

4.7.41 Host_Buffer_Size

Command	OCF	Command Parameters	Return Parameters
HCI_Host_Buffer_Size	0x0033	Host_ACL_Data_Packet_Length, Host_SCO_Data_Packet_Length, Host_Total_Num_ACL_Data_Packets, Host_Total_Num_SCO_Data_Packets	Status

Description:

The Host_Buffer_Size command is used by the Host to notify the Host Controller about the maximum size of the data portion of HCI ACL and SCO Data Packets sent from the Host Controller to the Host. The Host Controller will segment the data to be transmitted from the Host Controller to the Host according to these sizes, so that the HCI Data Packets will contain data with up to these sizes. The Host_Buffer_Size command also notifies the Host Controller about the total number of HCI ACL and SCO Data Packets that can be stored in the data buffers of the Host. If flow control from the Host Controller to the Host is turned off, and the Host_Buffer_Size command has not been issued by the Host, this means that the Host Controller will send HCI Data Packets to the Host with any lengths the Host Controller wants to use, and it is assumed that the data buffer sizes of the Host are unlimited. If flow control from the Host controller to the Host is turned on, the Host_Buffer_Size command must after a power-on or a reset always be sent by the Host before the first Host_Number_Of_Completed_Packets command is sent.

(The Set_Host_Controller_To_Host_Flow_Control command is used to turn flow control on or off.) The Host_ACL_Data_Packet_Length command parameter will be used to determine the size of the L2CAP segments contained in ACL Data Packets, which are transferred from the Host Controller to the Host. The Host_SCO_Data_Packet_Length command parameter is used to determine the maximum size of HCI SCO Data Packets. Both the Host and the Host Controller must support command and event packets, where the data portion (excluding header) contained in the packets is 255 bytes in size.

The Host_Total_Num_ACL_Data_Packets command parameter contains the total number of HCI ACL Data Packets that can be stored in the data buffers of the Host. The Host Controller will determine how the buffers are to be divided between different Connection Handles. The Host_Total_Num_SCO_Data_Packets command parameter gives the same information for HCI SCO Data Packets.

Note: the Host_ACL_Data_Packet_Length and Host_SCO_Data_Packet_Length command parameters do not include the length of the HCI Data Packet header.

Command Parameters:*Host_ACL_Data_Packet_Length:**Size: 2 Bytes*

Value	Parameter Description
0xXXXX	Maximum length (in bytes) of the data portion of each HCI ACL Data Packet that the Host is able to accept.

*Host_SCO_Data_Packet_Length:**Size: 1 Byte*

Value	Parameter Description
0xXX	Maximum length (in bytes) of the data portion of each HCI SCO Data Packet that the Host is able to accept.

*Host_Total_Num_ACL_Data_Packets:**Size: 2 Bytes*

Value	Parameter Description
0xXXXX	Total number of HCI ACL Data Packets that can be stored in the data buffers of the Host.

*Host_Total_Num_SCO_Data_Packets:**Size: 2 Bytes*

Value	Parameter Description
0xXXXX	Total number of HCI SCO Data Packets that can be stored in the data buffers of the Host.

Return Parameters:*Status:**Size: 1 Byte*

Value	Parameter Description
0x00	Host_Buffer_Size command succeeded.
0x01-0xFF	Host_Buffer_Size command failed. See Table 6.1 on page 745 for list of Error Codes.

Event(s) generated (unless masked away):

When the Host_Buffer_Size command has completed, a Command Complete event will be generated.

4.7.42 Host_Number_Of_Completed_Packets

Command	OCF	Command Parameters	Return Parameters
HCI_Host_Number_Of_Completed_Packets	0x0035	Number_Of_Handles, Connection_Handle[i], Host_Num_Of_Completed_Packets [i]	

Description:

The Host_Number_Of_Completed_Packets command is used by the Host to indicate to the Host Controller the number of HCI Data Packets that have been completed for each Connection Handle since the previous Host_Number_Of_Completed_Packets command was sent to the Host Controller. This means that the corresponding buffer space has been freed in the Host. Based on this information, and the Host_Total_Num_ACL_Data_Packets and Host_Total_Num_SCO_Data_Packets command parameters of the Host_Buffer_Size command, the Host Controller can determine for which Connection Handles the following HCI Data Packets should be sent to the Host. The command should only be issued by the Host if flow control in the direction from the Host Controller to the Host is on and there is at least one connection, or if the Host Controller is in local loopback mode. Otherwise, the command will be ignored by the Host Controller. While the Host has HCI Data Packets in its buffers, it must keep sending the Host_Number_Of_Completed_Packets command to the Host Controller at least periodically, until it finally reports that all buffer space in the Host used by ACL Data Packets has been freed. The rate with which this command is sent is manufacturer specific.

(The Set_Host_Controller_To_Host_Flow_Control command is used to turn flow control on or off.) If flow control from the Host controller to the Host is turned on, the Host_Buffer_Size command must after a power-on or a reset always be sent by the Host before the first Host_Number_Of_Completed_Packets command is sent.

Note: the Host_Number_Of_Completed_Packets command is a special command in the sense that no event is normally generated after the command has completed. The command may be sent at any time by the Host when there is at least one connection, or if the Host Controller is in local loopback mode independent of other commands. The normal flow control for commands is not used for the Host_Number_Of_Completed_Packets command.

Command Parameters:*Number_Of_Handles:**Size: 1 Byte*

Value	Parameter Description
0xXX	The number of Connection Handles and Host_Num_Of_Completed_Packets parameters pairs contained in this command. Range: 0-255

*Connection_Handle[i]: Size: Number_Of_Handles*2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

*Host_Num_Of_Completed_Packets [i]: Size: Number_Of_Handles * 2 Bytes*

Value	Parameter Description
N = 0xXXXX	The number of HCI Data Packets that have been completed for the associated Connection Handle since the previous time the event was returned. Range for N: 0x0000-0xFFFF

Return Parameters:

None.

Event(s) generated (unless masked away):

Normally, no event is generated after the Host_Number_Of_Completed_Packets command has completed. However, if the Host_Number_Of_Completed_Packets command contains one or more invalid parameters, the Host Controller will return a Command Complete event with a failure status indicating the Invalid HCI Command Parameters error code. The Host may send the Host_Number_Of_Completed_Packets command at any time when there is at least one connection, or if the Host Controller is in local loopback mode. The normal flow control for commands is not used for this command.

4.7.43 Read_Link_Supervision_Timeout

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Link_Supervision_Timeout	0x0036	Connection_Handle	Status, Connection_Handle, Link_Supervision_Timeout

Description:

This command will read the value for the Link_Supervision_Timeout parameter for the device. The Link_Supervision_Timeout parameter is used by the master or slave Bluetooth device to monitor link loss. If, for any reason, no Baseband packets are received from that Connection Handle for a duration longer than the Link_Supervision_Timeout, the connection is disconnected. The same timeout value is used for both SCO and ACL connections for the device specified by the Connection Handle.

Note: the Connection_Handle used for this command must be the ACL connection to the appropriate device. This command will set the Link_Supervision_Timeout values for other SCO Connection_Handle to that device.

Note: Setting the Link_Supervision_Timeout to No Link_Supervision_Timeout (0x0000) will disable the Link_Supervision_Timeout check for the specified Connection Handle. This makes it unnecessary for the master of the piconet to unpark and then park each Bluetooth Device every ~40 seconds. By using the No Link_Supervision_Timeout setting, the scalability of the Park mode is not limited.

Command Parameters:

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Link Supervision Timeout value is to be read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Byte*

Value	Parameter Description
0x00	Read_Link_Supervision_Timeout command succeeded.
0x01-0xFF	Read_Link_Supervision_Timeout command failed. See Table 6.1 on page 745 for list of Error Codes.

*Connection_Handle:**Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Link Supervision Timeout value was read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

*Link_Supervision_Timeout:**Size: 2 Bytes*

Value	Parameter Description
0x0000	No Link_Supervision_Timeout.
N = 0xXXXX	Measured in Number of Baseband slots Link_Supervision_Timeout = N * 0.625 msec (1 Baseband slot) Range for N: 0x0001 – 0xFFFF Time Range: 0.625ms - 40.9 sec

Event(s) generated (unless masked away):

When the Read_Link_Supervision_Timeout command has completed, a Command Complete event will be generated.

