

Bluetooth Library Manual

(Version 1.05)

CASIO Computer Co., Ltd.

Copyright ©2011. All rights reserved.

January 2011

Table of the Contents

	Editorial Record	4
Chapter 1.	Overview	5
Chapter 2.	Operation Environment	6
Chapter 3.	Structures	7
3.1	BTST_LOCALINFO	7
3.2	BTST_DEVICEINFO	8
Chapter 4.	Constants	9
4.1	Device Mode	9
4.2	Device Class	10
4.3	Service UUID	13
4.4	Error Flag	15
Chapter 5.	Functions List	18
5.1	BTInitialize	20
5.2	BTDeInitialize	21
5.3	BTGetLocalInfo	22
5.4	BTSetLocalInfo	23
5.5	BTInquiry	24
5.6	BTGetDeviceInfo	26
5.7	BTGetServiceInfo	28
5.8	BTSelectDevice	29
5.9	BTSetPassKey	31
5.10	BTTrustDevice	32
5.11	BTSetWakeOnStatus	33
5.12	BTGetWakeOnStatus	34
5.13	BTGetDeviceHandle	35
5.14	BTGetLastError	36
5.15	BTRegisterLocalInfo	37
5.16	BTRegisterDeviceInfo	38
5.17	BTSearchDeviceInfo	39
5.18	BTDeleteDeviceInfo	41
5.19	BTGetDefaultDeviceInfo	42
5.20	BTSetDefaultDevice	43
5.21	BTGetLibraryStatus	45
5.22	BTGetDeviceName	46
5.23	BTGetConnectionStatus	47
5.24	BTSetConnectionParameter	48
5.25	BTGetConnectionParameter	49
5.26	BTSetAFHStatus	50
5.27	BTGetAFHStatus	52
5.28	BTWaitForBtReady	53
5.29	BTConnectSerial	54
5.30	BTSendSerialData	56
5.31	BTReceiveSerialData	57
5.32	BTDisconnectSerial	59
5.33	BTSetPANStatus	60
5.34	BTGetPANStatus	61
5.35	BTConnectPAN	62

5.36	BTDisconnectPAN	63
5.37	BTConnectHeadset	64
5.38	BTDisconnectHeadset	65
5.39	BTSetSoundPath	66
5.40	BTGetSoundPath	67
5.41	BTSetHeadsetGain	68
5.42	BTGetHeadsetGain	69
5.43	BTSetHeadsetServerStatus	70
5.44	BTGetHeadsetServerStatus	71
5.45	BTSetOBEXFolder	72
5.46	BTGetOBEXFolder	73
5.47	BTSendOBEXFile	74
Chapter 6.	Connection Procedure by Profile	75
6.1	Registering Partner Bluetooth Equipment	75
6.2	Connections via Serial Profile	76
6.3	Connections via Dial-Up Profile	78
6.4	Connections via LAN Profile	79
Chapter 7.	Notes to Programming	80
7.1	Communication Profiles	80
7.2	Bluetooth Communication Modes	81
Chapter 8.	Device Emulator	82
8.1	BTInit.ini	82
8.2	BTDeviceInfo[n].ini	82
8.3	BTReg.ini	83

No part of this document may be produced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of CASIO Computer Co., Ltd. in Tokyo Japan. Information in this document is subject to change without advance notice. CASIO Computer Co., Ltd. makes no representations or warranties with respect to the contents or use of this manual and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose.

© 2011 CASIO Computer Co., Ltd. All rights reserved.

Editorial Record

[illegible]

1. Overview

The **Bluetooth Library** makes use of the Bluetooth module integrated in the terminal to provide connection and communication functions with other Bluetooth equipment.

The **Bluetooth Class Library** is a wrapper library layer. The library can be directly manipulated by .NET Compact Framework application.

The use of **Bluetooth Library** enables the enhancement of source code compatibility for mobile applications regardless of handheld terminal model.

With regard to the function for either the “unsupported error” or “parameter error”, at time of process execution, the user is notified that the function is unsupported or that the process itself should be disabled. In this way it is possible to describe a source code that is aware of the “functions” rather than a source code that is aware of the handheld terminal “models” (and the devices integrated in those handheld terminal models), which enables the development of a business application style that attaches “importance to functions”.

Note:

The aim of this library is to enhance the source code compatibility for mobile business application, it is not intended to guarantee the compatibility of functions of integrated devices. In actuality it decides whether the “unsupported error” or “parameter error” is required to notify the user that functions are not supported or that the actual process should be disabled.

Also, for the business application to actually run in multiple handheld terminal models, it must be rebuilt to agree with the CPU types (ARMV4T, ARMV4 ...).

2. Operation Environment

Applicable Handheld Terminals

- IT-600
- DT-X11
- DT-X7
- DT-X30
- IT-3100
- IT-800
- IT-300
- DT-X8

OS

- Microsoft® WindowsCE 5.0
- Microsoft® WindowsCE 6.0
- Microsoft® Windows Mobile 6.1
- Microsoft® Windows Mobile 6.5
- Microsoft® Windows Mobile 6.5.3

Development Environment

- Microsoft® eMbedded C++ Version 4.0 + Service Pack 4
- Microsoft® Visual Studio 2005 + Service Pack 1
- Microsoft® Visual Studio 2008 + Service Pack 1

Supplied Files

- BluetoothLib.h
- BluetoothLib.lib
- BluetoothLib.dll
- BluetoothLibNet.dll (Class Library)

Steps to Start Up

For Visual C++:

1. Include the header file, **BluetoothLib.h**, in the source program and define it as the library that uses the **BluetoothLib.lib** file.
2. **BluetoothLib.dll** is already integrated by default in the terminal.

For Visual Basic and Visual C#:

1. Add **BluetoothLibNet.dll** to the Project Reference.
2. **BluetoothLib.dll** is already integrated by default in the terminal.
3. Copy **BluetoothLibNet.dll** into the same folder where the execution module is stored.

3. Structures

The following shows the structure systems offered by the **Bluetooth Library**.

Table 3.1 Structure

Structure	Description	Function to use
BTST_LOCALINFO	Structure that saves the Bluetooth device information of the terminal.	BTGetLocalInfo BTSetLocalInfo
BTST_DEVICEINFO	Structure that saves the communication partner's Bluetooth device information.	BTGetDeviceInfo BTGetServiceInfo BTSelectDevice BTTrustDevice BTRegisterDeviceInfo BTSearchDeviceInfo BTDeleteDeviceInfo BTGetDefaultDeviceInfo BTSetDefaultDevice

3.1 BTST_LOCALINFO

This structure saves the terminal's Bluetooth device information.

```
struct _btlocalinfo {
    TCHAR    LocalName[82];    : Bluetooth equipment name
    TCHAR    LocalAddress[18]; : Bluetooth equipment address
    LONG     LocalDeviceMode;  : Bluetooth device mode
    LONG     LocalClass1       : Bluetooth device class 1 (local)
    LONG     LocalClass2       : Bluetooth device class 2 (local)
    LONG     LocalClass3       : Bluetooth device class 3 (local)
    BOOL     Authentication;    : Bluetooth authentication setting flag
    BOOL     Encryption;        : Cipher setting flag
} BTST_LOCALINFO;
```

The following functions use the BTST_LOCALINFO structure.

BTGetLocalInfo

BTSetLocalInfo

3.2 BTST_DEVICEINFO

This structure saves the communication partner's Bluetooth device information.

```
struct _btdeviceinfo {
    LONG    DeviceErrorFlag;      : Error flag
    HANDLE  DeviceHandle;        : Device handle
    TCHAR   DeviceName[82];      : Bluetooth equipment name
    TCHAR   DeviceAddress[18];   : Bluetooth equipment address
    LONG    DeviceClass1;        : Bluetooth device class 1(Destination)
    LONG    DeviceClass2;        : Bluetooth device class 2(Destination)
    LONG    DeviceClass 3;      : Bluetooth device class 3(Destination)
    DWORD   ProfileNumber;       : Usable profile number
    WORD    ProfileUUID[16];     : Usable profile type
} BTST_DEVICEINFO;
```

The following functions use the BTST_DEVICEINFO structure.

- BTGetDeviceInfo**
- BTGetServiceInfo**
- BTSelectDevice**
- BTTrustDevice**
- BTRegisterDeviceInfo**
- BTSearchDeviceInfo**
- BTDeleteDeviceInfo**
- BTGetDefaultDeviceInfo**
- BTSetDefaultDevice**

4. Constants

The following show the constants provided by the **Bluetooth Library**.

Table 4.1 Constants

Constant	Description
Device Mode	Access permission for other Bluetooth devices
Device Class	Bluetooth equipment attributes
Service UUID	Services usable with Bluetooth device
Error Flag	Error detailed information

4.1 Device Mode

The device mode is a set of parameters for deciding whether or not the integrated Bluetooth device in the terminal will give access permission to other Bluetooth devices. The device mode retrieves and sets parameters listed in the table below using the LocalDeviceMode member of BTST_LOCALINFO structure. The default setting is BTMODE_BOTH_ENABLED.

Table 4.2 Device mode

Setting Value	Description	DT-X11 IT-600 DT-X7 DT-X30 IT-3100 IT-800 IT-300 DT-X8
BTMODE_NO_SCANS	Inquiry and connection with other Bluetooth devices are not permitted.	Y
BTMODE_INQUIRY_ENABLED	Only inquiries from other Bluetooth devices are permitted.	Y
BTMODE_PAGE_ENABLED	Only connections with other Bluetooth devices are permitted.	Y
BTMODE_BOTH_ENABLED	Inquiry and connection with other Bluetooth devices are permitted.	Y
BTMODE_LIMITED_ACCESSIBLE	Only connections with specified parties are permitted.	--

4.2 Device Class

The device class is parameter showing what attributes Bluetooth equipment has. There are three kinds of device class parameters; major service class, major device class and minor device class. Set each class to the respective members of BTST_LOCALINFO structure.

- LocalClass1 ← Set the major device class.
- LocalClass2 ← Set the minor device class.
- LocalClass3 ← Set the major service class.

Major Device Class

The major device class is parameter that shows the attributes of Bluetooth device. Just one type of the parameters listed below can be used.

- BTCOD_MAJOR_MISC
- BTCOD_MAJOR_COMPUTER
- BTCOD_MAJOR_PHONE
- BTCOD_MAJOR_LAN_ACCESS_POINT
- BTCOD_MAJOR_AUDIO
- BTCOD_MAJOR_PERIPHERAL
- BTCOD_MAJOR_IMAGING
- BTCOD_MAJOR_UNCLASSIFIED

Minor Device Class

The minor device class is parameter that shows the attributes of Bluetooth device. The usable minor device classes are determined for each major device class.

The following parameter combines for use with all the major device classes except BTCOD_MAJOR_LAN_ACCESS_POINT.

- BTCOD_MINOR_UNCLASSIFIED

The following parameters combine for use with BTCOD_MAJOR_COMPUTER device class.

- BTCOD_COMPUTER_DESKTOP
- BTCOD_COMPUTER_SERVER
- BTCOD_COMPUTER_LAPTOP
- BTCOD_COMPUTER_HANDHELD
- BTCOD_COMPUTER_PALM
- BTCOD_COMPUTER_WEARABLE

The following parameters combine for use with BTCOD_MAJOR_PHONE device class.

- BTCOD_PHONE_CELLULAR
- BTCOD_PHONE_CORDLESS
- BTCOD_PHONE_SMART
- BTCOD_PHONE_MODEM

The following parameters combine for use with BTCOD_MAJOR_LAN_ACCESS_POINT device class.

- BTCOD_LAP_FULLY_AVAILABLE
- BTCOD_LAP_USAGE_1
- BTCOD_LAP_USAGE_2
- BTCOD_LAP_USAGE_3
- BTCOD_LAP_USAGE_4
- BTCOD_LAP_USAGE_5
- BTCOD_LAP_USAGE_6
- BTCOD_LAP_NOT_AVAILABLE

The following parameters combine for use with BTCOD_MAJOR_AUDIO device class.

