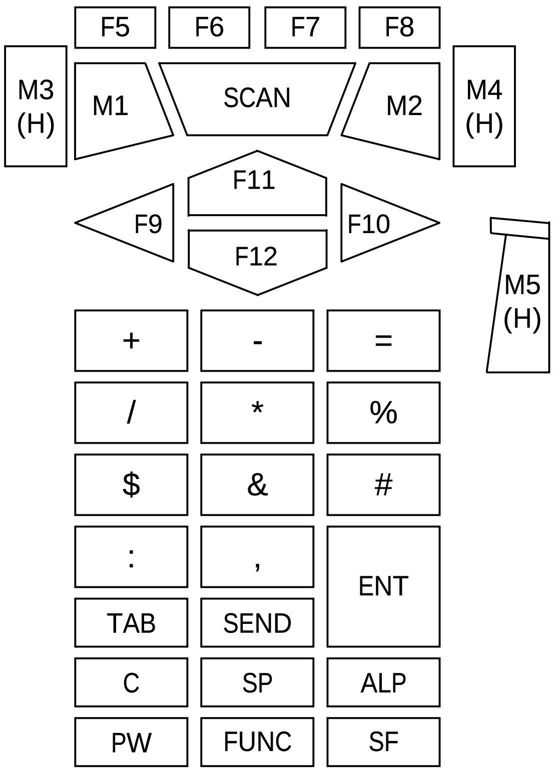
A.1. 31-key pad

A.1.1. Keyborard Arrangement

Numeric entry mode

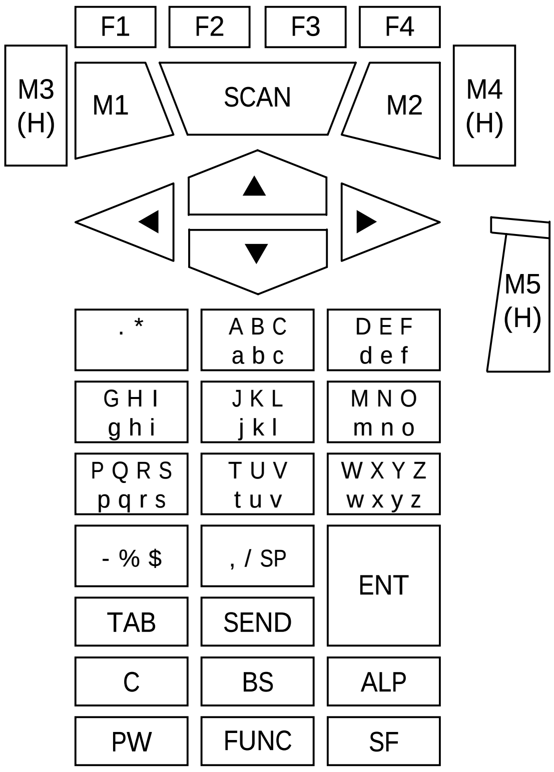


Normal status



Status with [SF] pressed

Alphabet entry mode



A.1.2. Virtual Key Codes and Character Codes

Numeric entry mode

Key	Normal status			Status with [SF] pressed		
	Virtual key code		Character code	Virtual key code		Character code
	Constant	Value		Constant	Value	
[F1]	VK_F1	70	-	VK_F5	74	-
[F2]	VK_F2	71	-	VK_F6	75	-
[F3]	VK_F3	72	-	VK_F7	76	-
[F4]	VK_F4	73	-	VK_F8	77	-
[◀]	VK_LEFT	25	-	VK_F9	78	-
[▶]	VK_RIGHT	27	-	VK_F10	79	-
[▲]	VK_UP	26	-	VK_F11	7A	-
[▼]	VK_DOWN	28	-	VK_F12	7B	-
[9]	VK_9	39	39(9)	VK_9	39	23(#)
[8]	VK_8	38	38(8)	VK_8	38	26(&)
[7]	VK_7	37	37(7)	VK_7	37	24($)
[6]	VK_6	36	36(6)	VK_6	36	25(%)
[5]	VK_5	35	35(5)	VK_5	35	2A(*)
[4]	VK_4	34	34(4)	VK_4	34	2F(/)
[3]	VK_3	33	33(3)	VK_3	33	3D(=)
[2]	VK_2	32	32(2)	VK_2	32	2D(-)
[1]	VK_1	31	31(1)	VK_1	31	2B(+)
[0]	VK_0	30	30(0)	VK_0	30	2C(,)
[.]	VK_PERIOD	BE	2E(.)	VK_PERIOD	BE	3A(:)
[ENT]	VK_RETURN	0D	0D	VK_RETURN	0D	0D
[TAB]	VK_TAB	09	09	VK_TAB	09	09
[SEND]	VK_SEND	D3	D3	VK_SEND	D3	D3
[C]	VK_CLEAR	0C	-	VK_CLEAR	0C	-
[BS]	VK_BACK	08	08	VK_SPACE	20	20( )
[ALP]	VK_ALP	D0	-	VK_ALP	D0	-
[Func]	VK_FUNC	D2	-	VK_FUNC	D2	-
[SF]	VK_SHIFT	10	-	VK_SHIFT	10	-