4.7.44 Write Link_Supervision_Timeout

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Link_Supervision_Timeout	0x0037	Connection_Handle, Link_Supervision_Timeout	Status, Connection_Handle

Description:

This command will write the value for the Link_Supervision_Timeout parameter for the device. The Link_Supervision_Timeout parameter is used by the master or slave Bluetooth device to monitor link loss. If, for any reason, no Baseband packets are received from that Connection_Handle for a duration longer than the Link_Supervision_Timeout, the connection is disconnected. The same timeout value is used for both SCO and ACL connections for the device specified by the Connection_Handle.

Note: the Connection_Handle used for this command must be the ACL connection to the appropriate device. This command will set the Link_Supervision_Timeout values for other SCO Connection_Handle to that device.

Note: Setting the Link_Supervision_Timeout parameter to No Link_Supervision_Timeout (0x0000) will disable the Link_Supervision_Timeout check for the specified Connection Handle. This makes it unnecessary for the master of the piconet to unpark and then park each Bluetooth Device every ~40 seconds. By using the No Link_Supervision_Timeout setting, the scalability of the Park mode is not limited.

Command Parameters:

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Link Supervision Timeout value is to be written. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Link_Supervision_Timeout: *Size: 2 Bytes*

Value	Parameter Description
0x0000	No Link_Supervision_Timeout.
N = 0xXXXX	Measured in Number of Baseband slots Link_Supervision_Timeout = N*0.625 msec (1 Baseband slot) Range for N: 0x0001 – 0xFFFF Time Range: 0.625ms – 40.9 sec Default: N = 0x7D00 Link_Supervision_Timeout = 20 sec

Return Parameters:*Status:**Size: 1 Byte*

Value	Parameter Description
0x00	Write_Link_Supervision_Timeout command succeeded.
0x01-0xFF	Write_Link_Supervision_Timeout command failed. See Table 6.1 on page 745 for list of Error Codes.

*Connection_Handle:**Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Link Supervision Timeout value was written. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Event(s) generated (unless masked away):

When the Write_Link_Supervision_Timeout command has completed, a Command Complete event will be generated.

4.7.45 Read_Number_Of_Supported_IAC

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Number_Of_Supported_IAC	0x0038		Status, Num_Support_IAC

Description:

This command will read the value for the number of Inquiry Access Codes (IAC) that the local Bluetooth device can simultaneous listen for during an Inquiry Scan. All Bluetooth devices are required to support at least one IAC, the General Inquiry Access Code (GIAC or UIAC), but some Bluetooth devices support additional IACs.

Command Parameters:

None

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Number_Of_Supported_IAC command succeeded.
0x01-0xFF	Read_Number_Of_Supported_IAC command failed. See Table 6.1 on page 745 for list of Error Codes.

Num_Support_IAC

Size: 1 Byte

Value	Parameter Description
0xXX	Specifies the number of Supported IAC that the local Bluetooth device can simultaneous listen for during an Inquiry Scan. Range: 0x01-0x40

Event(s) generated (unless masked away):

When the Read_Number_Of_Supported_IAC command has completed, a Command Complete event will be generated.

4.7.46 Read_Current_IAC_LAP

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Current_IAC_LAP	0x0039		Status, Num_Current_IAC, IAC_LAP[i]

Description:

This command reads the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans. All Bluetooth devices are required to support at least one IAC (GIAC or UIAC). Some Bluetooth devices support additional IACs.

Command Parameters:

None

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Current_IAC_LAP command succeeded.
0x01-0xFF	Read_Current_IAC_LAP command failed. See Table 6.1 on page 745 for list of Error Codes.

Num_Current_IAC

Size: 1 Byte

Value	Parameter Description
0xXX	Specifies the number of IACs which are currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan. Range: 0x01-0x40

IAC_LAP[i]

*Size: 3 Bytes * Num_Current_IAC*

Value	Parameter Description
0XXXXXXXX	LAPs used to create the IAC which is currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan. Range: 0x9E8B00-0x9E8B3F

Event(s) generated (unless masked away):

When the Read_Current_IAC_LAP command has completed, a Command Complete event will be generated.

4.7.47 Write_Current_IAC_LAP

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Current_IAC_LAP	0x003A	Num_Current_IAC, IAC_LAP[i]	Status

Description:

This command writes the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans. All Bluetooth devices are required to support at least one IAC (GIAC or UIAC). Some Bluetooth devices support additional IACs. Therefore, the LAP used to create the GIAC or UIAC must be among the IAC_LAP parameters of this command.

Note: this command writes over the current IACs used by the Bluetooth device. If the value of the NumCurrentIAC is more than the number of supported IACs, then only the first, X Inquiry Access Codes (where X equals the number of supported IACs) will be stored without any error.

Command Parameters:

Num_Current_IAC

Size: 1 Byte

Value	Parameter Description
0xXX	Specifies the number of IACs which are currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan. Range: 0x01-0x40

IAC_LAP[i]

*Size: 3 Bytes * Num_Current_IAC*

Value	Parameter Description
0XXXXXXXX	LAP(s) used to create IAC which is currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan. Range: 0x9E8B00-0x9E8B3F. The GIAC is the default IAC to be used. If additional IACs are supported, additional default IAC will be determined by the manufacturer.

Return Parameters:*Status:**Size: 1 Byte*

Value	Parameter Description
0x00	Write_Current_IAC_LAP command succeeded.
0x01-0xFF	Write_Current_IAC_LAP command failed. See Table 6.1 on page 745 for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Current_IAC_LAP command has completed, a Command Complete event will be generated.

4.7.48 Read_Page_Scan_Period_Mode

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Page_Scan_Period_Mode	0x003B		Status, Page_Scan_Period_Mode

Description:

This command is used to read the mandatory Page_Scan_Period_Mode of the local Bluetooth device. Every time an inquiry response message is sent, the Bluetooth device will start a timer (T_mandatory_pscan), the value of which is dependent on the Page_Scan_Period_Mode. As long as this timer has not expired, the Bluetooth device will use the Page_Scan_Period_Mode for all following page scans.

Note: the timer T_mandatory_pscan will be reset at each new inquiry response. For details see the "Baseband Specification" on page 33. (Keyword: SP-Mode, FHS-Packet, T_mandatory_pscan, Inquiry-Response).

After transmitting one or more inquiry response (FHS) packets as a result of the inquiry scan process, the local Bluetooth device is allowed to enter the page scan state using mandatory page scan mode regardless of the setting of the Scan_Enable parameter.

Command Parameters:

None

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Page_Scan_Period_Mode command succeeded.
0x01-0xFF	Read_Page_Scan_Period_Mode command failed. See Table 6.1 on page 745 for list of Error Codes.

Page_Scan_Period_Mode:

Size: 1 Byte

Value	Parameter Description
0x00	P0
0x01	P1
0x02	P2
0x03-0xFF	Reserved.

Event(s) generated (unless masked away):

When the Read_Page_Scan_Period_Mode command has completed, a Command Complete event will be generated.

4.7.49 Write_Page_Scan_Period_Mode

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Page_Scan_Period_Mode	0x003C	Page_Scan_Period_Mode	Status

Description:

This command is used to write the mandatory Page_Scan_Period_Mode of the local Bluetooth device. Every time an inquiry response message is sent, the Bluetooth device will start a timer (T_mandatory_pscan), the value of which is dependent on the Page_Scan_Period_Mode. As long as this timer has not expired, the Bluetooth device will use the Page_Scan_Period_Mode for all following page scans.

Note: the timer T_mandatory_pscan will be reset at each new inquiry response. For details see the "Baseband Specification" on page 33. (Keyword: SP-Mode, FHS-Packet, T_mandatory_pscan, Inquiry-Response).

After transmitting one or more inquiry response (FHS) packets as a result of the inquiry scan process, the local Bluetooth device is allowed to enter the page scan state using mandatory page scan mode regardless of the setting of the Scan_Enable parameter.