- BTCOD_AUDIO_HEADSET_PROFILE
- BTCOD_AUDIO_HANDS_FREE
- BTCOD_AUDIO_RESERVED1
- BTCOD_AUDIO_MICROPHONE
- BTCOD_AUDIO_LOUDSPEAKER
- BTCOD_AUDIO_HEADPHONES
- BTCOD_AUDIO_PORTABLE_AUDIO
- BTCOD_AUDIO_CAR_AUDIO
- BTCOD_AUDIO_SET_TOP_BOX
- BTCOD_AUDIO_HI_FI_DEVICE
- BTCOD_AUDIO_VCR
- BTCOD_AUDIO_VIDEO_CAMERA
- BTCOD_AUDIO_CAMCORDER
- BTCOD_AUDIO_VIDEO_MONITOR
- BTCOD_AUDIO_DISPLAY_LOUDSPEAKER
- BTCOD_AUDIO_VIDEO_CONFERENCING
- BTCOD_AUDIO_RESERVED
- BTCOD_AUDIO_GAMING_TOY

The following parameters combine for use with BTCOD_MAJOR_PERIPHERAL device class.

- BTCOD_PP_KEYBOARD
- BTCOD_PP_POINTING_DEVICE
- BTCOD_PP_COMBO_DEVICE

The following parameters combine for use with the minor device class of BTCOD_MAJOR_PERIPHERAL device class.

- BTCOD_PP_UNCATEGORIZED
- BTCOD_PP_JOYSTICK
- BTCOD_PP_GAMEPAD
- BTCOD_PP_REMOTE_CONTROL
- BTCOD_PP_SENSING_DEVICE

The following parameters combine for use with BTCOD_MAJOR_IMAGING device class.

BTCOD_IMAGING_RESERVED1
BTCOD_IMAGING_RESERVED2
BTCOD_IMAGING_DISPLAY
BTCOD_IMAGING_CAMERA
BTCOD_IMAGING_SCANNER
BTCOD_IMAGING_PRINTER

Major Service Class

The major service class is parameter that shows the service attributes of Bluetooth device. The following parameters are usable. By calculating a sum of the values in logical OR for each parameter, multiple major class service settings can be made.

BTCOD_LIMITED_DISCOVERABLE
BTCOD_RESERVED_1
BTCOD_RESERVED_2
BTCOD_POSITIONING
BTCOD_NETWORKING
BTCOD_RENDERING
BTCOD_CAPTURING
BTCOD_OBJECT_TRANSFER
BTCOD_AUDIO
BTCOD_TELEPHONY
BTCOD_INFORMATION

Retrieving and Setting Device Class Parameters

In order to retrieve the terminal's parameters of the device class, first carry out **BTGetLocalInfo** function, and then refer to each member, LocalClass1, LocalClass2, and LocalClass3, of BTST_LOCALINFO structure.

To retrieve other Bluetooth device's parameters of the device class, first carry out **BTGetDeviceInfo** function, and then refer to each member, DeviceClass1, DeviceClass2, and DeviceClass3, BTST_DEVICEINFO structure.

To set the device class for the terminal, first set the device class parameters for reach local class member of BTST_LOCALINFO structure, and then carry out **BTSetLocalInfo** function.

4.3 Service UUID

Service UUID is parameters used for retrieving services available on the communication partner Bluetooth device. Service class value is saved in the ProfileUUID member of BTST_DEVICEINFO structure when **BTGetServiceInfo** function is carried out. The following parameters can be used.

Table 4.3 Service UUID

Service UUID	DT-X11 IT-600 DT-X7 DT-X30 IT-3100 IT-800 IT-300 DT-X8
BTUUID_SDP_PROTOCOL	Y
BTUUID_RFCOMM_PROTOCOL	Y
BTUUID_TCS_PROTOCOL	Y
BTUUID_CTP_PROTOCOL	Y
BTUUID_L2CAP_PROTOCOL	Y
BTUUID_IP_PROTOCOL	Y
BTUUID_UDP_PROTOCOL	Y
BTUUID_TCP_PROTOCOL	Y
BTUUID_TCS_AT_PROTOCOL	Y
BTUUID_OBEX_PROTOCOL	Y
BTUUID_FTP_PROTOCOL	Y
BTUUID_HTTP_PROTOCOL	Y
BTUUID_WSP_PROTOCOL	Y
BTUUID_BNEP_PROTOCOL	Y
BTUUID_UPNP_PROTOCOL	Y
BTUUID_HIDP_PROTOCOL	Y
BTUUID_SERVICE_DISCOVERY_SERVER	Y
BTUUID_BROWSE_GROUP_DESCRIPTOR	Y
BTUUID_PUBLIC_BROWSE_ROOT	Y
BTUUID_PUBLIC_BROWSE_GROUP	Y
BTUUID_SERIAL_PORT	Y
BTUUID_LAN_ACCESS_USING_PPP	Y
BTUUID_DIALUP_NETWORKING	Y
BTUUID_IR_MC_SYNC	Y
BTUUID_OBEX_OBJECT_PUSH	Y
BTUUID_OBEX_FILE_TRANSFER	Y
BTUUID_IR_MC_SYNC_COMMAND	Y
BTUUID_HEADSET	Y
BTUUID_CORDLESS_TELEPHONY	Y
BTUUID_INTERCOM	Y
BTUUID_FAX	Y
BTUUID_HEADSET_AUDIO_GATEWAY	Y
BTUUID_WAP	Y

Continue.

Service UUID	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
BTUUID_WAP_CLIENT				Y				
BTUUID_PANU				Y				
BTUUID_NAP				Y				
BTUUID_GN				Y				
BTUUID_DIRECT_PRINTING				Y				
BTUUID_REFERENCE_PRINTING				Y				
BTUUID_IMAGING				Y				
BTUUID_IMAGING_RESPONDER				Y				
BTUUID_IMAGING_AUTOMATIC_ARCHIVE				Y				
BTUUID_IMAGING_REFERENCE_OBJECTS				Y				
BTUUID_HANDSFREE				Y				
BTUUID_HANDSFREE_AUDIO_GATEWAY				Y				
BTUUID_DIRECT_PRINTING_REFERENCE_OBJECTS				Y				
BTUUID_REFLECTED_UI				Y				
BTUUID_BASIC_PRINTING				Y				
BTUUID_PRINTING_STATUS				Y				
BTUUID_HUMAN_INTERFACE_DEVICE_SERVICE				Y				
BTUUID_HARDCOPY_CABLE_REPLACEMENT				Y				
BTUUID_CHR_PRINT				Y				
BTUUID_HCR_SCAN				Y				
BTUUID_COMMON_ISDN_ACCESS				Y				
BTUUID_VIDEO_CONFERENCING_GW				Y				
BTUUID_PNP_INFORMATION				Y				
BTUUID_GENERIC_NETWORKING				Y				
BTUUID_GENERIC_FILE_TRANSFER				Y				
BTUUID_GENERIC_AUDIO				Y				
BTUUID_GENERIC_TELEPHONY				Y				
BTUUID_CTP				Y				

4.4 Error Flag

The error flag consists of error category flag and error status flag returned by function.

Error Category Flag

The error category flags express the category of an error that has occurred.

BTERR_CAT_NO_CATEGORY
BTERR_CAT_UART
BTERR_CAT_OSIF
BTERR_CAT_L2HCI
BTERR_CAT_RFCOMM
BTERR_CAT_SDK
BTERR_CAT_LYM
BTERR_CAT_IPC_RPC
BTERR_CAT_OBEX
BTERR_CAT_BLUETOOTH
BTERR_CAT_WSADAPTER
BTERR_CAT_WINSOCK_2X
BTERR_CAT_BNEP
BTERR_CAT_WINDOWS_SYSTEM
BTERR_CAT_WINDOWS_REGISTRY

Error Status Flag

The error status flags express the status of an error.

BTERR_SUCCESS
BTERR_INVALID_PARAMETER_1
BTERR_INVALID_PARAMETER_2
BTERR_INVALID_PARAMETER_3
BTERR_INVALID_PARAMETER_4
BTERR_INVALID_PARAMETER_5
BTERR_INVALID_PARAMETER_6
BTERR_INVALID_PARAMETER_7
BTERR_INVALID_PARAMETER_8
BTERR_INVALID_PARAMETER_9
BTERR_INVALID_PARAMETER_10
BTERR_INVALID_PARAMETER_11_OR_MORE
BTERR_FAILED
BTERR_PENDING
BTERR_NO_MEMORY
BTERR_INVALID_PARAMETER
BTERR_OPERATION_FAILED
BTERR_INVALID_HANDLE
BTERR_CONNECTION_CLOSED
BTERR_BUFFER_TOO_SMALL
BTERR_END_OF_LIST
BTERR_ALREADY_EXISTS
BTERR_NOT_FOUND
BTERR_OVERFLOW
BTERR_TIMEOUT
BTERR_NOT_IMPLEMENTED
BTERR_NO_RESOURCES
BTERR_INVALID_CONNECTION
BTERR_UNINITIALIZED
BTERR_UNLOADING
BTERR_NO_SERVER
BTERR_INVALID_STATE
BTERR_HW_ERROR
BTERR_DOES_NOT_EXIST
BTERR_CONNECTION_FAILED
BTERR_CONNECTION_LOST
BTERR_EARLY_RETURN
BTERR_CANCELLED_BY_USER
BTERR_UNAUTHORIZED
BTERR_INVALID_CHANNEL
BTERR_CONFLICT
BTERR_COULD_NOT_WRITE_TO_FILE
BTERR_SHARE_DOES_NOT_EXIST
BTERR_SCATTERNET
BTERR_PACKET_DROPPED
BTERR_MALFORMED_PACKET

BTERR_REDUNDANT
BTERR_COULD_NOT_OPEN_FILE
BTERR_TCPIP_NOT_AVAILABLE
BTERR_INVALID_CRITICAL_SECTION
BTERR_BIND
BTERR_OPENING_SOCKET
BTERR_MAXIMUM_RECURSION
BTERR_NO_MATCH
BTERR_PROTOCOL_UNAVAILABLE
BTERR_VERSION
BTERR_VALUE_NOT_FOUND
BTERR_SET_STRING_VALUE
BTERR_SET_UINT_VALUE
BTERR_SET_BIN_VALUE
BTERR_UNK_VALUE_TYPE
BTERR_MALFORMED_ADDRESS
BTERR_INVALID_PORT
BTERR_INVALID_UUID
BTERR_SERVICE_DOES_NOT_EXIST
BTERR_OBJECT_TYPE_INVALID
BTERR_DEFAULT_OBJECT_NOT_SET
BTERR_MALFORMED_PROPERTY
BTERR_COULD_NOT_READ_FILE
BTERR_FILE_NOT_FOUND
BTERR_DIRECTORY_NOT_FOUND
BTERR_CONNECTED
BTERR_MALFORMED_PRINTABLE_STRING
BTERR_MAX_FILESIZE_REACH
BTERR_LIB_INIT
BTERR_APP_EXIST
BTERR_DEVICE_LIST
BTERR_DEVICE_ADDRESS
BTERR_POWER_MODULE
BTERR_LIB_REINIT
BTERR_REG_OPEN
BTERR_REG_WRITE
BTERR_REG_READ
BTERR_REG_DELETE
BTERR_REG_NO_DATA
BTERR_REG_NOT_FOUND