[SCAN]	VK_SCAN(*1)	D1(*1)	- (*1)	VK_SCAN(*1)	D1(*1)	-(*1)
[M1]	VK_M1(*1)	C1(*1)	- (*1)	VK_M1(*1)	C1(*1)	-(*1)
[M2]	VK_M2(*1)	C2(*1)	- (*1)	VK_M2(*1)	C2(*1)	-(*1)
[M3]	VK_M3(*1)	C3(*1)	- (*1)	VK_M3(*1)	C3(*1)	-(*1)
[M3H]	VK_M3H(*1)	C8(*1)	- (*1)	VK_M3H(*1)	C8(*1)	-(*1)
[M4]	VK_M4(*1)	C4(*1)	- (*1)	VK_M4(*1)	C4(*1)	-(*1)
[M4H]	VK_M4H(*1)	C9(*1)	- (*1)	VK_M4H(*1)	C9(*1)	-(*1)
[M5]	VK_M5(*1)	C5(*1)	- (*1)	VK_M5(*1)	C5(*1)	-(*1)
[M5H]	VK_M5H(*1)	CA(*1)	-(*1)	VK_M5H(*1)	CA(*1)	-(*1)

(*1) Virtual key codes and character codes will differ based on the key settings.
For details, refer to section “6.4 Magic Key Control”

A.1.3. Character Codes in Alphabet Entry Mode

In the alphabetic entry mode, the 0 to 9 and period (.) keys are used to enter alphabets. The table below lists the relationship between keys to be pressed, the number of depressions, and character codes.

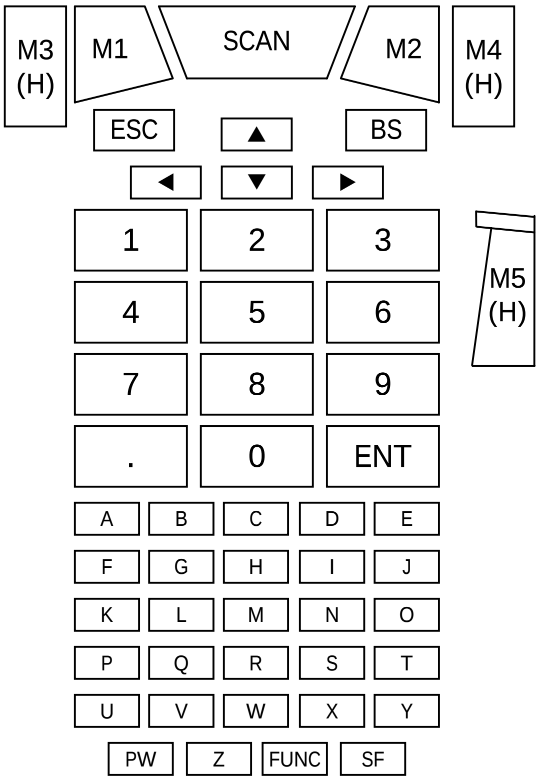
Depre-ssion / Key	1st	2nd	3rd	4 th	5 th	6 th	7 th	8th	9th
[0]	','	'/'	' ' (blank)	(*1)					
[1]	‘.’	‘*’	(*1)						
[2]	'A'	'B'	'C'	'a'	'b'	'c'	(*1)		
[3]	'D'	'E'	'F'	'd	'e	'f'	(*1)		
[4]	'G'	'H'	'I'	'g	'h	'i'	(*1)		
[5]	'J'	'K'	'L'	'j	'k	'l'	(*1)		
[6]	'M'	'N'	'O'	'm'	'n'	'o'	(*1)		
[7]	'P'	'Q'	'R'	‘S’	'p'	'q'	'r'	‘s’	(*1)
[8]	'T'	'U'	'V'	't'	'u'	'v'	(*1)		
[9]	'W'	'X'	'Y'	'Z'	'w'	'x'	'y'	'z'	(*1)
[.]	'-'	'%'	'$'	(*1)					