Command Parameters:

Page_Scan_Period_Mode:

Size: 1 Byte

Value	Parameter Description
0x00	P0. Default.
0x01	P1
0x02	P2
0x03-0xFF	Reserved.

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Write_Page_Scan_Period_Mode command succeeded.
0x01-0xFF	Write_Page_Scan_Period_Mode command failed. See Table 6.1 on page 745 for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Page_Scan_Period_Mode command has completed, a Command Complete event will be generated.

4.7.50 Read_Page_Scan_Mode

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Page_Scan_Mode	0x003D		Status, Page_Scan_Mode

Description:

This command is used to read the default page scan mode of the local Bluetooth device. The Page_Scan_Mode parameter indicates the page scan mode that is used for default page scan. Currently one mandatory page scan mode and three optional page scan modes are defined. Following an inquiry response, if the Baseband timer T_mandatory_pscan has not expired, the mandatory page scan mode must be applied. For details see the "Baseband Specification" on page 33 (Keyword: Page-Scan-Mode, FHS-Packet, T_mandatory_pscan)

Command Parameters:

None

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Page_Scan_Mode command succeeded.
0x01-0xFF	Read_Page_Scan_Mode command failed. See Table 6.1 on page 745 for list of Error Codes.

Page_Scan_Mode:

Size: 1 Byte

Value	Parameter Description
0x00	Mandatory Page Scan Mode
0x01	Optional Page Scan Mode I
0x02	Optional Page Scan Mode II
0x03	Optional Page Scan Mode III
0x04-0xFF	Reserved

Event(s) generated (unless masked away):

When the Read_Page_Scan_Mode command has completed, a Command Complete event will be generated.

4.7.51 Write_Page_Scan_Mode

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Page_Scan_Mode	0x003E	Page_Scan_Mode	Status

Description:

This command is used to write the default page scan mode of the local Bluetooth device. The Page_Scan_Mode parameter indicates the page scan mode that is used for the default page scan. Currently, one mandatory page scan mode and three optional page scan modes are defined. Following an inquiry response, if the Baseband timer T_mandatory_pscan has not expired, the mandatory page scan mode must be applied. For details see the "Baseband Specification" on page 33. (Keyword: Page-Scan-Mode, FHS-Packet, T_mandatory_pscan).

Command Parameters:

Page_Scan_Mode:

Size: 1 Byte

Value	Parameter Description
0x00	Mandatory Page Scan Mode. Default.
0x01	Optional Page Scan Mode I
0x02	Optional Page Scan Mode II
0x03	Optional Page Scan Mode III
0x04-0xFF	Reserved.

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Write_Page_Scan_Mode command succeeded.
0x01-0xFF	Write_Page_Scan_Mode command failed. See Table 6.1 on page 745 for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Page_Scan_Mode command has completed, a Command Complete event will be generated.

4.8 INFORMATIONAL PARAMETERS

The Informational Parameters are fixed by the manufacturer of the Bluetooth hardware. These parameters provide information about the Bluetooth device and the capabilities of the Host Controller, Link Manager, and Baseband. The host device cannot modify any of these parameters. For Informational Parameters Commands, the OGF is defined as 0x04

Command	Command Summary Description
Read_Local_Version_Information	The Read_Local_Version_Information command will read the values for the version information for the local Bluetooth device.
Read_Local_Supported_Features	The Read_Local_Supported_Features command requests a list of the supported features for the local device.
Read_Buffer_Size	The Read_Buffer_Size command returns the size of the HCI buffers. These buffers are used by the Host Controller to buffer data that is to be transmitted.
Read_Country_Code	The Read_Country_Code command will read the value for the Country Code status parameter. The Country Code defines which range of frequency band of the ISM 2.4 GHz band will be used by the device.
Read_BD_ADDR	The Read_BD_ADDR command will read the value for the BD_ADDR parameter. The BD_ADDR is a 48-bit unique identifier for a Bluetooth device.

4.8.1 Read_Local_Version_Information

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Version_Information	0x0001		Status, HCI Version, HCI Revision, LMP Version, Manufacturer_Name, LMP Subversion

Description:

This command will read the values for the version information for the local Bluetooth device. The version information consists of two parameters: the version and revision parameters.

The version parameter defines the major hardware version of the Bluetooth hardware. The version parameter only changes when new versions of the Bluetooth hardware are produced for new Bluetooth SIG specifications. The version parameter is controlled by the SIG.

The revision parameter should be controlled by the manufacturer and should be changed as needed. The Manufacturer_Name parameter indicates the manufacturer of the local Bluetooth module as specified by the LMP definition of this parameter. The subversion parameter should be controlled by the manufacturer and should be changed as needed. The subversion parameter defines the various revisions that each version of the Bluetooth hardware will go through as design processes change and errors are fixed. This allows the software to determine what Bluetooth hardware is being used, and to work around various bugs in the hardware if necessary.

Command Parameters:

None.

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Local_Version_Information command succeeded.
0x01-0xFF	Read_Local_Version_Information command failed. See Table 6.1 on page 745 for list of Error Codes.

*HCI_Version:**Size: 1 Byte*

Value	Parameter Description
0xXX	Version of the Current HCI in the Bluetooth hardware. 0x00: Bluetooth HCI Specification 1.0 0x01-0xFF: Reserved

*HCI_Revision:**Size: 2 Bytes*

Value	Parameter Description
0XXXXX	Revision of the Current HCI in the Bluetooth hardware.

*LMP_Version:**Size: 1 Byte*

Value	Parameter Description
0xXX	Version of the Current LMP in the Bluetooth Hardware, see Table 5.2 on page 231 in the Link Manager Protocol for assigned values (VersNr).

*Manufacturer_Name:**Size: 2 Bytes*

Value	Parameter Description
0XXXXX	Manufacturer Name of the Bluetooth Hardware, see Table 5.2 on page 231 in the Link Manager Protocol for assigned values (Compld).

*LMP_Subversion:**Size: 2 Bytes*

Value	Parameter Description
0XXXXX	Subversion of the Current LMP in the Bluetooth Hardware, see Table 5.2 on page 231 in the Link Manager Protocol for assigned values (SubVersNr).

Event(s) generated (unless masked away):

When the Read_Local_Version_Information command has completed, a Command Complete event will be generated.

4.8.2 Read_Local_Supported_Features

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Supported_Features	0x0003		Status, LMP_Features

Description:

This command requests a list of the supported features for the local device. This command will return a list of the LMP features. For details see "Link Manager Protocol" on page 185.

Command Parameters:

None.

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Local_Supported_Features command succeeded.
0x01-0xFF	Read_Local_Supported_Features command failed. See Table 6.1 on page 745 for list of Error Codes.

LMP_Features:

Size: 8 Bytes

Value	Parameter Description
0XXXXXXXXX XXXXXXXX	Bit Mask List of LMP features. For details see "Link Manager Protocol" on page 185

Event(s) generated (unless masked away):

When the Read_Local_Supported_Features command has completed, a Command Complete event will be generated.

4.8.3 Read_Buffer_Size

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Buffer_Size	0x0005		Status, HC_ACL_Data_Packet_Length, HC_SCO_Data_Packet_Length, HC_ Total_Num_ACL_Data_Packets, HC_Total_Num_SCO_Data_Packets

Description:

The Read_Buffer_Size command is used to read the maximum size of the data portion of HCI ACL and SCO Data Packets sent from the Host to the Host Controller. The Host will segment the data to be transmitted from the Host to the Host Controller according to these sizes, so that the HCI Data Packets will contain data with up to these sizes. The Read_Buffer_Size command also returns the total number of HCI ACL and SCO Data Packets that can be stored in the data buffers of the Host Controller. The Read_Buffer_Size command must be issued by the Host before it sends any data to the Host Controller.

The HC_ACL_Data_Packet_Length return parameter will be used to determine the size of the L2CAP segments contained in ACL Data Packets, which are transferred from the Host to the Host Controller to be broken up into baseband packets by the Link Manager. The HC_SCO_Data_Packet_Length return parameter is used to determine the maximum size of HCI SCO Data Packets. Both the Host and the Host Controller must support command and event packets, where the data portion (excluding header) contained in the packets is 255 bytes in size. The HC_Total_Num_ACL_Data_Packets return parameter contains the total number of HCI ACL Data Packets that can be stored in the data buffers of the Host Controller. The Host will determine how the buffers are to be divided between different Connection Handles. The HC_Total_Num_SCO_Data_Packets return parameter gives the same information but for HCI SCO Data Packets.