5. Functions List

Table 5.1 Bluetooth Library functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
BTInitialize	Initializes Bluetooth protocol stack.	Y	Y	Y	Y	Y	Y	Y	Y
BTDeInitialize	Closes the resource for Bluetooth protocol stack.	Y	Y	Y	Y	Y	Y	Y	Y
BTGetLocalInfo	Retrieves Bluetooth device information.	Y	Y	Y	Y	Y	Y	Y	Y
BTSetLocalInfo	Sets up Bluetooth device information.	Y	Y	Y	Y	Y	Y	Y	Y
BTInquiry	Carries out inquiry for Bluetooth device.	Y	Y	Y	Y	Y	Y	Y	Y
BTGetDeviceInfo	Retrieves device information about Bluetooth device to be connected.	Y	Y	Y	Y	Y	Y	Y	Y
BTGetServiceInfo	Retrieves service information for Bluetooth device that communicates.	Y	Y	Y	Y	Y	Y	Y	Y
BTSelectDevice	Specifies Bluetooth device to be connected.	Y	Y	Y	Y	Y	Y	Y	Y
BTSetPassKey	Sets up Pass key.	Y	Y	Y	Y	Y	Y	Y	Y
BTTrustDevice	Trusts Bluetooth device.	Y	Y	Y	Y	Y	Y	Y	Y
BTSetWakeOnStatus	Sets up Bluetooth WakeOn function.	-	Y	Y	-	-	-	-	-
BTGetWakeOnStatus	Retrieves setting status for Bluetooth WakeOn function.	-	Y	Y	-	-	-	-	-
BTGetDeviceHandle	Retrieves device handle for Bluetooth device.	-	-	-	-	-	-	-	-
BTGetLastError	Retrieves detailed error information.	Y	Y	Y	Y	Y	Y	Y	Y
BTRRegisterLocalInfo	Registers Bluetooth local information in the registry.	Y	Y	Y	Y	Y	Y	Y	Y
BTRRegisterDeviceInfo	Registers Bluetooth device information in the registry.	Y	Y	Y	Y	Y	Y	Y	Y
BTSearchDeviceInfo	Retrieves Bluetooth device information from the registry.	Y	Y	Y	Y	Y	Y	Y	Y
BTDeleteDeviceInfo	Deletes Bluetooth device information in the registry.	Y	Y	Y	Y	Y	Y	Y	Y
BTSetDefaultDevice	Sets up default Bluetooth device.	Y	Y	Y	Y	Y	Y	Y	Y
BTGetDefaultDeviceInfo	Retrieves default Bluetooth device.	Y	Y	Y	Y	Y	Y	Y	Y
BTGetLibraryStatus	Retrieves current status of Bluetooth library.	Y	Y	Y	Y	Y	Y	Y	Y
BTGetDeviceName	Retrieves Bluetooth device name by specifying Bluetooth address.	Y	Y	Y	Y	Y	Y	Y	Y
BTGetConnectionStatus	Retrieves connection status with Bluetooth device.	Y	Y	Y	Y	Y	Y	Y	Y
BTSetConnectionParameter	Sets up parameters to be used for connecting to Bluetooth device.	Y	Y	Y	Y	Y	-	-	-
BTGetConnectionParameter	Retrieves parameters to be used to connecting Bluetooth device.	Y	Y	Y	Y	Y	-	-	-

BTSetAFHStatus	Sets up Bluetooth AFH mode.	Y	Y	Y	Y	Y	-	-	-
BTGetAFHStatus	Retrieves setting status of Bluetooth AFH.	Y	Y	Y	Y	Y	-	-	-
BTWaitForBtReady	Wait for time period to be elapsed until when communication takes place again after BTDeinitialize function is carried out.	-	Y	Y	Y	Y	Y	Y	Y
BTConnectSerial	Establishes connection using Bluetooth virtual serial profile.	Y	Y	Y	Y	Y	Y	Y	Y
BTSendSerialData	Sends data using Bluetooth virtual serial profile.	Y	Y	Y	Y	Y	Y	Y	Y
BTReceiveSerialData	Receives data using Bluetooth virtual serial profile.	Y	Y	Y	Y	Y	Y	Y	Y
BTDisconnectSerial	Disconnects Bluetooth virtual serial profile.	Y	Y	Y	Y	Y	Y	Y	Y
BTSetPANStatus	Sets up Bluetooth PAN adaptor status.	Y	Y	Y	Y	Y	-	-	-
BTGetPANStatus	Retrieves Bluetooth PAN adaptor status.	Y	Y	Y	Y	Y	-	-	-
BTConnectPAN	Establishes connection using Bluetooth PAN profile.	Y	Y	Y	Y	Y	-	-	-
BTDisconnectPAN	Disconnects Bluetooth PAN profile.	Y	Y	Y	Y	Y	-	-	-
BTConnectHeadset	Establishes connection to Bluetooth headset.	Y	Y	Y	-	-	-	-	-
BTDisconnectHeadset	Disconnects from Bluetooth headset.	Y	Y	Y	-	-	-	-	-
BTSetSoundPath	Sets up sound path for terminal.	Y	Y	Y	-	-	-	-	-
BTGetSoundPath	Retrieves status of sound path for terminal.	Y	Y	Y	-	-	-	-	-
BTSetHeadsetGain	Sets up gain for connected Bluetooth headset.	Y	Y	Y	-	-	-	-	-
BTGetHeadsetGain	Retrieves gain for connected Bluetooth headset.	Y	Y	Y	-	-	-	-	-
BTSetHeadsetServerStatus	Sets up operation to that of Bluetooth headset server.	Y	Y	Y	-	-	-	-	-
BTGetHeadsetServerStatus	Retrieves operating status as Bluetooth headset server.	Y	Y	Y	-	-	-	-	-
BTSetOBEXFolder	Sets up folder to be used in object push profile communication.	Y	Y	Y	Y	Y	-	-	-
BTGetOBEXFolder	Retrieves folder to be used in object push profile communication.	Y	Y	Y	Y	Y	-	-	-
BTSendOBEXFile	Uses object push profile to send file.	Y	Y	Y	Y	Y	-	-	-

Y: Supported.

- : Unsupported error responses when the function is specified.

5.1 BTInitialize

This function initializes the Bluetooth protocol stack and the Bluetooth virtual COM port to use the integrated Bluetooth device (module) in the terminal. Be sure to carry out this function before establishing a connection with other Bluetooth device.

In the Device Emulator, the function performs according to the instruction specified in the **BTInit.ini** file. See **BTInit.ini** for detail.

Calling Sequences

```
[C++]  
LONG BTInitialize( )
```

```
[Visual Basic]  
Public Shared Function BTInitialize() As Int32
```

```
[C#]  
public static Int32 BTInitialize()
```

Parameter

None

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to initialize.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.2 BTDeInitialize

This function disables the Bluetooth protocol stack and the Bluetooth virtual COM port to close the use of the integrated Bluetooth device in the terminal. Be sure to carry out this function after communication with other Bluetooth equipment is terminated.

In the Device Emulator, the function deletes also the PassKey set with **BTSetPassKey** function.

Calling Sequences

```
[C++]  
LONG BTDeInitialize( )
```

```
[Visual Basic]  
Public Shared Function BTDeInitialize() As Int32
```

```
[C#]  
public static Int32 BTDeInitialize()
```

Parameter

None

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to release the resource.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

If user application closes Bluetooth communication without carrying out this function, an error may occur when **BTInitialize** function is carried out a next time causing the Bluetooth device integrated in the terminal not operable.

5.3 BTGetLocalInfo

This function retrieves Bluetooth device information including all the members of BTST_LOCALINFO structure. For parameters that can be retrieved for Bluetooth device mode and Bluetooth device class, refer to Chapter 4 “Constants”.

In the Device Emulator, the function performs according to the instruction specified in the **BTInit.ini** file. See **BTInit.ini** for detail.

Calling Sequences

```
[C++]
LONG BTGetLocalInfo(
    BTST_LOCALINFO *LocalInfo
)
```

```
[Visual Basic]
Public Shared Function BTGetLocalInfo( _
    ByVal LocalInfo As BTST_LOCALINFO _
) As Int32
```

```
[C#]
public static Int32 BTGetLocalInfo(
    BTST_LOCALINFO LocalInfo
);
```

Parameters

LocalInfo

This parameter is for specifying the Bluetooth device information structure.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to retrieve device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.4 BTSetLocalInfo

This function sets up information about the Bluetooth device integrated in the terminal. Prior to carrying out the function, be sure to carry out **BTGetLocalInfo** function and retrieve the current Bluetooth device information. With this function, all device information can be set for the members of the BTST_LOCALINFO structure, except for Bluetooth address. Refer to Chapter 4 “Constants” concerning parameters that can be set for Bluetooth device mode and Bluetooth device class.

In the Device Emulator, the function performs according to the instruction specified in the **BTInit.ini** file. See **BTInit.ini** for detail.

Calling Sequences

```
[C++]
LONG BTSetLocalInfo(
    BTST_LOCALINFO *LocalInfo
)
```

```
[Visual Basic]
Public Shared Function BTSetLocalInfo( _
    ByVal LocalInfo As BTST_LOCALINFO _
) As Int32
```

```
[C#]
public static Int32 BTSetLocalInfo(
    BTST_LOCALINFO LocalInfo
);
```

Parameters

LocalInfo

This parameter is for specifying the Bluetooth device information structure to set.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- Among the Bluetooth device information, the Bluetooth device address should not be changed. If it is changed and this function is carried out, an error may occur.
- The cipher (encryption) setting flag can be set only when the Bluetooth authentication flag has been set enabled. Set the cipher setting flag disabled when the Bluetooth authentication flag has been disabled.

5.5 BTInquiry

This function is for carrying out an inquiry for Bluetooth equipment.

In the Device Emulator, the function searches **BTDeviceInfo[n].ni** files and then retrieves the number of the files resided.

Calling Sequences

```
[C++]
LONG BTInquiry(
    HANDLE *DeviceHandle,
    DWORD *DeviceNumber,
    DWORD InquiryTime
)
```

```
[Visual Basic]
Overloads Public Shared Function BTInquiry( _
    ByVal DeviceHandle As Int32(), _
    ByRef DeviceNumber As Int32, _
    ByVal InquiryTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTInquiry(
    Int32[] DeviceHandle,
    ref Int32 DeviceNumber,
    Int32 InquiryTime
);
```


Parameters

DeviceHandle

Set always “NULL” (in case of C++) or “IntPtr.Zero” (in case of Visual Basic or C#).

DeviceNumber

This parameter stores the number of Bluetooth equipment found by inquiry.

InquiryTime

This parameter is for specifying a time period in millisecond for carrying out the inquiry.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to inquire Bluetooth equipment.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

After carrying out **BTInquiry** function, retrieve Bluetooth equipment information with **BTGetDeviceInfo** function.

5.6 BTGetDeviceInfo

This function retrieves device information (equipment name, equipment address and device class) of Bluetooth equipment inquired with **BTInquiry** function. Be sure to carry out this function after carrying out **BTInquiry** function to retrieve inquired information.

In the Device Emulator, the function retrieves information about **BTDeviceInfo[n].ini** file. See **BTDeviceInfo[n].ini** for detail.

Calling Sequences

```
[C++]
LONG BTGetDeviceInfo(
    BTST_DEVICEINFO *DeviceInfo,
    DWORD DeviceNumber,
    HANDLE *DeviceHandle
)
```

```
Overloads Public Shared Function BTGetDeviceInfo( _
    ByVal DeviceInfo As BTST_DEVICEINFO(), _
    ByVal DeviceNumber As Int32, _
    ByVal DeviceHandle As Int32() _
) As Int32
```

```
public static Int32 BTGetDeviceInfo(
    BTST_DEVICEINFO[] DeviceInfo,
    Int32 DeviceNumber,
    Int32[] DeviceHandle
);
```

Parameters

DeviceInfo

This parameter is for specifying structure in array that stores Bluetooth equipment information.

Be sure to allocate enough memories for the number of Bluetooth equipment retrieved with **BTInquiry** function.

DeviceNumber

This parameter is for the number of structures in the *DeviceInfo* array.

Normally, this parameter is for the number of Bluetooth equipment retrieved with **BTInquiry** function.

DeviceHandle

Set always "NULL" (in case of C++) or "IntPtr.Zero" (in case of Visual Basic or C#).

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to retrieve Bluetooth equipment information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- If Bluetooth equipment names cannot be retrieved, the Bluetooth equipment addresses will be stored in the parameter for Bluetooth equipment names. Among the retrieved Bluetooth equipment information, do not change information other than the Bluetooth equipment names. As doing so may disable communication with the Bluetooth device.
- Once this function is carried out, the service information for Bluetooth equipment will be initialized. To retrieve service information of Bluetooth equipment, first carry out **BTGetDeviceInfo** function and then carry out **BTGetServiceInfo** function.

5.7 BTGetServiceInfo

This function retrieves service information of Bluetooth equipment. In addition to the information retrieved with **BTGetDeviceInfo** function, this function retrieves ProfileUUID information in **BTST_DEVICEINFO** structure. See Chapter 4 "Constants" for concerning service class parameters to retrieve. Prior to carrying out this function, carry out **BTGetDeviceInfo** function and retrieve device information other than the service information.

In the Device Emulator, the function retrieves information about **BTDeviceInfo[n].ini** file. See **BTDeviceInfo[n].ini** for detail.