(*1): Returns to the 1st letter.

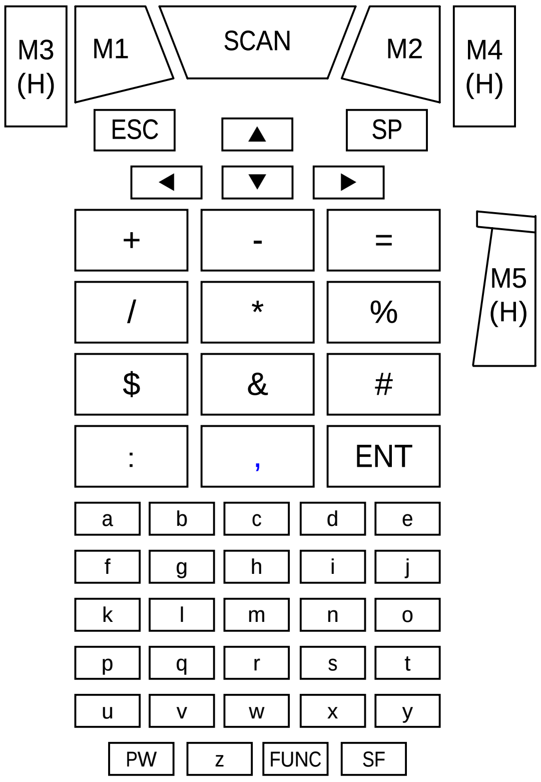
A.2. 50-key pad (Phone-type key layout)