Note: the HC_ACL_Data_Packet_Length and HC_SCO_Data_Packet_Length return parameters do not include the length of the HCI Data Packet header.

Command Parameters:

None.

Return Parameters:*Status:**Size: 1 Byte*

Value	Parameter Description
0x00	Read_Buffer_Size command succeeded.
0x01-0xFF	Read_Buffer_Size command failed. See Table 6.1 on page 745 for list of Error Codes.

*HC_ACL_Data_Packet_Length:**Size: 2 Bytes*

Value	Parameter Description
0xFFFF	Maximum length (in bytes) of the data portion of each HCI ACL Data Packet that the Host Controller is able to accept.

*HC_SCO_Data_Packet_Length:**Size: 1 Byte*

Value	Parameter Description
0xFF	Maximum length (in bytes) of the data portion of each HCI SCO Data Packet that the Host Controller is able to accept.

*HC_Total_Num_ACL_Data_Packets:**Size: 2 Bytes*

Value	Parameter Description
0xFFFF	Total number of HCI ACL Data Packets that can be stored in the data buffers of the Host Controller.

*HC_Total_Num_SCO_Data_Packets:**Size: 2 Bytes*

Value	Parameter Description
0xFFFF	Total number of HCI SCO Data Packets that can be stored in the data buffers of the Host Controller.

Event(s) generated (unless masked away):

When the Read_Buffer_Size command has completed, a Command Complete event will be generated.

4.8.4 Read_Country_Code

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Country_Code	0x0007		Status, Country_Code

Description:

This command will read the value for the Country_Code return parameter. The Country_Code defines which range of frequency band of the ISM 2.4 GHz band will be used by the device. Each country has local regulatory bodies regulating which ISM 2.4 GHz frequency ranges can be used.

Command Parameters:

None.

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Country_Code command succeeded.
0x01-0xFF	Read_Country_Code command failed. See Table 6.1 on page 745 for list of Error Codes.

Country_Code:

Size: 1 Byte

Value	Parameter Description
0x00	North America & Europe*
0x01	France
0x02	Spain
0x03	Japan
0x04-FF	Reserved for Future Use.

*. Except Spain and France

Event(s) generated (unless masked away):

When the Read_Country_Code command has completed, a Command Complete event will be generated.

4.8.5 Read_BD_ADDR

Command	OCF	Command Parameters	Return Parameters
HCI_Read_BD_ADDR	0x0009		Status, BD_ADDR

Description:

This command will read the value for the BD_ADDR parameter. The BD_ADDR is a 48-bit unique identifier for a Bluetooth device. See the "Baseband Specification" on page 33 for details of how BD_ADDR is used.

Command Parameters:

None.

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_BD_ADDR command succeeded.
0x01-0xFF	Read_BD_ADDR command failed. See Table 6.1 on page 745 for list of Error Codes.

BD_ADDR:

Size: 6 Bytes

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR of the Device

Event(s) generated (unless masked away):

When the Read_BD_ADDR command has completed, a Command Complete event will be generated.

4.9 STATUS PARAMETERS

The Host Controller modifies all status parameters. These parameters provide information about the current state of the Host Controller, Link Manager, and Baseband. The host device cannot modify any of these parameters other than to reset certain specific parameters. For the Status and baseband, the OGF is defined as 0x05

Command	Command Summary Description
Read_Failed_Contact_Counter	The Read_Failed_Contact_Counter will read the value for the Failed_Contact_Counter parameter for a particular connection to another device. The Failed_Contact_Counter records the number of consecutive incidents in which either the slave or master didn't respond after the flush timeout had expired, and the L2CAP packet that was currently being transmitted was automatically 'flushed'.
Reset_Failed_Contact_Counter	The Reset_Failed_Contact_Counter will reset the value for the Failed_Contact_Counter parameter for a particular connection to another device. The Failed_Contact_Counter records the number of consecutive incidents in which either the slave or master didn't respond after the flush timeout had expired and the L2CAP packet that was currently being transmitted was automatically 'flushed'.
Get_Link_Quality	The Get_Link_Quality command will read the value for the Link_Quality for the specified Connection Handle.
Read_RSSI	The Read_RSSI command will read the value for the Received Signal Strength Indication (RSSI) for a connection handle to another Bluetooth device.

4.9.1 Read_Failed_Contact_Counter

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Failed_Contact_Counter	0x0001	Connection_Handle	Status, Connection_Handle, Failed_Contact_Counter

Description:

This command will read the value for the Failed_Contact_Counter parameter for a particular connection to another device. The Connection_Handle must be a Connection_Handle for an ACL connection. The Failed_Contact_Counter records the number of consecutive incidents in which either the slave or master didn't respond after the flush timeout had expired, and the L2CAP packet that was currently being transmitted was automatically 'flushed'. When this occurs, the Failed_Contact_Counter is incremented by 1. The Failed_Contact_Counter for a connection is reset to zero on the following conditions:

1. When a new connection is established
2. When the Failed_Contact_Counter is > zero and an L2CAP packet is acknowledged for that connection
3. When the Reset_Failed_Contact_Counter command has been issued

Command Parameters:

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Failed Contact Counter should be read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Byte*

Value	Parameter Description
0x00	Read_Failed_Contact_Counter command succeeded.
0x01-0xFF	Read_Failed_Contact_Counter command failed. See Table 6.1 on page 745 for list of Error Codes.

*Connection_Handle:**Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Failed Contact Counter has been read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

*Failed_Contact_Counter:**Size: 2 Bytes*

Value	Parameter Description
0xXXXX	Number of consecutive failed contacts for a connection corresponding to the connection handle.

Event(s) generated (unless masked away):

When the Read_Failed_Contact_Counter command has completed, a Command Complete event will be generated.

4.9.2 Reset_Failed_Contact_Counter

Command	OCF	Command Parameters	Return Parameters
HCI_Reset_Failed_Contact_Counter	0x0002	Connection_Handle	Status, Connection_Handle

Description:

This command will reset the value for the Failed_Contact_Counter parameter for a particular connection to another device. The Connection_Handle must be a Connection_Handle for an ACL connection. The Failed_Contact_Counter records the number of consecutive incidents in which either the slave or master didn't respond after the flush timeout had expired, and the L2CAP packet that was currently being transmitted was automatically 'flushed'. When this occurs, the Failed_Contact_Counter is incremented by 1. The Failed_Contact_Counter for a connection is reset to zero on the following conditions:

1. When a new connection is established
2. When the Failed_Contact_Counter is > zero and an L2CAP packet is acknowledged for that connection
3. When the Reset_Failed_Contact_Counter command has been issued

Command Parameters:

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Failed Contact Counter should be reset. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Byte*

Value	Parameter Description
0x00	Reset_Failed_Contact_Counter command succeeded.
0x01-0xFF	Reset_Failed_Contact_Counter command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Failed Contact Counter has been reset. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Event(s) generated (unless masked away):

When the Reset_Failed_Contact_Counter command has completed, a Command Complete event will be generated.

4.9.3 Get_Link_Quality

Command	OCF	Command Parameters	Return Parameters
HCI_Get_Link_Quality	0x0003	Connection_Handle	Status, Connection_Handle, Link_Quality

Description:

This command will return the value for the Link_Quality for the specified Connection Handle. The Connection_Handle must be a Connection_Handle for an ACL connection. This command will return a Link_Quality value from 0-255, which represents the quality of the link between two Bluetooth devices. The higher the value, the better the link quality is. Each Bluetooth module vendor will determine how to measure the link quality.

Command Parameters:

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the connection for which link quality parameters are to be read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Byte*

Value	Parameter Description
0x00	Get_Link_Quality command succeeded.
0x01-0xFF	Get_Link_Quality command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the connection for which the link quality parameter has been read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

*Link_Quality:**Size: 1 Byte*

Value	Parameter Description
0xXX	The current quality of the Link connection between the local device and the remote device specified by the Connection Handle Range: 0x00 – 0xFF The higher the value, the better the link quality is.