Calling Sequences

```
[C++]
LONG BTGetServiceInfo(
    BTST_DEVICEINFO *DeviceInfo
)
```

```
[Visual Basic]
Public Shared Function BTGetServiceInfo( _
    ByVal DeviceInfo As BTST_DEVICEINFO _
) As Int32
```

```
[C#]
public static Int32 BTGetServiceInfo(
    BTST_DEVICEINFO DeviceInfo
);
```

Parameters

DeviceInfo

This parameter is for specifying the Bluetooth equipment information structure.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to retrieve Bluetooth equipment service information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- Even if this function is not carried out, it is still possible to establish communication with Bluetooth equipment. Use this function to check types of supported communication profiles by the partner Bluetooth equipment.
- Depending on Bluetooth equipment, it may be impossible to retrieve available profile information for reasons such as that SDP (Service Discovery Profile) is not supported by the equipment.
- After carrying out this function, if **BTGetDeviceInfo** function is carried out again, the Bluetooth service information stored in the *DeviceInfo* parameter will be deleted.

5.8 BTSelectDevice

This function specifies a partner Bluetooth equipment to communicate with. Carry out this function to select partner Bluetooth equipment before starting Bluetooth communication.

Calling Sequences

```
[C++]
LONG BTSelectDevice(
    BTST_DEVICEINFO *DeviceInfo,
    LPTSTR PortName
)
```

```
[Visual Basic]
Overloads Public Shared Function BTSelectDevice( _
    ByVal DeviceInfo As BTST_DEVICEINFO, _
    ByVal PortName As String _
) As Int32
```

```
[C#]
public static Int32 BTSelectDevice(
    BTST_DEVICEINFO DeviceInfo,
    string PortName
);
;
```

Parameters

DeviceInfo

This parameter is for specifying partner Bluetooth equipment to communicate with. Specify “NULL” (in case of C++) or “IntPtr.Zero” (in case of Visual Basic or C#) if the default Bluetooth equipment is to be specified.

PortName

This parameter is for specifying the communication port selecting either of the values listed below.

BTPORT_SERIAL	: Virtual serial
BTPORT_DIALUP	: Dial up
BTPORT_PAN	: Bluetooth PAN
BTPORT_OBEX	: OBEX Object Push
BTPORT_HEADSET	: Headset

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to specify Bluetooth equipment.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

If this function is not carried out, Bluetooth equipment set by default with **BTSetDefaultDevice** function will be specified as its partner Bluetooth equipment to communicate with.

5.9 BTSetPassKey

This function sets up PassKey used by the terminal. It is used when the terminal bonds with other Bluetooth equipment, or when a request for PassKey is made by other Bluetooth equipment. Setting the parameter can be made so that a request of the PassKey made by other Bluetooth equipment is refused. The PassKey has been set will be effect until when **BTSetPassKey** or **BTDeInitialize** function is carried out.

In the Device Emulator, this function writes information in **BTDeviceInfo[n].ini** file.

Calling Sequences

```
[C++]
LONG BTSetPassKey(
    LPTSTR PassKey
)
```

```
[Visual Basic]
Overloads Public Shared Function BTSetPassKey( _
    ByVal PassKey As String _
) As Int32
```

```
[C#]
public static Int32 BTSetPassKey(
    string PassKey
);
```

Parameters

PassKey

This parameter is for specifying the PassKey to set. If “NULL” (in case of C++) or “IntPtr.Zero” (in case of Visual Basic or C#) is specified, a request from other Bluetooth equipment will be refused.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set PassKey.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

Prior to carrying out **BTTrustDevice** function, always carry out this function. If it is necessary for other Bluetooth equipment to make a request of the PassKey to the terminal, carry out this function to set it in advance.

5.10 BTTrustDevice

This function bonds with other Bluetooth equipment specified by the terminal.

Calling Sequences

```
[C++]
LONG BTTrustDevice(
    BTST_DEVICEINFO *DeviceInfo,
    LPTSTR PortName
)
```

```
[Visual Basic]
Overloads Public Shared Function BTTrustDevice( _
    ByVal DeviceInfo As BTST_DEVICEINFO, _
    ByVal PortName As String _
) As Int32
```

```
[C#]
public static Int32 BTTrustDevice(
    BTST_DEVICEINFO DeviceInfo,
    string PortName
);
```

Parameters

DeviceInfo

This parameter is for specifying information about Bluetooth equipment that is bonded by the terminal. If “NULL” (in case of C++) or “IntPtr.Zero” (in case of Visual Basic or C#) is specified, the default Bluetooth equipment will be bonded.

PortName

BTPORT_SERIAL	: Virtual serial
BTPORT_DIALUP	: Dial up

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to bond Bluetooth equipment.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- Prior to carrying out the function, carry out **BTSetPassKey** function to set PassKey used when bonding. This function will return an error if a PassKey has not been set, or “NULL” or null character string (‘’) has been specified in the parameter of **BTSetPassKey** function.
- In case where there is a bonding request made by other Bluetooth equipment, the PassKey set with **BTSetPassKey** will automatically be sent, so this function does not need to be carried out.

5.11 BTSetWakeOnStatus

This function sets up the Bluetooth WakeOn function.

In the Device Emulator, the function writes information in **BTInit.ini** file.

Calling Sequences

```
[C++]
LONG BTSetWakeOnStatus(
    DWORD WakeOnStatus
)
```

```
[Visual Basic]
Public Shared Function BTSetWakeOnStatus( _
    ByVal WakeOnStatus As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTSetWakeOnStatus(
    Int32 WakeOnStatus
);
```

Parameters

WakeOnStatus

This parameter is for specifying the Bluetooth WakeOn function for the terminal selecting one of the values listed below.

BTWAKEON_ENABLE	: Enable the Bluetooth WakeOn function.
BTWAKEON_DISABLE	: Disable the Bluetooth WakeOn function.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

Set up the Bluetooth WakeOn function before suspending the terminal.

5.12 BTGetWakeOnStatus

This function retrieves the settings for Bluetooth WakeOn function.

In the Device Emulator, the function retrieves information about **BTInit.ini** file. See **BTInit.ini** for detail.

Calling Sequences

```
[C++]
LONG BTGetWakeOnStatus(
    DWORD *WakeOnStatus
)
```

```
[Visual Basic]
Public Shared Function BTGetWakeOnStatus( _
    ByRef WakeOnStatus As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetWakeOnStatus(
    ref Int32 WakeOnStatus
);
```

Parameters

WakeOnStatus

This parameter is for retrieving Bluetooth WakeOn function settings. See **BTSetWakeOnStatus** function for the values to retrieve.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to retrieve device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.13 BTGetDeviceHandle

This function retrieves the device handle of Bluetooth equipment.

Calling Sequences

```
[C++]
LONG BTGetDeviceHandle(
    HANDLE *DeviceHandle,
    LPTSTR BTAddress
)
```

```
[Visual Basic]
Public Shared Function BTGetDeviceHandle( _
    ByRef DeviceHandle As Int32, _
    ByVal BTAddress As String _
) As Int32
```

```
[C#]
public static Int32 BTGetDeviceHandle(
    ref Int32 DeviceHandle,
    string BTAddress
);
```

Parameters

DeviceHandle

This parameter is for retrieving the device handle of Bluetooth equipment.

BTAddress

This parameter is for specifying the Bluetooth address.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to retrieve the device handle of Bluetooth equipment.
BTERR_DRIVER	: Driver error
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- After retrieving the device handle for Bluetooth equipment, always carry out **BTGetDeviceInfo** function to retrieve device information for Bluetooth equipment.
- An error will occur if this function is carried out when device handles of Bluetooth equipment with specified address have been already retrieved with either **BTGetDeviceHandle** function or **BTInquiry** function. If so, use device handles that have been already retrieved.

5.14 BTGetLastError

This function retrieves detail information about an error occurred when the **Bluetooth Library** is called. The error detail retrieved with the function varies from model number of terminal to model number. For this reason, it is necessary to check the terminal model number with **SysGetModelName** function of the System Library.

Calling Sequences

```
[C++]  
LONG BTGetLastError( )
```

```
[Visual Basic]  
Public Shared Function BTGetLastError() As Int32
```

```
[C#]  
public static Int32 BTGetLastError()
```

Parameter

None

Return Values

This returns error code details. See Chapter 4.4 “Error Flag”.

Otherwise

FUNCTION_UN SUPPORT : Unsupported error

5.15 BTRegisterLocalInfo

This function registers information about the Bluetooth device integrated in the terminal in the registry. The registered device information in the registry will be set again when **BTInitialize** function is carried out a next time.

In the Device Emulator, the function writes information in **BTReg.ini** file.

Calling Sequences

```
[C++]  
LONG BTRegisterLocalInfo( )
```

```
[Visual Basic]  
Public Shared Function BTRegisterLocalInfo() As Int32
```

```
[C#]  
public static Int32 BTRegisterLocalInfo()
```

Parameter

None

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to register device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

To change Bluetooth device information and then register it in the registry, carry out **BTSetLocalInfo** function to change prior to carrying out this function.

5.16 BTRegisterDeviceInfo

This function registers Bluetooth equipment information in the registry. If the address of the Bluetooth equipment information to be registered is already registered in the registry, the registered Bluetooth equipment information will be overwritten by the new information.

In the Device Emulator, the function writes information in **BTReg.ini** file.

Calling Sequences

```
[C++]
LONG BTRegisterDeviceInfo(
    BTST_DEVICEINFO *DeviceInfo
)
```

```
[Visual Basic]
Public Shared Function BTRegisterDeviceInfo( _
    ByVal DeviceInfo As BTST_DEVICEINFO _
) As Int32
```

```
[C#]
public static Int32 BTRegisterDeviceInfo(
    BTST_DEVICEINFO DeviceInfo
);
```

Parameters

DeviceInfo

This is for specifying information for the Bluetooth equipment to register.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to register Bluetooth equipment information in the registry.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.17 BTSearchDeviceInfo

This function retrieves Bluetooth equipment information in the registry by specifying search key. Bluetooth equipment information that consistent with the search key can be also retrieved in the registry.

In the Device Emulator, the function retrieves information about **BTReg.ini** file. See **BTReg.ini** file.

Calling Sequences

```
[C++]
LONG BTSearchDeviceInfo(
    BTST_DEVICEINFO *DeviceInfo,
    DWORD *DeviceNumber,
    LPTSTR SearchKey
)
```

```
[Visual Basic]
Overloads Public Shared Function BTSearchDeviceInfo( _
    ByVal DeviceInfo As BTST_DEVICEINFO[], _
    ByRef DeviceNumber As Int32, _
    ByVal SearchKey As String _
) As Int32
```

```
[C#]
public static Int32 BTSearchDeviceInfo(
    BTST_DEVICEINFO[] DeviceInfo,
    ref Int32 DeviceNumber,
    string SearchKey
);
```

Parameters

DeviceInfo

This parameter is for specifying Bluetooth equipment information consistent with the search key. Prepare number of array greater than the value specified in **DeviceNumber** parameter. If “NULL” (in case of C++) or “IntPtr.Zero” (in case of Visual Basic or C#) is specified in this parameter, only the number of Bluetooth equipment consistent with the search key will be returned.

DeviceNumber

This parameter is for specifying the maximum value for Bluetooth equipment information to retrieve. After carrying out this, the number of Bluetooth equipment consistent with the search key will be stored.

SearchKey

This is for specifying Bluetooth equipment information search key, either Bluetooth address or Bluetooth equipment name. If “NULL” (in case of C++) or “IntPtr.Zero” (in case of Visual Basic or C#) is specified in this parameter, all information about the Bluetooth equipment registered in the registry will be returned.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to search Bluetooth equipment information in the registry.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- Prepare an array for the variable of structure that saves Bluetooth equipment information. If a small value is specified in ***DeviceNumber*** parameter, Bluetooth equipment information consistent with the search key may not be completely retrieved. Specify a value greater than the number of Bluetooth devices expected to be consistent with the search key.
- If the number of Bluetooth equipment information consistent with the search key cannot be predicted, first of all retrieve the number of Bluetooth equipment information consistent with the search key. After that, secure dramatically variable for the structure that saves Bluetooth equipment information, and then use the same search key to retrieve Bluetooth equipment information.

5.18 BTDeleteDeviceInfo

This function deletes Bluetooth equipment information in the registry.

In the Device Emulator, the function writes information in **BTReg.ini** file.