A.2.1. Keyborard Arrangement

Non-function mode

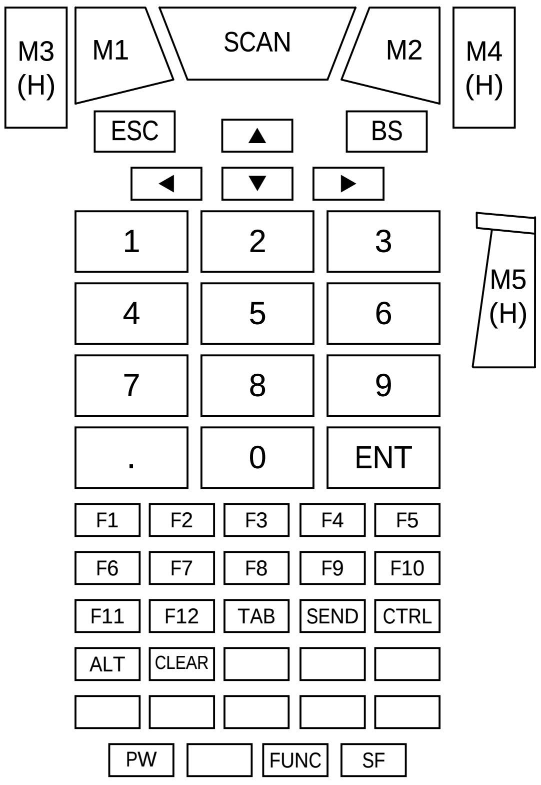


Normal status

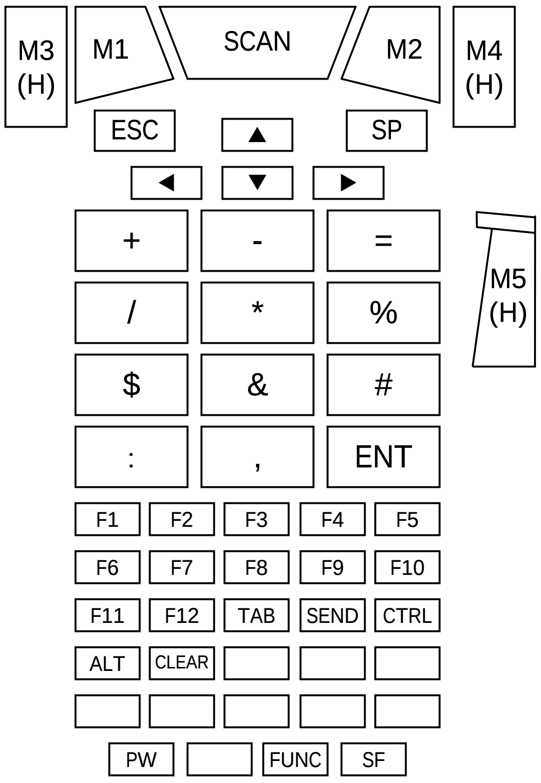


Status with [SF] pressed

Function mode



Normal status



Status with [SF] pressed

A.2.2. Virtual Key Codes and Character Codes

Non-function mode

Key	Normal status			Status with [SF] pressed		
	Virtual key code		Character code	Virtual key code		Character code
	Constant	Value		Constant	Value	
[◀]	VK_LEFT	25	-	VK_LEFT	25	-
[▶]	VK_RIGHT	27	-	VK_RIGHT	27	-
[▲]	VK_UP	26	-	VK_UP	26	-
[▼]	VK_DOWN	28	-	VK_DOWN	28	-
[9]	VK_9	39	39(9)	VK_9	39	3D(=)
[8]	VK_8	38	38(8)	VK_8	38	2D(-)
[7]	VK_7	37	37(7)	VK_7	37	2B(+)
[6]	VK_6	36	36(6)	VK_6	36	25(%)
[5]	VK_5	35	35(5)	VK_5	35	2A(*)
[4]	VK_4	34	34(4)	VK_4	34	2F(/)
[3]	VK_3	33	33(3)	VK_3	33	23(#)
[2]	VK_2	32	32(2)	VK_2	32	26(&)
[1]	VK_1	31	31(1)	VK_1	31	24($)
[0]	VK_0	30	30(0)	VK_0	30	3A(:)
[.]	VK_PERIOD	BE	2E(.)	VK_PERIOD	BE	2C(,)
[A]	VK_A	41	41(A)	VK_A	41	61(a)
[B]	VK_B	42	42(B)	VK_B	42	62(b)
[C]	VK_C	43	43(C)	VK_C	43	63(c)
[D]	VK_D	44	44(D)	VK_D	44	64(d)
[E]	VK_E	45	45(E)	VK_E	45	65(e)
[F]	VK_F	46	46(F)	VK_F	46	66(f)
[G]	VK_G	47	47(G)	VK_G	47	67(g)
[H]	VK_H	48	48(H)	VK_H	48	68(h)