Event(s) generated (unless masked away):

When the Get_Link_Quality command has completed, a Command Complete event will be generated.

4.9.4 Read_RSSI

Command	OCF	Command Parameters	Return Parameters
HCI_Read_RSSI	0x0005	Connection_Handle	Status, Connection_Handle,RSSI

Description:

This command will read the value for the difference between the measured Received Signal Strength Indication (RSSI) and the limits of the Golden Receive Power Range (see Radio Specification Section 4.7 on page 26) for a connection handle to another Bluetooth device. The Connection_Handle must be a Connection_Handle for an ACL connection. Any positive RSSI value returned by the Host Controller indicates how many dB the RSSI is above the upper limit, any negative value indicates how many dB the RSSI is below the lower limit. The value zero indicates that the RSSI is inside the Golden Receive Power Range.

Note: how accurate the dB values will be depends on the Bluetooth hardware. The only requirements for the hardware are that the Bluetooth device is able to tell whether the RSSI is inside, above or below the Golden Device Power Range.

Command Parameters:

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the RSSI is to be read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Byte*

Value	Parameter Description
0x00	Read_RSSI command succeeded.
0x01-0xFF	Read_RSSI command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the RSSI has been read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

RSSI:*Size: 1 Byte*

Value	Parameter Description
N = 0xXX	Size: 1 Byte (signed integer) Range: $-128 \leq N \leq 127$ Units: dB

Event(s) generated (unless masked away):

When the Read_RSSI command has completed, a Command Complete event will be generated.

4.10 TESTING COMMANDS

The Testing commands are used to provide the ability to test various functionalities of the Bluetooth hardware. These commands provide the ability to arrange various conditions for testing. For the Testing Commands, the OGF is defined as 0x06

Command	Command Summary Description
Read_Loopback_Mode	The Read_Loopback_Mode will read the value for the setting of the Host Controllers Loopback Mode. The setting of the Loopback Mode will determine the path of information.
Write_Loopback_Mode	The Write_Loopback_Mode will write the value for the setting of the Host Controllers Loopback Mode. The setting of the Loopback Mode will determine the path of information.
Enable_Device_Under_Test_Mode	The Enable_Device_Under_Test_Mode command will allow the local Bluetooth module to enter test mode via LMP test commands. The Host issues this command when it wants the local device to be the DUT for the Testing scenarios as described in the Bluetooth Test Mode document.

4.10.1 Read_Loopback_Mode

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Loopback_Mode	0x0001		Status, Loopback_Mode

Description:

This command will read the value for the setting of the Host Controller's Loopback Mode. The setting of the Loopback Mode will determine the path of information. In Non-testing Mode operation, the Loopback Mode is set to Non-testing Mode and the path of the information is as specified by the Bluetooth specifications. In Local Loopback Mode, every Data Packet (ACL and SCO) and Command Packet that is sent from the Host to the Host Controller is sent back with no modifications by the Host Controller, as shown in Fig. 4.5 on page 697.

When the Bluetooth Host Controller enters Local Loopback Mode, it shall respond with four Connection Complete events, one for an ACL channel and three for SCO channels, so that the Host gets connection handles to use when sending ACL and SCO data. When in Local Loopback Mode the Host Controller loops back commands and data to the Host. The Loopback Command event is used to loop back commands that the Host sends to the Host Controller.

There are some commands that are not looped back in Local Loopback Mode: Reset, Set_Host_Controller_To_Host_Flow_Control, Host_Buffer_Size, Host_Number_Of_Completed_Packets, Read_Buffer_Size, Read_Loopback_Mode and Write_Loopback_Mode. These commands should be executed in the way they are normally executed. The commands Reset and Write_Loopback_Mode can be used to exit local loopback mode. If Write_Loopback_Mode is used to exit Local Loopback Mode, four Disconnection Complete events should be sent to the Host, corresponding to the Connection Complete events that were sent when entering Local Loopback Mode. Furthermore, no connections are allowed in Local Loopback mode. If there is a connection and there is an attempt to set the device to Local Loopback Mode, the attempt will be refused. When the device is in Local Loopback Mode, the Host Controller will refuse incoming connection attempts. This allows the Host Controller Transport Layer to be tested without any other variables.

If a device is set to Remote Loopback Mode, it will send back all data (ACL and SCO) that comes over the air, and it will only allow a maximum of one ACL connection and three SCO connections – and these should be all to the same remote device. If there already are connections to more than one remote device and there is an attempt to set the local device to Remote Loopback Mode, the attempt will be refused. See Fig. 4.6 on page 697 where the rightmost device is set to Remote Loopback Mode and the leftmost device is set to

Non-testing Mode. This allows the Bluetooth Air link to be tested without any other variables.

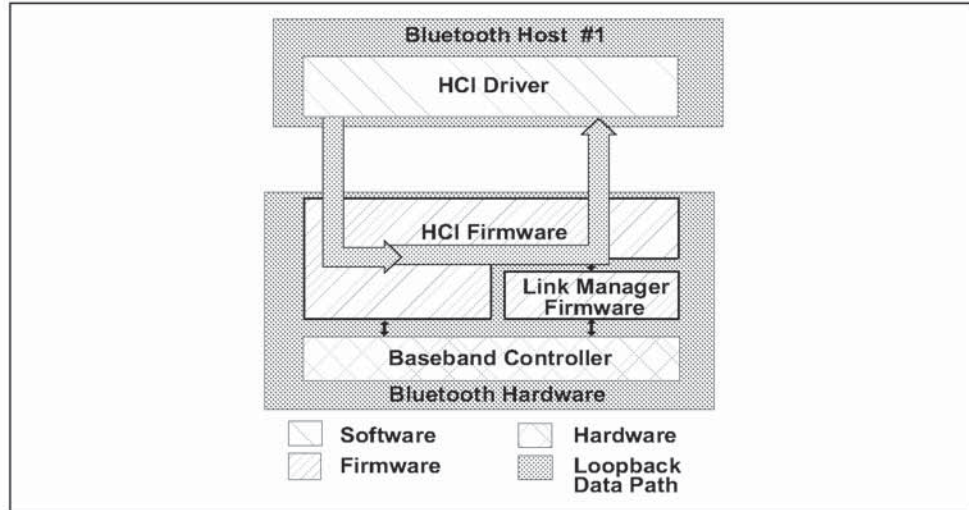


Figure 4.5: Local Loopback Mode

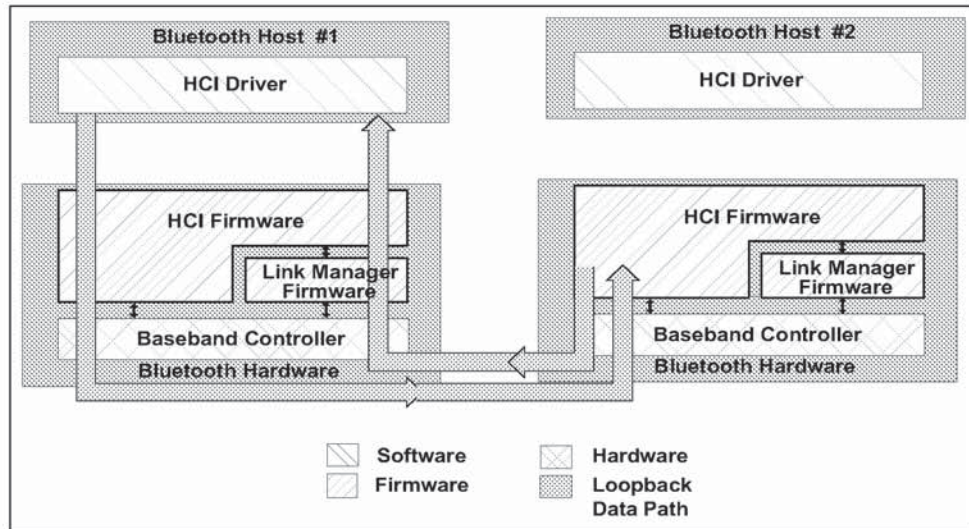


Figure 4.6: Remote Loopback Mode

Command Parameters:

None.