Calling Sequences

```
[C++]
LONG BTDeleteDeviceInfo(
    BT_DEVICEINFO *DeviceInfo
)
```

```
[Visual Basic]
Public Shared Function BTDeleteDeviceInfo( _
    ByVal DeviceInfo As BTST_DEVICEINFO _
) As Int32
```

```
[C#]
public static Int32 BTDeleteDeviceInfo(
    BTST_DEVICEINFO DeviceInfo
);
```

Parameters

DeviceInfo

This parameter is for specifying structure that stores Bluetooth equipment information to delete.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to delete Bluetooth equipment in the registry.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

Use **BTSearchDeviceInfo** function to retrieve Bluetooth equipment information. An error will occur if the Bluetooth equipment information (used as an argument) is not consistent with the Bluetooth equipment information registered in the registry.

5.19 BTGetDefaultDeviceInfo

This function retrieves information about Bluetooth equipment by default.

Calling Sequences

```
[C++]
LONG BTGetDefaultDeviceInfo(
    BTST_DEVICEINFO *DeviceInfo,
    LPTSTR PortName
)
```

```
[Visual Basic]
Public Shared Function BTGetDefaultDeviceInfo( _
    ByVal DeviceInfo As BTST_DEVICEINFO, _
    ByVal PortName As String _
) As Int32
```

```
[C#]
public static Int32 BTGetDefaultDeviceInfo(
    BTST_DEVICEINFO DeviceInfo,
    string PortName
);
```

Parameters

DeviceInfo

This parameter is for specifying the structure to retrieve information about Bluetooth equipment set by default.

PortName

BTPORT_SERIAL	: Virtual serial
BTPORT_DIALUP	: Dial up

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to retrieve information about Bluetooth equipment set by default.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

An error will occur if the terminal attempts to make communication establishment with Bluetooth equipment that has not been set by default.

5.20 BTSetDefaultDevice

This function sets up Bluetooth equipment by default to communicate with the terminal. The setting will become effect when **BTInitialize** function is carried out a next time. Once Bluetooth equipment registered by default to communicate with the terminal is specified by calling this function, the same Bluetooth equipment can be searched and selected with **BTSearchDeviceInfo** and **BTSelectDevice** functions as communication partner in a subsequent Bluetooth communication for the terminal.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **BTGetDefaultDeviceInfo** function.

Calling Sequences

```
[C++]
LONG BTSetDefaultDevice(
    BTST_DEVICEINFO *DeviceInfo,
    LPTSTR PortName
)
```

```
[Visual Basic]
Overloads Public Shared Function BTSetDefaultDevice( _
    ByVal DeviceInfo As BTST_DEVICEINFO, _
    ByVal PortName As String _
) As Int32
```

```
[C#]
public static Int32 BTSetDefaultDevice(
    BTST_DEVICEINFO DeviceInfo,
    string PortName
);
```

Parameters

DeviceInfo

This parameter is for specifying structure that saves information about Bluetooth equipment set by default. If “NULL” (in case of C++) or “IntPtr.Zero” (in case of Visual Basic or C#) is set in this parameter, there will be no Bluetooth equipment that is set by default.

PortName

BTPORT_SERIAL	: Virtual serial
BTPORT_DIALUP	: Dial up

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up Bluetooth equipment by default that communicates with the terminal.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UN SUPPORT	: Unsupported error

Notes:

- Even if this function is carried out, Bluetooth equipment specified as partner with **BTSelectDevice** function cannot be changed. Prior to carrying out this function, it is necessary to carry out **BTRegisterDeviceInfo** function to register the Bluetooth equipment information in the registry.
- Use **BTSearchDeviceInfo** function to retrieve the Bluetooth equipment information to use with this function. An error will occur if the Bluetooth equipment information is not consistent with the Bluetooth equipment information registered in the registry.

5.21 BTGetLibraryStatus

This function retrieves the current status of Bluetooth library. It can be carried out even if **BTInitialize** function has not been carried out.

Calling Sequences

```
[C++]
LONG BTGetLibraryStatus (
    DWORD *LibraryStatus
)
```

```
[Visual Basic]
Public Shared Function BTGetLibraryStatus( _
    ByRef LibraryStatus As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetLibraryStatus(
    ref Int32 LibraryStatus
);
```

Parameters

LibraryStatus

This parameter is variable that retrieves one of the values listed below for the current status of Bluetooth library.

BTSTATUS_NOT_INITIALIZED	: Not initialized
BTSTATUS_INITIALIZED	: Initialization completed
BTSTATUS_REINITIALIZING	: Processing initial initialization. In the Device Emulator, this parameter is ignored.
BTSTATUS_REINIT_FAILED	: Initial initialization failed. In the Device Emulator, this parameter is ignored.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.22 BTGetDeviceName

This function retrieves Bluetooth equipment name by specifying its Bluetooth address without carrying out **BTInquiry** function if it is known.

In the Device Emulator, the function retrieves information about **BTDeviceInfo[n].ini** file. See **BTDeviceInfo[n].ini** for detail.

Calling Sequences

```
[C++]
LONG BTGetDeviceName(
    BTST_DEVICEINFO *DeviceInfo,
    LPTSTR DeviceAddress
)
```

```
[Visual Basic]
Public Shared Function BTGetDeviceName( _
    ByVal DeviceInfo As BTST_DEVICEINFO, _
    ByVal DeviceAddress As String _
) As Int32
```

```
[C#]
public static Int32 BTGetDeviceName(
    BTST_DEVICEINFO DeviceInfo,
    string DeviceAddress
);
```

Parameters

DeviceInfo

This parameter is structure variable that saves Bluetooth equipment name.

DeviceAddress

This parameter is Bluetooth equipment address, such as "00:80:37:17:78:DA", for retrieving its equipment name.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.23 BTGetConnectionStatus

This function retrieves connection status of the terminal with other Bluetooth equipment.

Calling Sequences

```
[C++]
LONG BTGetConnectionStatus(
    DWORD *ConnectionStatus
)
```

```
[Visual Basic]
Public Shared Function BTGetConnectionStatus( _
    ByRef ConnectionStatus As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetConnectionStatus(
    ref Int32 ConnectionStatus
);
```

Parameters

ConnectionStatus

This parameter is variable that retrieves one of the values listed below for connection status with other Bluetooth equipment.

BTCONNECT_NO_CONNECTION	: Connection not established.
BTCONNECT_SERIAL_CLIENT	: Connection established via virtual serial port.
BTCONNECT_SERIAL_SERVER	: Connection established via virtual serial port.
BTCONNECT_PAN	: Connection established via PAN profile.
BTCONNECT_OBEX_CLIENT	: Connection established via OBEX profile.
BTCONNECT_OBEX_SERVER	: Connection established via OBEX profile.
BTCONNECT_HEADSET_CLIENT	: Connection established with BT headset.
BTCONNECT_HEADSET_SERVER	: Connection established with BT headset.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

Every time **BTConnectHeadset** function is carried out, make sure that the connection in Bluetooth has been established by carrying out **BTGetConnectionStatus** function.

5.24 BTSetConnectionParameter

This function sets up parameters used for communication with other Bluetooth equipment. Setting the parameter to either *SR mode* or *Fast Connection* changes frequency wave type used by the integrated Bluetooth module and may reduce a time required to complete the communication.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **BTGetConnectionParameter** function.

Calling Sequences

```
[C++]
LONG BTSetConnectionParameter(
    DWORD SRMode
    DWORD FastConnection
)
```

```
[Visual Basic]
Public Shared Function BTSetConnectionParameter( _
    ByVal SRMode As Int32, _
    ByVal FastConnection As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTSetConnectionParameter(
    Int32 SRMode,
    Int32 FastConnection
);
```

Parameters

SRMode

This parameter is variable that sets up the SR mode to one of the modes listed below.

BTSRMODE_R0	: Sets SR mode to R0
BTSRMODE_R1	: Sets SR mode to R1
BTSRMODE_R2	: Sets SR mode to R2

FastConnection

This parameter is variable that sets up the *Fast Connection* mode to either of the status listed below.

BTPARAM_DISABLE	: Sets Fast Connection mode disabled.
BTPARAM_ENABLE	: Sets Fast Connection mode enabled.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.25 BTGetConnectionParameter

This function retrieves parameters used when Bluetooth connection is established.

Calling Sequences

```
[C++]
LONG BTGetConnectionParameter(
    DWORD *SRMode
    DWORD *FastConnection
)
```

```
[Visual Basic]
Public Shared Function BTGetConnectionParameter( _
    ByRef SRMode As Int32, _
    ByRef FastConnection As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetConnectionParameter(
    ref Int32 SRMode,
    ref Int32 FastConnection
);
```

Parameters

SRMode

This parameter is for retrieving *SRMode* of the terminal. See **BTSetConnectionParameter** function for the values to retrieve.

FastConnection

This parameter is for retrieving *Fast Connection* mode of the terminal. See **BTSetConnectionParameter** function for the values to retrieve.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.26 BTSetAFHStatus

This function sets up Bluetooth AFH function.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **BTGetAFHStatus** function.

Calling Sequences

```
[C++]
LONG BTSetAFHStatus(
    DWORD AFHMode
    BYTE *AFHChannel
)
```

```
[Visual Basic]
Public Shared Function BTSetAFHStatus( _
    ByVal AFHMode As Int32, _
    ByVal AFHChannel As Byte() _
) As Int32
```

```
[C#]
public static Int32 BTSetAFHStatus(
    Int32 AFHMode,
    Byte[] AFHChannel
);
```

Parameters

AFHMode

This parameter is variable that sets up Bluetooth *AFH mode* selecting one of the values listed below.

- | | |
|---------------|--|
| BTAfh_DISABLE | : Disable the AFH setting. |
| BTAfh_AUTO | : Enable the AFH setting (Automatic setting). |
| BTAfh_MANUAL | : Enable the AFH setting. Frequency is specified in <i>AFHChannel</i> parameter. |

AFHChannel

This parameter is variable that sets up *AFH channel* using BYTE-type variable with 10 bytes area. This mode is operable only when BTAfh_MANUAL is set enabled in the *AFHMode* parameter.

However, even if BTAfh_DISABLE or BTAfh_AUTO is set in the *AFHMode* parameter, be sure to specify a BYTE-type variable with 10 bytes or more.

How to set up the *AFHchannel* parameter

For each channel of 79, you can specify “Enable” or “Disable”.

In the 10 bytes (i.e. 80 bits) area, set up the relevant bit to “1” for enabling the channel or “0” for disabling. The mapping of channel numbers "BTch" to the respective byte positions is as follows.

Table 5.2

Byte	9	8	7	6	5	4	3	2	1	0 (note)
BTch	0 to 7	8 to 15	16 to 23	24 to 31	32 to 39	40 to 47	48 to 55	56 to 63	64 to 71	72 to 78

Note:

The most significant bit of the first byte must always be set to 0.

The relation of BTch and Frequency (2402 - 2480 MHz) is as follows:

Frequency [MHz] = 2402 + BTch

Example:

Frequency range : 2451[MHz] to 2473[MHz]
BTch : 49ch to 71ch
Setting value in *AFHChannel* : 00 FF FF FE 00 00 00 00 00 00

Return Values

BTERR_SUCCESS : Normal end
BTERR_FAILED : Failed to set up device information.
BTERR_DRIVER : Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED : Unsupported error

5.27 BTGetAFHStatus

This function retrieves Bluetooth AFH function.

Calling Sequences

```
[C++]
LONG BTGetAFHStatus(
    DWORD *AFHMode
    BYTE *AFHChannel
)
```

```
[Visual Basic]
Public Shared Function BTGetAFHStatus( _
    ByRef AFHMode As Int32, _
    ByVal AFHChannel As Byte() _
) As Int32
```

```
[C#]
public static Int32 BTGetAFHStatus(
    ref Int32 AFHMode,
    Byte[] AFHChannel
);
```

Parameters

AFHMode

This parameter is variable that retrieves Bluetooth *AFH mode*. See **BTSetAFHStatus** function for the values to retrieve.

AFHChannel

This parameter is variable that retrieves Bluetooth *AFH channel*. See **BTSetAFHStatus** function for the values to retrieve.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.28 BTWaitForBtReady

After **BTDeInitialize** function has been carried out, this function waits for a time period to be elapsed until when communication in Bluetooth takes place again.

In the Device Emulator, the function does not perform, but returns “BTERR_SUCCESS”.