[I]	VK_I	49	49(I)	VK_I	49	69(i)
[J]	VK_J	4A	4A(J)	VK_J	4A	6A(j)
[K]	VK_K	4B	4B(K)	VK_K	4B	6B(k)
[L]	VK_L	4C	4C(L)	VK_L	4C	6C(l)
[M]	VK_M	4D	4D(M)	VK_M	4D	6D(m)
[N]	VK_N	4E	4E(N)	VK_N	4E	6E(n)
[O]	VK_O	4F	4F(O)	VK_O	4F	6F(o)
[P]	VK_P	50	50(P)	VK_P	50	70(p)
[Q]	VK_Q	51	51(Q)	VK_Q	51	71(q)
[R]	VK_R	52	52(R)	VK_R	52	72(r)
[S]	VK_S	53	53(S)	VK_S	53	73(s)
[T]	VK_T	54	54(T)	VK_T	54	74(t)
[U]	VK_U	55	55(U)	VK_U	55	75(u)
[V]	VK_V	56	56(V)	VK_V	56	76(v)
[W]	VK_W	57	57(W)	VK_W	57	77(w)
[X]	VK_X	58	58(X)	VK_X	58	78(x)
[Y]	VK_Y	59	59(Y)	VK_Y	59	79(y)
[Z]	VK_Z	5A	5A(Z)	VK_Z	5A	7A(z)
[ENT]	VK_RETURN	0D	0D	VK_RETURN	0D	0D
[ESC]	VK_ESCAPE	1B	1B	VK_ESCAPE	1B	1B
[BS]	VK_BACK	08	08	VK_SPACE	20	20( )
[FUNC]	VK_FUNC	D2	-	VK_FUNC	D2	-
[SF]	VK_SHIFT	10	-	VK_SHIFT	10	-
[SCAN]	VK_SCAN	D1(*1)	-(*1)	VK_SCAN	D1(*1)	-(*1)
[M1]	VK_M1	C1(*1)	-(*1)	VK_M1	C1(*1)	-(*1)
[M2]	VK_M2	C2(*1)	-(*1)	VK_M2	C2(*1)	-(*1)
[M3H]	VK_M3H	C8(*1)	-(*1)	VK_M3H	C8(*1)	-(*1)
[M3]	VK_M3	C3(*1)	-(*1)	VK_M3	C3(*1)	-(*1)
[M4H]	VK_M4H	C9(*1)	-(*1)	VK_M4H	C9(*1)	-(*1)
[M4]	VK_M4	C4(*1)	-(*1)	VK_M4	C4(*1)	-(*1)
[M5H]	VK_M5H	CA(*1)	-(*1)	VK_M5H	CA(*1)	-(*1)
[M5]	VK_M5	C5(*1)	-(*1)	VK_M5	C5(*1)	-(*1)