Return Parameters:*Status:**Size: 1 Byte*

Value	Parameter Description
0x00	Read_Loopback_Mode command succeeded.
0x01-0xFF	Read_Loopback_Mode command failed. See Table 2 on page 260 for list of Error Codes.

*Loopback_Mode:**Size: 1 Byte*

Value	Parameter Description
0x00	No Loopback mode enabled. Default.
0x01	Enable Local Loopback.
0x02	Enable Remote Loopback.
0x03-0xFF	Reserved for Future Use.

Event(s) generated (unless masked away):

When the Read_Loopback_Mode command has completed, a Command Complete event will be generated.

4.10.2 Write_Loopback_Mode

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Loopback_Mode	0x0002	Loopback_Mode	Status

Description:

This command will write the value for the setting of the Host Controller's Loopback Mode. The setting of the Loopback Mode will determine the path of information. In Non-testing Mode operation, the Loopback Mode is set to Non-testing Mode and the path of the information as specified by the Bluetooth specifications. In Local Loopback Mode, every Data Packet (ACL and SCO) and Command Packet that is sent from the Host to the Host Controller is sent back with no modifications by the Host Controller, as shown in Fig. 4.7 on page 700.

When the Bluetooth Host Controller enters Local Loopback Mode, it shall respond with four Connection Complete events, one for an ACL channel and three for SCO channels, so that the Host gets connection handles to use when sending ACL and SCO data. When in Local Loopback Mode, the Host Controller loops back commands and data to the Host. The Loopback Command event is used to loop back commands that the Host sends to the Host Controller.

There are some commands that are not looped back in Local Loopback Mode: Reset, Set_Host_Controller_To_Host_Flow_Control, Host_Buffer_Size, Host_Number_Of_Completed_Packets, Read_Buffer_Size, Read_Loopback_Mode and Write_Loopback_Mode. These commands should be executed in the way they are normally executed. The commands Reset and Write_Loopback_Mode can be used to exit local loopback mode.

If Write_Loopback_Mode is used to exit Local Loopback Mode, four Disconnection Complete events should be sent to the Host corresponding to the Connection Complete events that were sent when entering Local Loopback Mode. Furthermore, no connections are allowed in Local Loopback mode. If there is a connection, and there is an attempt to set the device to Local Loopback Mode, the attempt will be refused. When the device is in Local Loopback Mode, the Host Controller will refuse incoming connection attempts. This allows the Host Controller Transport Layer to be tested without any other variables.

If a device is set to Remote Loopback Mode, it will send back all data (ACL and SCO) that comes over the air. It will only allow a maximum of one ACL connection and three SCO connections, and these should all be to the same remote device. If there already are connections to more than one remote device and there is an attempt to set the local device to Remote Loopback Mode, the attempt will be refused.

See Fig. 4.8 on page 700, where the rightmost device is set to Remote Loopback Mode and the leftmost device is set to Non-testing Mode. This allows the Bluetooth Air link to be tested without any other variables.

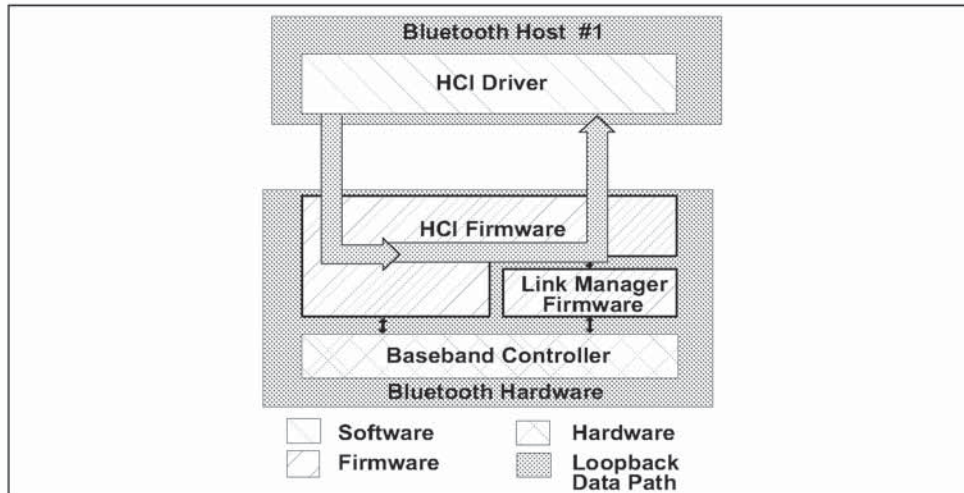


Figure 4.7: Local Loopback Mode

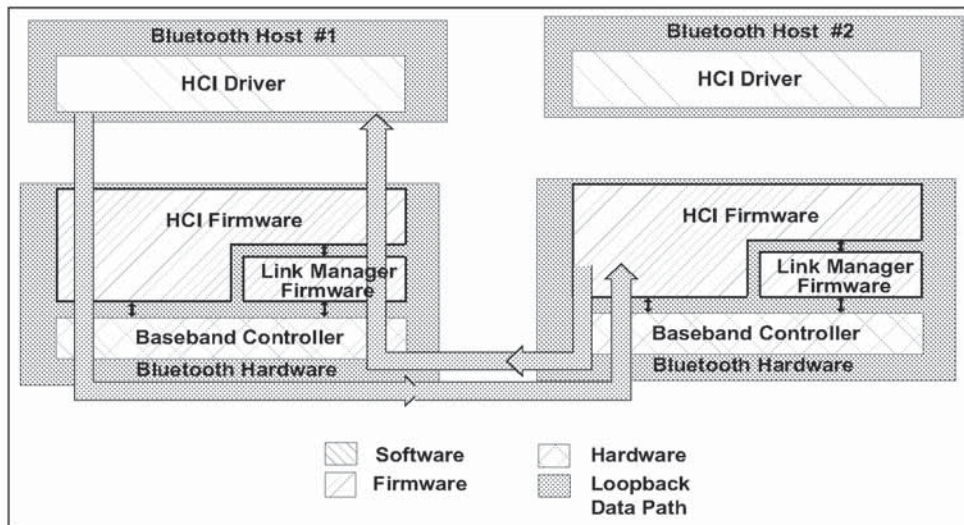


Figure 4.8: Remote Loopback Mode

Command Parameters:*Loopback_Mode:**Size: 1 Byte*

Value	Parameter Description
0x00	No Loopback mode enabled. Default.
0x01	Enable Local Loopback.
0x02	Enable Remote Loopback.
0x03-0xFF	Reserved for Future Use.

Return Parameters:*Status:**Size: 1 Byte*

Value	Parameter Description
0x00	Write_Loopback_Mode command succeeded.
0x01-0xFF	Write_Loopback_Mode command failed. See Table 6.1 on page 745 for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Loopback_Mode command has completed, a Command Complete event will be generated.

4.10.3 Enable_Device_Under_Test_Mode

Command	OCF	Command Parameters	Return Parameters
HCI_Enable_Device_Under_Test_Mode	0x0003		Status

Description:

The Enable_Device_Under_Test_Mode command will allow the local Bluetooth module to enter test mode via LMP test commands. For details see "Link Manager Protocol" on page 185. The Host issues this command when it wants the local device to be the DUT for the Testing scenarios as described in the "Bluetooth Test Mode" on page 803. When the Host Controller receives this command, it will complete the command with a Command Complete event. The Host Controller functions as normal until the remote tester issues the LMP test command to place the local device into Device Under Test mode. To disable and exit the Device Under Test Mode, the Host can issue the HCI_Reset command. This command prevents remote Bluetooth devices from causing the local Bluetooth device to enter test mode without first issuing this command.

Command Parameters:

None.

Return Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Enter_Device_Under_Test_Mode command succeeded.
0x01-0xFF	Enter_Device_Under_Test_Mode command failed. See Table 6.1 on page 745 for list of Error Codes.

Event(s) generated (unless masked away):

When the Enter_Device_Under_Test_Mode command has completed, a Command Complete event will be generated.

5 EVENTS

5.1 EVENT

In addition to the events listed below, event code 0xFF is reserved for the event code used for vendor-specific debug events, and event code 0xFE is reserved for Bluetooth Logo Testing.