Calling Sequences

```
[C++]
LONG BTWaitForBtReady( )
```

```
[Visual Basic]
Public Shared Function BTWaitForBtReady( ) As Int32
```

```
[C#]
public static Int32 BTWaitForBtReady( );
```

Parameters

None

Return Values

BTERR_SUCCESS	: Normal end (possible to call BTInitialize function.)
BTERR_FAILED	: Failed to set up device information. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.29 BTConnectSerial

This function establishes connection with Bluetooth equipment via virtual serial profile.

In the Device Emulator, the function initializes only the **Bluetooth Library** and stores a result of the initialization as internal variable if it is succeeded.

Calling Sequences

```
[C++]
LONG BTConnectSerial (
    DWORD ConnectionMode,
    DWORD ConnectionTimeout,
    DWORD ReceiveTimeout
)
```

```
[Visual Basic]
Public Shared Function BTConnectSerial( _
    ByVal ConnectionMode As Int32, _
    ByVal ConnectionTimeout As Int32, _
    ByVal ReceiveTimeout As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTConnectSerial(
    Int32 ConnectionMode,
    Int32 ConnectionTimeout,
    Int32 ReceiveTimeout
);
```

Parameters

ConnectionMode

This parameter is for specifying connection mode selecting either of the values listed below.

BTCONNECT_SERIAL_CLIENT	: Establish connection in master mode.
BTCONNECT_SERIAL_SERVER	: Establish connection in slave mode.

ConnectionTimeout

This parameter is for specifying a time period in millisecond for timeout used in virtual serial profile mode.

ReceiveTimeout

This parameter is for specifying a time period in millisecond for timeout used in receiving data with **BTRemoveSerialData** function.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSupport	: Unsupported error

5.30 BTSendSerialData

This function sends data to Bluetooth equipment via virtual serial profile. It will operate in the same way as when **WriteFile** of Windows API is carried out in access to virtual serial port.

In the Device Emulator, the function sets up the size of data set in the *DataSize* parameter in the *SendSize* parameter.

Calling Sequences

```
[C++]
LONG BTSendSerialData(
    LPVOID Buffer,
    DWORD DataSize,
    DWORD *SendSize
)
```

```
[Visual Basic]
Public Shared Function BTSendSerialData( _
    ByVal Buffer As IntPtr, _
    ByVal DataSize As Int32, _
    ByRef SendSize As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTSendSerialData(
    IntPtr Buffer,
    Int32 DataSize,
    ref Int32 SendSize
);
```

Parameters

Buffer

Pointer to data to be sent (equivalent to the second parameter of WriteFile function)

DataSize

Size of data in byte to be sent (equivalent to the third parameter of WriteFile function)

SendSize

Size of data in byte actually sent (equivalent to the fourth parameter of WriteFile function)

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.31 BTReceiveSerialData

This function receives data from Bluetooth equipment via virtual serial profile. It will operate in the same way as when ReadFile function of Windows API is carried out in access to virtual serial port.

In the Device Emulator, the function sets up the maximum data size set in the *DataSize* parameter in the *ReceivedSize* parameter.

Calling Sequences

```
[C++]
LONG BTReceiveSerialData(
    LPVOID Buffer,
    DWORD DataSize,
    DWORD *ReceivedSize
)
```

```
[Visual Basic]
Public Shared Function BTReceiveSerialData( _
    ByRef Buffer As IntPtr, _
    ByVal DataSize As Int32, _
    ByRef ReceivedSize As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTReceiveSerialData (
    ref IntPtr Buffer,
    Int32 DataSize,
    ref Int32 ReceivedSize
);
```

Parameters

Buffer

Pointer to variable that stores data to be received (equivalent to the second parameter of ReadFile function)

DataSize

Maximum size of data in byte to receive (equivalent to the third parameter of ReadFile function)

ReceivedSize

Size of data in byte actually received (equivalent to the fourth parameter of ReadFile function)

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSupport	: Unsupported error

5.32 BTDisconnectSerial

This function disconnects connection established with Bluetooth equipment via virtual serial profile.

In the Device Emulator, the function checks only if **BTConnectSerial** function has been carried out.

Calling Sequences

```
[C++]  
LONG BTDisconnectSerial( )
```

```
[Visual Basic]  
Public Shared Function BTDisconnectSerial() As Int32
```

```
[C#]  
public static Int32 BTDisconnectSerial()
```

Parameter

None

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.33 BTSetPANStatus

This function sets up status of Bluetooth PAN Adapter.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **BTGetPANStatus** function.

Calling Sequences

```
[C++]
LONG BTSetPANStatus (
    DWORD BTPANStatus
)
```

```
[Visual Basic]
Public Shared Function BTSetPANStatus( _
    ByVal BTPANStatus As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTSetPANStatus(
    Int32 BTPANStatus
);
```

Parameters

BTPANStatus

This parameter is variable that sets up status of PAN adaptor. Select either of the values listed below.

BTPARAM_DISABLE	: Disable the BT PAN adaptor.
BTPARAM_ENABLE	: Enable the BT PAN adaptor.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.34 BTGetPANStatus

This function retrieves the status of Bluetooth PAN Adapter.

Calling Sequences

```
[C++]
LONG BTGetPANStatus (
    DWORD *BTPANStatus
)
```

```
[Visual Basic]
Public Shared Function BTGetPANStatus( _
    ByRef BTPANStatus As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetPANStatus(
    ref Int32 BTPANStatus
);
```

Parameters

BTPANStatus

The parameter is variable that retrieves status of PAN adaptor. See **BTSetPANStatus** function for the values to retrieve.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error
FUNCTION_UNSUPPORTED	: Unsupported error

5.35 BTConnectPAN

This function establishes connection with Bluetooth equipment via Bluetooth PAN Profile.

In the Device Emulator, the function checks only if the **Bluetooth Library** has been initialized and then stores a result of the checking as internal variable if it is succeeded.

Calling Sequences

```
[C++]
LONG BTConnectPAN(
    DWORD ConnectionTimeout
)
```

```
[Visual Basic]
Public Shared Function BTConnectPAN( _
    ByVal ConnectionTimeout As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTConnectPAN(
    Int32 ConnectionTimeout
);
```

Parameters

ConnectionTimeout

This parameter is for specifying a time period in millisecond for timeout in Bluetooth connection via Bluetooth PAN Profile.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.36 BTDisconnectPAN

This function terminates connection established with Bluetooth equipment via Bluetooth PAN Profile.

In the Device Emulator, the function checks only if **BTConnectPAN** function has been carried out.

Calling Sequences

```
[C++]  
LONG BTDisconnectPAN( )
```

```
[Visual Basic]  
Public Shared Function BTDisconnectPAN() As Int32
```

```
[C#]  
public static Int32 BTDisconnectPAN()
```

Parameter

None

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.37 BTConnectHeadset

This function establishes connection to Bluetooth headset.

In the Device Emulator, the function checks only if the **Bluetooth Library** has been initialized and then stores a result of the checking as internal variable if it is succeeded.

Calling Sequences

```
[C++]  
LONG BTConnectHeadset( )
```

```
[Visual Basic]  
Public Shared Function BTConnectHeadset() As Int32
```

```
[C#]  
public static Int32 BTConnectHeadset()
```

Parameter

None

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- Prior to carrying out this function, it is necessary to set up Bluetooth equipment to be connected using **BTSelectDevice** function or **BTSetDefaultDevice** function. An error will occur if the Bluetooth equipment to be connected has not been set.
- Connection to Bluetooth headset will not be established by carrying out this function. After carrying out this function, the partner Bluetooth headset must be operated to complete the connection.
- To check whether or not the connection with Bluetooth headset has been established, carry out this function followed by **BTGetConnectionStatus** function.

5.38 BTDisconnectHeadset

This function terminates connection established with Bluetooth headset.

In the Device Emulator, the function checks only if **BTConnectHeadset** function has been carried out.

Calling Sequences

```
[C++]  
LONG BTDisconnectHeadset( )
```

```
[Visual Basic]  
Public Shared Function BTDisconnectHeadset() As Int32
```

```
[C#]  
public static Int32 BTDisconnectHeadset()
```

Parameter

None

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- An error will occur if Bluetooth connection has not been established with **BTConnectHeadset** function.
- Terminating connection established with partner Bluetooth equipment may take prolonged period depending on the condition of the Bluetooth equipment.
- Carrying out this function is not necessary if the Bluetooth headset is operated to terminate the connection.

5.39 BTSetSoundPath

This function sets up status of sound path.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **BTGetSoundPath** function.

Calling Sequences

```
[C++]
LONG BTSetSoundPath(
    DWORD SoundPath
)
```

```
[Visual Basic]
Public Shared Function BTSetSoundPath( _
    ByVal SoundPath As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTSetSoundPath(
    Int32 SoundPath
);
```

Parameters

SoundPath

This parameter is variable that sets up status of sound path. Select either of the values listed below.

BTSOUND_INTERNAL	: Enable built-in speaker and microphone.
BTSOUND_HEADSET	: Enable Bluetooth headset.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.40 BTGetSoundPath

This function retrieves the status of the sound path in the terminal.

Calling Sequences

```
[C++]
LONG BTGetSoundPath(
    DWORD *SoundPath
)
```

```
[Visual Basic]
Public Shared Function BTGetSoundPath( _
    ByRef SoundPath As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetSoundPath(
    ref Int32 SoundPath
);
```

Parameters

SoundPath

This parameter is variable that retrieves the status of sound path. See **BTSetSoundPath** function for the values to retrieve.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.41 BTSetHeadsetGain

This function sets up gains for speaker and microphone of the connected Bluetooth headset.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **BTGetHeadsetGain** function.

Calling Sequences

```
[C++]
LONG BTSetHeadsetGain(
    DWORD SpeakerGain,
    DWORD MicrophoneGain
)
```

```
[Visual Basic]
Public Shared Function BTSetHeadsetGain( _
    ByVal SpeakerGain As Int32, _
    ByVal MicrophoneGain As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTSetHeadsetGain(
    Int32 SpeakerGain,
    Int32 MicrophoneGain
);
```

Parameters

SpeakerGain

This parameter is variable that sets up speaker's gain in the range of 0 to 15 for the Bluetooth headset.

MicrophoneGain

This parameter is variable that sets up microphone's gain in the range of 0 to 15 for Bluetooth headset.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.42 BTGetHeadsetGain

This function retrieves gains for the connected Bluetooth headset.

Calling Sequences

```
[C++]
LONG BTGetHeadsetGain(
    DWORD *SpeakerGain,
    DWORD *MicrophoneGain
)
```

```
[Visual Basic]
Public Shared Function BTGetHeadsetGain( _
    ByRef SpeakerGain As Int32, _
    ByRef MicrophoneGain As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetHeadsetGain(
    ref Int32 SpeakerGain,
    ref Int32 MicrophoneGain
);
```

Parameters

SpeakerGain

This parameter is for retrieving the gain for speaker. See **BTSetHeadsetGain** function for the values to retrieve.

MicrophoneGain

This parameter is for retrieving the gain for microphone. See **BTSetHeadsetGain** function for the values to retrieve.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.43 BTSetHeadsetServerStatus

This function sets up status of Bluetooth headset server.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **BTGetHeadsetServerStatus** function.

Calling Sequences

```
[C++]
LONG BTSetHeadsetServerStatus(
    DWORD ServerStatus
)
```

```
[Visual Basic]
Public Shared Function BTSetHeadsetServerStatus( _
    ByVal ServerStatus As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTSetHeadsetServerStatus(
    Int32 ServerStatus
);
```

Parameters

ServerStatus

This parameter is variable that sets up status of Bluetooth headset server. Select either of the values listed below.

BTPARAM_DISABLE	: Disable the Bluetooth headset server.
BTPARAM_ENABLE	: Enable the Bluetooth headset server.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.44 BTGetHeadsetServerStatus

This function retrieves the status of Bluetooth headset server.

Calling Sequences

```
[C++]
LONG BTGetHeadsetServerStatus(
    DWORD *ServerStatus
)
```

```
[Visual Basic]
Public Shared Function BTGetHeadsetServerStatus( _
    ByRef ServerStatus As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetHeadsetServerStatus(
    ref Int32 ServerStatus
);
```

Parameters

ServerStatus

This parameter is for retrieving the status of the server. See **BTSetHeadsetServerStatus** function for the values to retrieve.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.45 BTSetOBEXFolder

This function sets up a folder used for communication via Object Push profile.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **BTGetOBEXFolder** function.