(*1) : Virtual key codes and character codes will differ based on the key settings.

Function mode

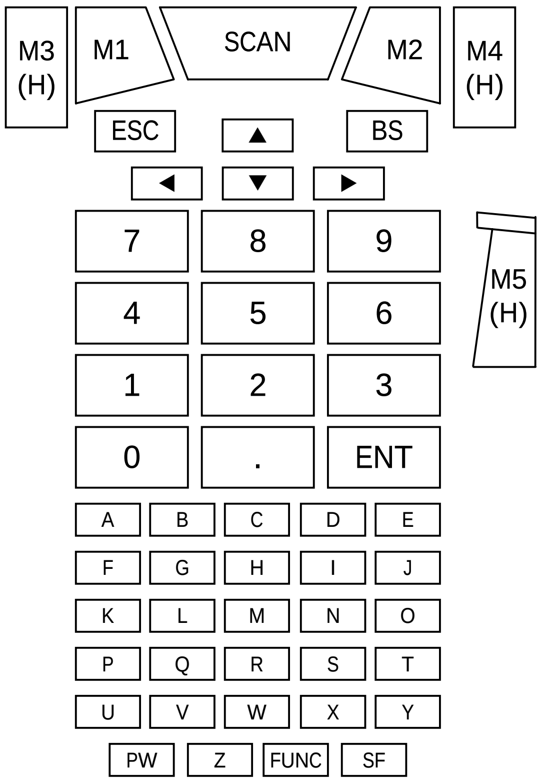
Key	Normal status			Status with [SF] pressed		
	Virtual key code		Character code	Virtual key code		Character code
	Constant	Value		Constant	Value	
[◀]	VK_LEFT	25	-	VK_LEFT	25	-
[▶]	VK_RIGHT	27	-	VK_RIGHT	27	-
[▲]	VK_UP	26	-	VK_UP	26	-
[▼]	VK_DOWN	28	-	VK_DOWN	28	-
[9]	VK_9	39	39(9)	VK_9	39	3D(=)
[8]	VK_8	38	38(8)	VK_8	38	2D(-)
[7]	VK_7	37	37(7)	VK_7	37	2B(+)
[6]	VK_6	36	36(6)	VK_6	36	25(%)
[5]	VK_5	35	35(5)	VK_5	35	2A(*)
[4]	VK_4	34	34(4)	VK_4	34	2F(/)
[3]	VK_3	33	33(3)	VK_3	33	23(#)
[2]	VK_2	32	32(2)	VK_2	32	26(&)
[1]	VK_1	31	31(1)	VK_1	31	24($)
[0]	VK_0	30	30(0)	VK_0	30	3A(:)
[.]	VK_PERIOD	BE	2E(.)	VK_PERIOD	BE	2C(,)
[A]	VK_F1	70	-	VK_F1	70	-
[B]	VK_F2	71	-	VK_F2	71	-
[C]	VK_F3	72	-	VK_F3	72	-
[D]	VK_F4	73	-	VK_F4	73	-
[E]	VK_F5	74	-	VK_F5	74	-
[F]	VK_F6	75	-	VK_F6	75	-
[G]	VK_F7	76	-	VK_F7	76	-
[H]	VK_F8	77	-	VK_F8	77	-
[I]	VK_F9	78	-	VK_F9	78	-
[J]	VK_F10	79	-	VK_F10	79	-
[K]	VK_F11	7A	-	VK_F11	7A	-
[L]	VK_F12	7B	-	VK_F12	7B	-
[M]	VK_TAB	09	09	VK_TAB	09	09
[N]	VK_SEND	D3	D3	VK_SEND	D3	D3
[O]	VK_CONTROL	11	11	VK_CONTROL	11	11
[P]	VK_MENU	12	12	VK_MENU	12	12
[Q]	VK_CLEAR	0C	0C	VK_CLEAR	0C	0C
[R]	-	-	-	-	-	-
[S]	-	-	-	-	-	-
[T]	-	-	-	-	-	-
[U]	-	-	-	-	-	-
[V]	-	-	-	-	-	-
[W]	-	-	-	-	-	-
[X]	-	-	-	-	-	-
[Y]	-	-	-	-	-	-
[Z]	-	-	-	-	-	-
[ENT]	VK_RETURN	0D	0D	VK_RETURN	0D	0D
[ESC]	VK_ESCAPE	1B	1B	VK_ESCAPE	1B	1B
[BS]	VK_BACK	08	08	VK_SPACE	20	20( )
[FUNC]	VK_FUNC	D2	-	VK_FUNC	D2	-
[SF]	VK_SHIFT	10	-	VK_SHIFT	10	-
[SCAN]	VK_SCAN	D1(*1)	-(*1)	VK_SCAN	D1(*1)	-(*1)
[M1]	VK_M1	C1(*1)	-(*1)	VK_M1	C1(*1)	-(*1)
[M2]	VK_M2	C2(*1)	-(*1)	VK_M2	C2(*1)	-(*1)
[M3H]	VK_M3H	C8(*1)	-(*1)	VK_M3H	C8(*1)	-(*1)
[M3]	VK_M3	C3(*1)	-(*1)	VK_M3	C3(*1)	-(*1)
[M4H]	VK_M4H	C9(*1)	-(*1)	VK_M4H	C9(*1)	-(*1)
[M4]	VK_M4	C4(*1)	-(*1)	VK_M4	C4(*1)	-(*1)
[M5H]	VK_M5H	CA(*1)	-(*1)	VK_M5H	CA(*1)	-(*1)
[M5]	VK_M5	C5(*1)	-(*1)	VK_M5	C5(*1)	-(*1)

(*1) : Virtual key codes and character codes will differ based on the key settings.