Event	Event Summary Description
Inquiry Complete event	The Inquiry Complete event indicates that the Inquiry is finished.
Inquiry Result event	The Inquiry Result event indicates that a Bluetooth device or multiple Bluetooth devices have responded so far during the current Inquiry process.
Connection Complete event	The Connection Complete event indicates to both of the Hosts forming the connection that a new connection has been established.
Connection Request event	The Connection Request event is used to indicate that a new incoming connection is trying to be established.
Disconnection Complete event	The Disconnection Complete event occurs when a connection has been terminated.
Authentication Complete event	The Authentication Complete event occurs when authentication has been completed for the specified connection.
Remote Name Request Complete event	The Remote Name Request Complete event is used to indicate a remote name request has been completed. The Remote_Name event parameter is a UTF-8 encoded string with up to 248 bytes in length.
Encryption Change event	The Encryption Change event is used to indicate that the change in the encryption has been completed for the Connection Handle specified by the Connection_Handle event parameter.
Change Connection Link Key Complete event	The Change Connection Link Key Complete event is used to indicate that the change in the Link Key for the Connection Handle specified by the Connection_Handle event parameter had been completed.
Master Link Key Complete event	The Master Link Key Complete event is used to indicate that the change in the temporary Link Key or in the semi-permanent link keys on the Bluetooth master side has been completed.
Read Remote Supported Features Complete event	The Read Remote Supported Features Complete event is used to indicate the completion of the process of the Link Manager obtaining the supported features of the remote Bluetooth device specified by the Connection_Handle event parameter.

Table 5.1: List of Supported Events

Event	Event Summary Description
Read Remote Version Information Complete event	The Read Remote Version Information Complete event is used to indicate the completion of the process of the Link Manager obtaining the version information of the remote Bluetooth device specified by the Connection_Handle event parameter.
QoS Setup Complete event	The QoS Setup Complete event is used to indicate the completion of the process of the Link Manager setting up QoS with the remote Bluetooth device specified by the Connection_Handle event parameter.
Command Complete event	The Command Complete event is used by the Host Controller to pass the return status of a command and the other event parameters for each HCI Command.
Command Status event	The Command Status event is used to indicate that the command described by the Command_Opcode parameter has been received and the Host Controller is currently performing the task for this command.
Hardware Error event	The Error event is used to indicate some type of hardware failure for the Bluetooth device.
Flush Occurred event	The Flush Occurred event is used to indicate that, for the specified Connection Handle, the current user data to be transmitted has been removed.
Role Change event	The Role Change event is used to indicate that the current Bluetooth role related to the particular connection has been changed.
Number Of Completed Packets event	The Number Of Completed Packets event is used by the Host Controller to indicate to the Host how many HCI Data Packets have been completed for each Connection Handle since the previous Number Of Completed Packets event was sent.
Mode Change event	The Mode Change event is used to indicate when the device associated with the Connection Handle changes between Active, Hold, Sniff and Park mode.
Return Link Keys event	The Return Link Keys event is used to return stored link keys after a Read_Stored_Link_Key command is used.
PIN Code Request event	The PIN Code Request event is used to indicate that a PIN code is required to create a new link key for a connection.
Link Key Request event	The Link Key Request event is used to indicate that a Link Key is required for the connection with the device specified in BD_ADDR.
Link Key Notification event	The Link Key Notification event is used to indicate to the Host that a new Link Key has been created for the connection with the device specified in BD_ADDR.
Loopback Command event	The Loopback Command event is used to loop back most commands that the Host sends to the Host Controller.

Table 5.1: List of Supported Events

Event	Event Summary Description
Data Buffer Overflow event	The Data Buffer Overflow event is used to indicate that the Host Controller's data buffers have overflowed, because the Host has sent more packets than allowed.
Max Slots Change event	This event is used to notify the Host about the LMP_Max_Slots parameter when the value of this parameter changes.
Read Clock Offset Complete event	The Read Clock Offset Complete event is used to indicate the completion of the process of the LM obtaining the Clock offset information.
Connection Packet Type Changed event	The Connection Packet Type Changed event is used to indicate the completion of the process of the Link Manager changing the Packet Types used for the specified Connection_Handle.
QoS Violation event	The QoS Violation event is used to indicate the Link Manager is unable to provide the current QoS requirement for the Connection Handle.
Page Scan Mode Change event	The Page Scan Mode Change event indicates that the connected remote Bluetooth device with the specified Connection_Handle has successfully changed the Page_Scan_Mode.
Page Scan Repetition Mode Change event	The Page Scan Repetition Mode Change event indicates that the connected remote Bluetooth device with the specified Connection_Handle has successfully changed the Page_Scan_Repetition_Mode (SR).

Table 5.1: List of Supported Events

5.2 POSSIBLE EVENTS

The events provide a method to return parameters and data associated for each event.

5.2.1 Inquiry Complete event

Event	Event Code	Event Parameters
Inquiry Complete	0x01	Status, Num_Responses

Description:

The Inquiry Complete event indicates that the Inquiry is finished. This event contains a status parameter, which is used to indicate if the Inquiry completed successfully or if the Inquiry was not completed. In addition, the Num_Responses parameter contains the number of Bluetooth devices, which responded during the latest inquiry.

Event Parameters:

*Status:**Size: 1 Byte*

Value	Parameter Description
0x00	Inquiry command completed successfully.
0x01-0xFF	Inquiry command failed. See Table 6.1 on page 745 for list of Error Codes.

*Num_Responses:**Size: 1 Byte*

Value	Parameter Description
0xXX	Number of responses from the Inquiry.

5.2.2 Inquiry Result event

Event	Event Code	Event Parameters
Inquiry Result	0x02	Num_Responses, BD_ADDR[i], Page_Scan_Repetition_Mode[i], Page_Scan_Period_Mode[i], Page_Scan_Mode[i], Class_of_Device[i] Clock_Offset[i]

Description:

The Inquiry Result event indicates that a Bluetooth device or multiple Bluetooth devices have responded so far during the current Inquiry process. This event will be sent from the Host Controller to the Host as soon as an Inquiry Response from a remote device is received if the remote device supports only mandatory paging scheme. The Host Controller may queue these Inquiry Responses and send multiple Bluetooth devices information in one Inquiry Result event. The event can be used to return one or more Inquiry responses in one event. This event contains the BD_ADDR, Page_Scan_Repetition_Mode, Page_Scan_Period_Mode, Page_Scan_Mode, Clock_Offset, and the Class of Device for each Bluetooth device that responded to the latest inquiry.

Event Parameters:

Num_Responses:

Size: 1 Byte

Value	Parameter Description
0xXX	Number of responses from the Inquiry.

BD_ADDR[i]:

*Size: 6 Bytes * Num_Responses*

Value	Parameter Description
0XXXXXXXXXX XX	BD_ADDR for each device which responded.

Page_Scan_Repetition_Mode[i]:

Size: 1 Byte Num_Responses*

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03 – 0xFF	Reserved

Page_Scan_Period_Mode[i]:

Size: 1 Byte Num_Responses*

Value	Parameter Description
0x00	P0
0x01	P1
0x02	P2
0x03 – 0xFF	Reserved

Page_Scan_Mode[i]:

Size: 1 Byte Num_Responses*

Value	Parameter Description
0x00	Mandatory Page Scan Mode
0x01	Optional Page Scan Mode I
0x02	Optional Page Scan Mode II
0x03	Optional Page Scan Mode III
0x04 – 0xFF	Reserved

Class_of_Device[i]:

*Size: 3 Bytes * Num_Responses*

Value	Parameter Description
0xXXXXXX	Class of Device for the device

Clock_Offset[i]:

*Size: 2 Bytes * Num_Responses*

Bit format	Parameter Description
Bit 14.0	Bit 16.2 of CLKslave-CLKmaster.
Bit 15	Reserved

5.2.3 Connection Complete event

Event	Event Code	Event Parameters
Connection Complete	0x03	Status, Connection_Handle, BD_ADDR, Link_Type, Encryption_Mode

Description:

The Connection Complete event indicates to both of the Hosts forming the connection that a new connection has been established. This event also indicates to the Host, which issued the Create Connection, Add_SCO_Connection, or Accept_Connection_Request or Reject_Connection_Request command and then received a Command Status event, if the issued command failed or was successful.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Connection successfully completed.
0x01-0xFF	Connection failed to Complete. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection between to Bluetooth devices. The Connection Handle is used as an identifier for transmitting and receiving voice or data. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

BD_ADDR:

Size: 6 Bytes

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR of the other connected Device forming the connection.