Calling Sequences

```
[C++]
LONG BTSetOBEXFolder(
    LPTSTR FolderName
)
```

```
[Visual Basic]
Public Shared Function BTSetOBEXFolder( _
    ByVal FolderName As String _
) As Int32
```

```
[C#]
public static Int32 BTSetOBEXFolder(
    string FolderName
);
```

Parameters

FolderName

This parameter is variable that sets up a folder used for communication via Object Push profile.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.46 BTGetOBEXFolder

This function retrieves folder used for communication via Object Push profile.

In the Device Emulator, the function checks only if the **Bluetooth Library** has been initialized.

Calling Sequences

```
[C++]
LONG BTGetOBEXFolder(
    LPTSTR FolderName,
    DWORD FolderLength
)
```

```
[Visual Basic]
Public Shared Function BTGetOBEXFolder( _
    ByVal FolderName As String, _
    ByVal FolderLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTGetOBEXFolder(
    string FolderName,
    Int32 FolderLength
);
```

Parameters

FolderName

This parameter is variable that retrieves folder used for communication via Object Push profile.

FolderLength

This parameter is variable that sets up the length of variable in *FolderName* parameter.

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

5.47 BTSendOBEXFile

This function sends a file via Object Push profile.

In the Device Emulator, the function checks only if the **Bluetooth Library** has been initialized.

Calling Sequences

```
[C++]
LONG BTSendOBEXFile(
    LPTSTR FileName,
    DWORD ObjectType
)
```

```
[Visual Basic]
Public Shared Function BTSendOBEXFile( _
    ByVal FileName As String, _
    ByVal ObjectType As Int32 _
) As Int32
```

```
[C#]
public static Int32 BTSendOBEXFile(
    string FileName,
    Int32 ObjectType
);
```

Parameters

FileName

This parameter is for the name of file to send.

ObjectType

This parameter is object type of the file to be sent.

BTOBEX_OBJECT_VCARD	: Business card
BTOBEX_OBJECT_VCALENDAR	: Schedule
BTOBEX_OBJECT_VNOTE	: Memo
BTOBEX_OBJECT_VMESSAGE	: Mail

Return Values

BTERR_SUCCESS	: Normal end
BTERR_FAILED	: Failed to set up device information.
BTERR_DRIVER	: Driver error. In the Device Emulator, this value is not returned.
FUNCTION_UNSUPPORTED	: Unsupported error

6. Connection Procedure by Profile

This chapter explains the procedures for connecting and communicating with other Bluetooth equipment using profiles supported by the terminal.

6.1 Registering Partner Bluetooth Equipment

Register partner Bluetooth equipment in accordance with the following procedures when the Bluetooth device integrated in the terminal is used in master mode.

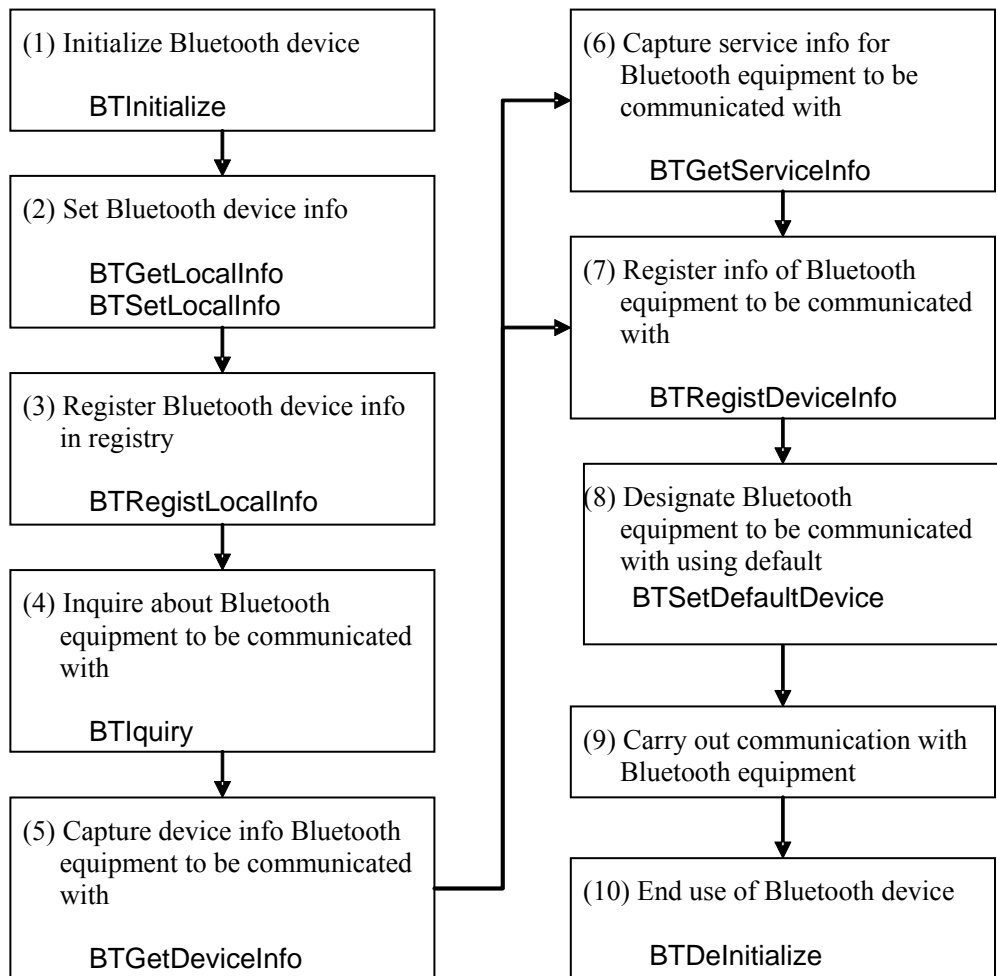


Figure 6.1

- The procedure (6) can be omitted.
- Carry out the procedure (8) to specify Bluetooth equipment used by default for partner equipment.
- Carry out the procedure (9) to start a communication immediately after registering Bluetooth equipment information.

6.2 Connections via Serial Profile

There are two connection methods available both via serial profile depending on the operation mode of Bluetooth device integrated in the terminal.

Connections in Client (Master) Mode

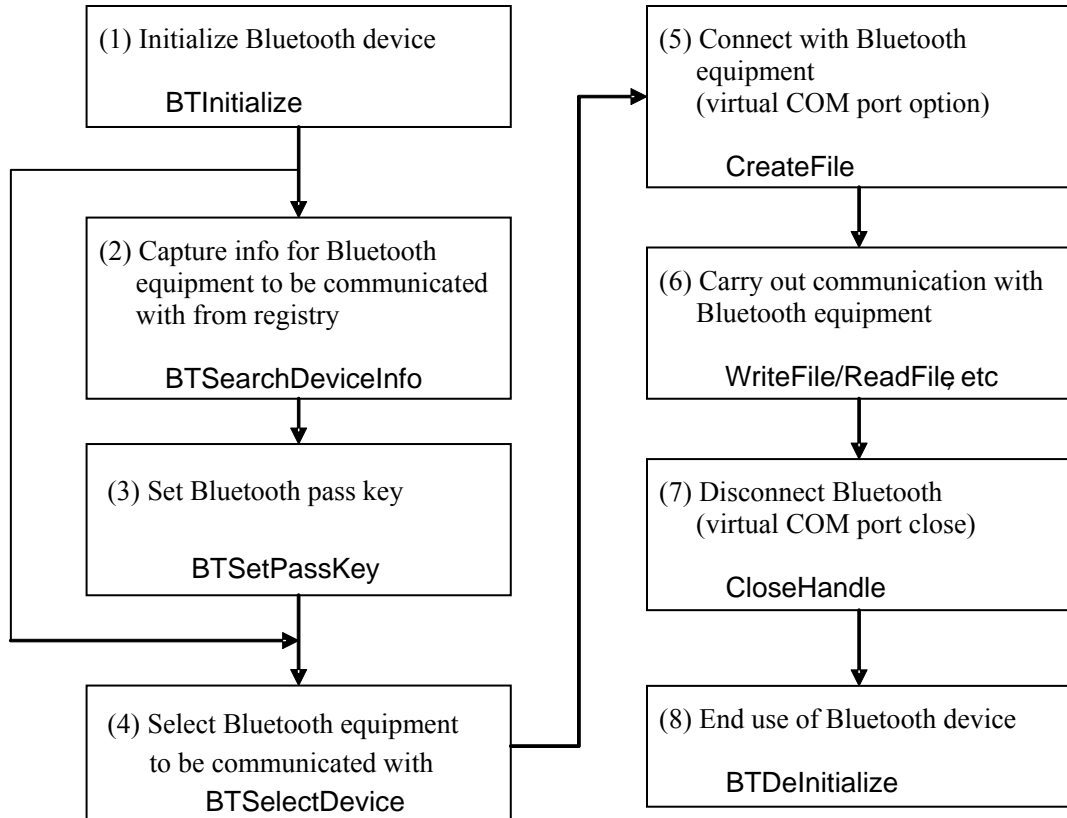


Figure 6.2

- The procedures (2) and (3) are not required for connection to be established with Bluetooth equipment specified by default.
- Carry out the procedure (4) when partner Bluetooth equipment makes a request for PassKey.
- Repeat the procedures (2) through (7) when serial communication takes place with multiple Bluetooth equipments.

Connections in Server (Slave) Mode

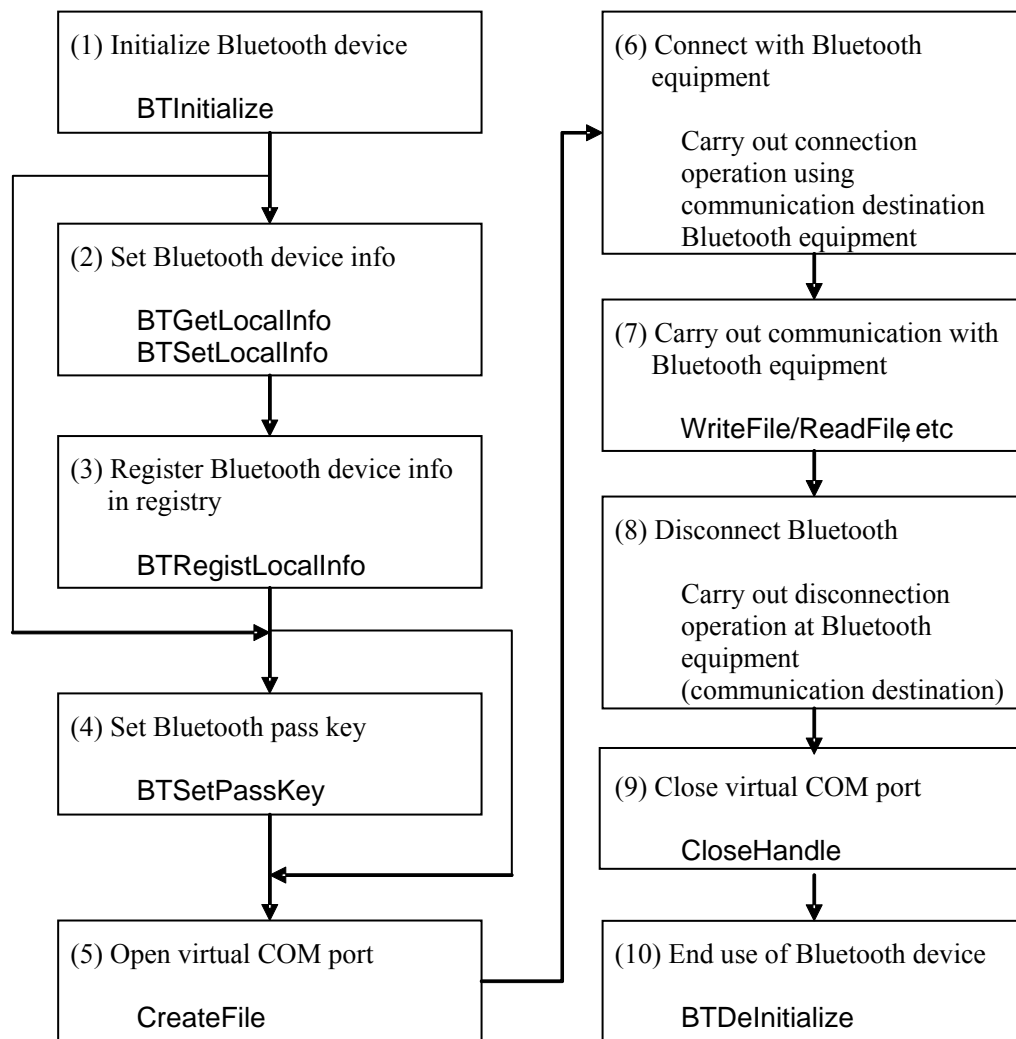


Figure 6.3

- Repeat the procedures (2) through (9) when serial communication takes place with multiple Bluetooth equipments.