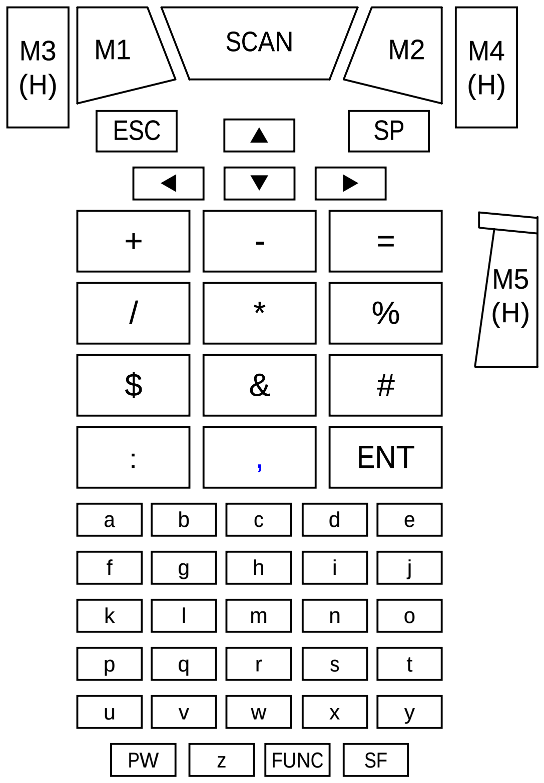
A.3. 50-key pad (Calculator-type key layout)

A.3.1. Keyborard Arrangement

Non-function mode

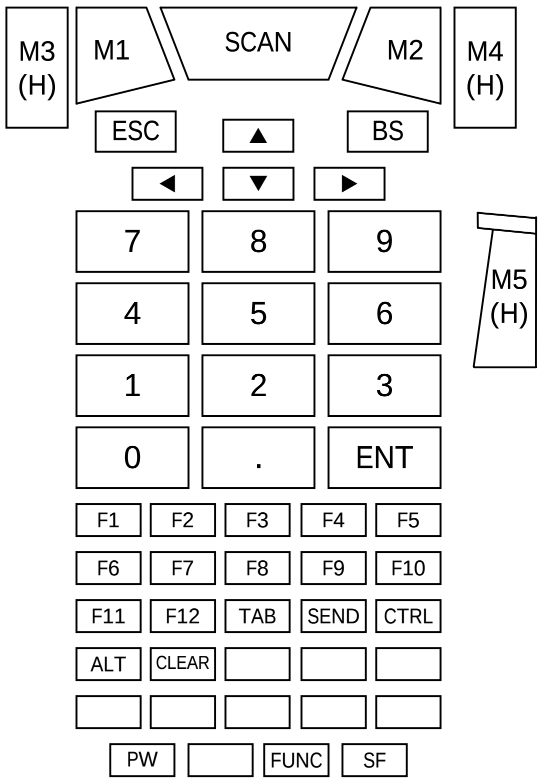


Normal status

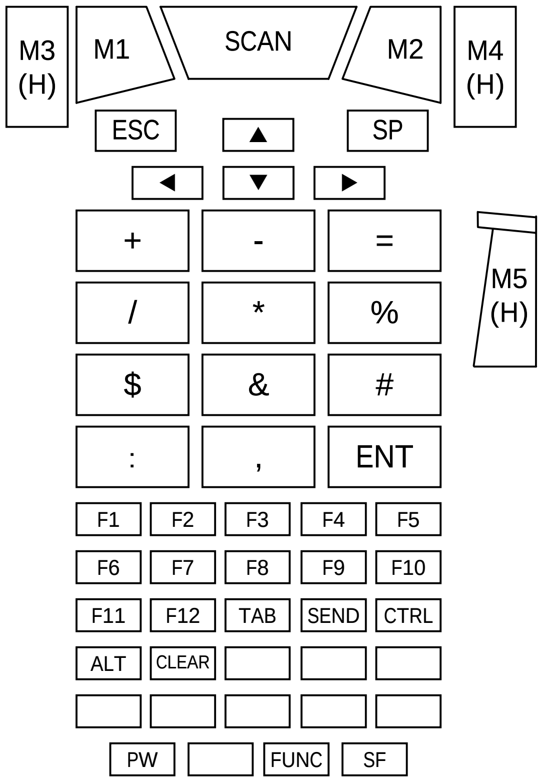


Status with [SF] pressed

Function mode



Normal status



Status with [SF] pressed

A.3.2. Virtual Key Codes and Character Codes

Same as "A.2.2. Virtual Key Codes and Character Codes".

BHT-400-CE

API Reference Manual

Second Edition, September 2006

DENSO WAVE INCORPORATED