6.3 Connections via Dial-Up Profile

Navigate to the terminal's control panel and then **Network and Dial-up Connections** icon to perform the procedures (1), (6), (7), and (8) described in the following flow chart.

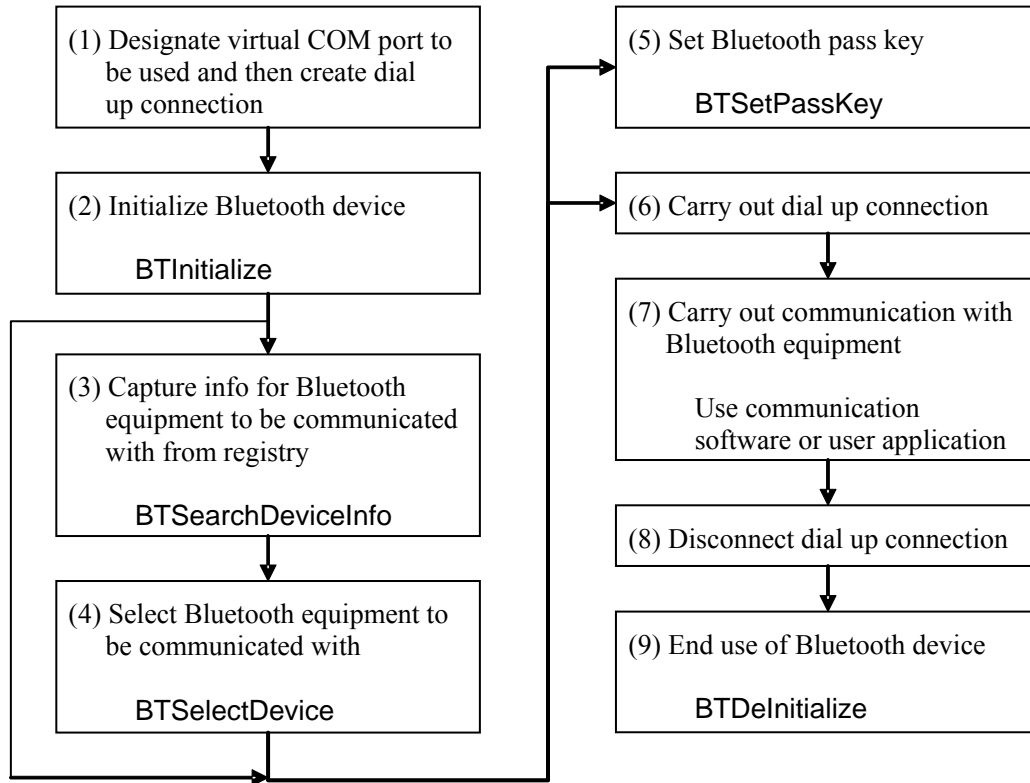


Figure 6.4

- The procedures (3) and (4) are not required for connection with Bluetooth equipment by default.
- Carry out the procedure (4) when partner Bluetooth equipment makes a request for PassKey.
- Repeat the procedures (3) through (8) when serial communication takes place with multiple Bluetooth equipments.

6.4 Connections via LAN Profile

Navigate to the terminal's control panel and then **Network and Dial-up Connections** icon to perform the procedures (1), (6), (7), and (8) described in the following flow chart.

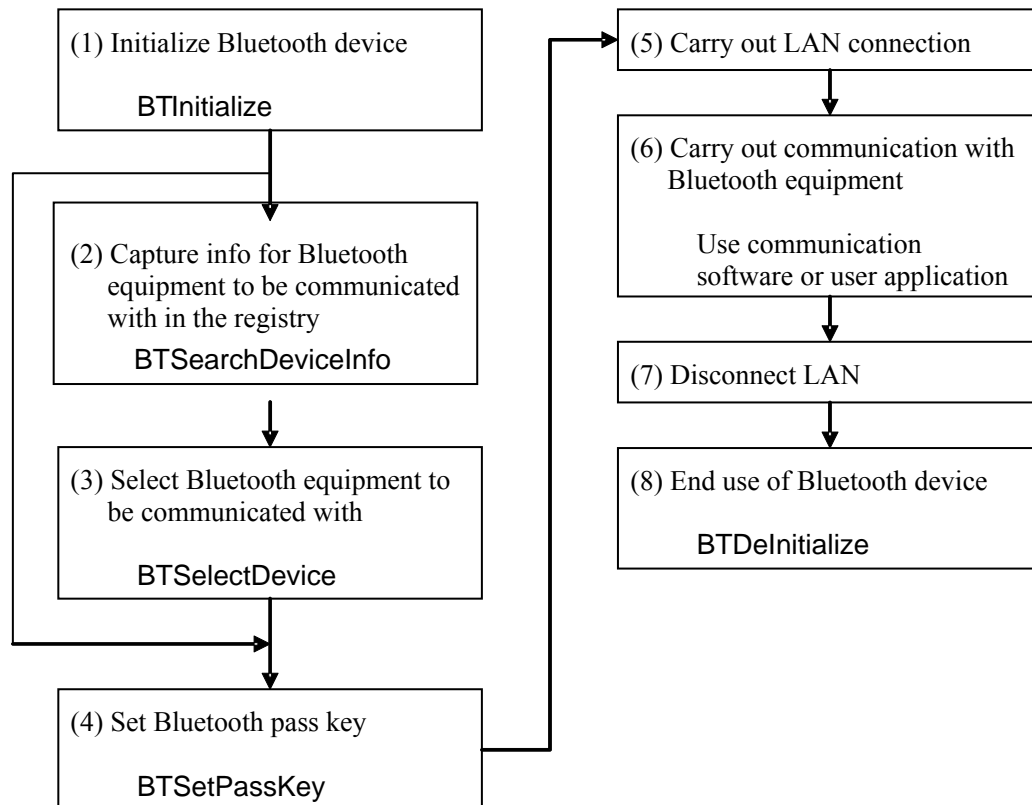


Figure 6.5

- The procedures (2) and (3) are not required for connection with default Bluetooth equipment.
- Carry out the procedure (4) when partner Bluetooth equipment makes a request for PassKey.
- Repeat the procedures (2) through (7) when serial communication takes place with multiple Bluetooth equipments.

7. Notes to Programming

Bluetooth connection tool, **Bluetooth Connection** icon, built in the terminal is available to establish a connection with other Bluetooth equipment. However, by using this Bluetooth library, direct establishment and communication can be made by the terminal with Bluetooth equipment in user application. Note however that the Bluetooth library and the Bluetooth connection tool cannot be used simultaneously. For example, if the Bluetooth connection tool is running, close it prior to starting up user application that uses the Bluetooth library.

7.1 Communication Profiles

The following shows communication profiles supported by the respective models. All the profiles use virtual COM port for communication.

Serial profile (master)	: COM6
Serial profile (slave)	: COM7
Dial up profile	: BTP1

7.2 Bluetooth Communication Modes

To communicate between pieces of Bluetooth equipment, they are configured in a wireless network called “Piconet”. Each Bluetooth equipment runs as either master or slave within this network.

If partner Bluetooth equipment is either Bluetooth modem, mobile phone, Access-Point, or printer, the terminal operates in master mode and the partner Bluetooth equipment operates in slave mode. If a partner Bluetooth equipment is either PC or other terminal (handheld terminal), one side operates in master mode and the other operates in slave mode.

The following procedures explain in case the terminal operates in master mode.

Registering device information of partner Bluetooth equipment

1. Carry out Bluetooth equipment inquiry.
2. Retrieve information about Bluetooth equipment discovered by the inquiry.
3. Register the retrieved information about Bluetooth equipment into the registry.
4. If necessary, set partner Bluetooth equipment by default that communicates with the terminal.

Selecting Bluetooth equipment and carrying out communication

1. Retrieve information about Bluetooth equipment in the registry.
2. Select partner Bluetooth equipment to communicate with the terminal.
3. If the partner Bluetooth equipment has been set by default, the operation in the step above is not required.
4. Establish a connection with the selected Bluetooth equipment and carry out communication.

The following procedures show in case the terminal operates in slave mode.

Registering device information about the terminal

1. Carry out an inquiry about the terminal by the partner Bluetooth equipment.
2. Register the device information for the terminal into the partner Bluetooth equipment.

Carrying out Bluetooth communication

1. Specify the terminal as Bluetooth equipment at the partner Bluetooth equipment to communicate.
2. Establish connection between the partner Bluetooth equipment and the terminal at the partner Bluetooth equipment.
3. Carry out communication between the partner Bluetooth equipment and the terminal.

8. Device Emulator

The following configuration files are required to make the **Bluetooth Library** run in the Device Emulator. These configuration files are installed by default in the “\Storage Card\Bluetooth” path.

- **BTInit.ini**
- **BTDeviceInfo[n].ini** (n; a numeric in the range of 0 to 256)
- **BTReg.ini**

8.1 BTInit.ini

This configuration file stores information about initialization on devices. The following shows a sample of the **BTInit.ini** file.

```
[Init]
Already=0
[LocalInfo]
LocalName="Emulator"
LocalAddress="00:00:00:00:00:00"
LocalDeviceMode=5
LocalClass1=1
LocalClass2=5
LocalClass3=0
Authentication=0
Encryption=0
[WakeOnStatus]
WakeOn=0
```

8.2 BTDeviceInfo[n].ini

This configuration file stores information about a device at communication partner. Adding this file in the folder can increase the number of devices at the communication partner. The following shows a sample of the **BTDeviceInfo[n].ini** file.

```
[DeviceInfo]
DeviceAddress="00:00:00:00:00:01"
DeviceName="BTDevice1"
DeviceClass1=1
DeviceClass2=5
DeviceClass3=0
ProfileUUID0=4353
ProfileUUID1=4354
PassKey="123"
```

Table 8.1

[DeviceInfo] Section for device information at communication partner	
DeviceAddress	Specifies the address of a device in character string. Refer to BTST_DEVICEINFO structure..
DeviceName	Specifies the name of a device in character string with 81 characters (maximum) or less. Refer to BTST_DEVICEINFO structure.
DeviceClass1	Specifies the class setup in numeric. Refer to BTST_DEVICEINFO structure.
DeviceClass2	Specifies the class setup in numeric. Refer to BTST_DEVICEINFO structure.
DeviceClass3	Specifies the class setup in numeric. Refer to BTST_DEVICEINFO structure.
ProfileUUID0 : ProfileUUID16	Specifies the profile in numeric. Refer to BTST_DEVICEINFO structure.
PassKey	Specifies PassKey for certification in character string. Refer to BTST_DEVICEINFO structure.

8.3 BTReg.ini

This configuration stores information about Bluetooth equipment retrieved in carrying out **BTGetDefaultDeviceInfo** function instead of **BTSetDefaultDevice** function.

In any event that **BTSetDefaultDevice** function is carried out, this configuration must be updated, too.

The following shows a sample of the **BTReg.ini** file. For each parameter written in the sample, refer to **BTInit.ini** and **BTDeviceInfo[n].ini** files.

```
[LocalInfo]
LocalName="Emulator"
LocalAddress="00:00:00:00:00:00"
LocalDeviceMode=5
LocalClass1=1
LocalClass2=5
LocalClass3=0
Authentication=0
Encryption=0
[DeviceInfo]
DeviceAddress="00:00:00:00:00:01"
DeviceName="BTDevice1"
DeviceClass1=1
DeviceClass2=5
DeviceClass3=0
ProfileUUID0=4353
ProfileUUID1=4354
PassKey="123"
```