*Link_Type:**Size: 1 Byte*

Value	Parameter Description
0x00	SCO connection (Voice Channels).
0x01	ACL connection (Data Channels).
0x02-0xFF	Reserved for Future Use.

*Encryption_Mode:**Size: 1 Byte*

Value	Parameter Description
0x00	Encryption disabled.
0x01	Encryption only for point-to-point packets.
0x02	Encryption for both point-to-point and broadcast packets.
0x03-0xFF	Reserved.

5.2.4 Connection Request event

Event	Event Code	Event Parameters
Connection Request	0x04	BD_ADDR, Class_of_Device, Link_Type

Description:

The Connection Request event is used to indicate that a new incoming connection is trying to be established. The connection may either be accepted or rejected. If this event is masked away and there is an incoming connection attempt and the Host Controller is not set to auto-accept this connection attempt, the Host Controller will automatically refuse the connection attempt. When the Host receives this event, it should respond with either an Accept_Connection_Request or Reject_Connection_Request command before the timer Conn_Accept_Timeout expires.

Event Parameters:

BD_ADDR: *Size: 6 Bytes*

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR of the device that requests the connection.

Class_of_Device: *Size: 3 Bytes*

Value	Parameter Description
0XXXXXX	Class of Device for the device, which request the connection.

Link_Type: *Size: 1 Byte*

Value	Parameter Description
0x00	SCO connection requested (Voice Channels).
0x01	ACL connection requested (Data Channels).
0x02-0xFF	Reserved for Future Use.

5.2.5 Disconnection Complete event

Event	Event Code	Event Parameters
Disconnection Complete	0x05	Status, Connection_Handle, Reason

Description:

The Disconnection Complete event occurs when a connection is terminated. The status parameter indicates if the disconnection was successful or not. The reason parameter indicates the reason for the disconnection if the disconnection was successful. If the disconnection was not successful, the value of the reason parameter can be ignored by the Host. For example, this can be the case if the Host has issued the Disconnect command and there was a parameter error, or the command was not presently allowed, or a connection handle that didn't correspond to a connection was given.

Note: When a physical link fails, one Disconnection Complete event will be returned for each logical channel on the physical link with the corresponding Connection handle as a parameter.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Disconnection has occurred.
0x01-0xFF	Disconnection failed to complete. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle which was disconnected. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Reason:

Size: 1 Byte

Value	Parameter Description
0x08, 0x13-0x16, 0x1A	Connection Timeout (0x08), Other End Terminated Connection error codes (0x13-0x15), Connection Terminated by Local Host (0x16), and Unsupported Remote Feature error code (0x1A). See Table 6.1 on page 745 for list of Error Codes.

5.2.6 Authentication Complete event

Event	Event Code	Event Parameters
Authentication Complete	0x06	Status, Connection_Handle

Description:

The Authentication Complete event occurs when authentication has been completed for the specified connection. The Connection_Handle must be a Connection_Handle for an ACL connection.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Authentication Request successfully completed.
0x01-0xFF	Authentication Request failed to complete. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xFFFF	Connection Handle for which Authentication has been performed. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

5.2.7 Remote Name Request Complete event

Event	Event Code	Event Parameters
Remote Name Request Complete	0x07	Status, BD_ADDR, Remote_Name

Description:

The Remote Name Request Complete event is used to indicate that a remote name request has been completed. The Remote_Name event parameter is a UTF-8 encoded string with up to 248 bytes in length. The Remote_Name event parameter will be null-terminated (0x00) if the UTF-8 encoded string is less than 248 bytes. The BD_ADDR event parameter is used to identify which device the user-friendly name was obtained from.

Note: the Remote_Name Parameter is received starting with the first byte of the name. This is an exception to the Little Endian order format for transmitting multi-byte parameters.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Remote_Name_Request command succeeded.
0x01-0xFF	Remote_Name_Request command failed. See Table 6.1 on page 745 for list of Error Codes.

BD_ADDR:

Size: 6 Bytes

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR for the device whose name was requested.

Remote_Name:

Size: 248 Bytes

Value	Parameter Description
Name[248]	A UTF-8 encoded user-friendly descriptive name for the remote device. A UTF-8 encoded name can be up to 248 bytes in length. If it is shorter than 248 bytes, the end is indicated by a NULL byte (0x00).

5.2.8 Encryption Change event

Event	Event Code	Event Parameters
Encryption Change	0x08	Status, Connection_Handle, Encryption_Enable

Description:

The Encryption Change event is used to indicate that the change in the encryption has been completed for the Connection Handle specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. The Encryption_Enable event parameter specifies the new Encryption Enable for the Connection Handle specified by the Connection_Handle event parameter. This event will occur on both devices to notify both Hosts when Encryption has changed for the specified Connection Handle between two devices.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Encryption Change has occurred.
0x01-0xFF	Encryption Change failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle for which the link layer encryption has been enabled/ disabled for all Connection Handles with the same Bluetooth device endpoint as the specified Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Encryption_Enable:

Size: 1 Byte

Value	Parameter Description
0x00	Link Level Encryption is OFF.
0x01	Link Level Encryption is ON.

5.2.9 Change Connection Link Key Complete event

Event	Event Code	Event Parameters
Change Connection Link Key Complete	0x09	Status, Connection_Handle

Description:

The Change Connection Link Key Complete event is used to indicate that the change in the Link Key for the Connection Handle specified by the Connection_Handle event parameter has been completed.

The Connection_Handle will be a Connection_Handle for an ACL connection. The Change Connection Link Key Complete event is sent only to the Host which issued the Change_Connection_Link_Key command.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Change_Connection_Link_Key command succeeded.
0x01-0xFF	Change_Connection_Link_Key command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle which the Link Key has been change for all Connection Handles with the same Bluetooth device end point as the specified Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

5.2.10 Master Link Key Complete event

Event	Event Code	Event Parameters
Master Link Key Complete	0x0A	Status, Connection_Handle, Key_Flag

Description:

The Master Link Key Complete event is used to indicate that the Link Key managed by the master of the piconet has been changed. The Connection_Handle will be a Connection_Handle for an ACL connection. The link key used for the connection will be the temporary link key of the master device or the semi-permanent link key indicated by the Key_Flag. The Key_Flag event parameter is used to indicate which Link Key (temporary link key of the Master, or the semi-permanent link keys) is now being used in the piconet.

Note: for a master, the change from a semi-permanent Link Key to temporary Link Key will affect all Connection Handles related to the piconet. For a slave, this change affects only this particular connection handle. A temporary link key must be used when both broadcast and point-to-point traffic shall be encrypted.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Master_Link_Key command succeeded.
0x01-0xFF	Master_Link_Key command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle for which the Link Key has been changed for all devices in the same piconet. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Key_Flag:

Size: 1 Byte

Value	Parameter Description
0x00	Using Semi-permanent Link Key.
0x01	Using Temporary Link Key.

5.2.11 Read Remote Supported Features Complete event

Event	Event Code	Event Parameters
Read Remote Supported Features Complete	0x0B	Status, Connection_Handle, LMP_Features

Description:

The Read Remote Supported Features Complete event is used to indicate the completion of the process of the Link Manager obtaining the supported features of the remote Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. The event parameters include a list of LMP features. For details see "Link Manager Protocol" on page 185.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Remote_Supported_Features command succeeded.
0x01-0xFF	Read_Remote_Supported_Features command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle which is used for the Read_Remote_Supported_Features command. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

LMP_Features:

Size: 8 Bytes

Value	Parameter Description
0XXXXXXXX XXXXXXXX	Bit Mask List of LMP features. See "Link Manager Protocol" on page 185.

5.2.12 Read Remote Version Information Complete event

Event	Event Code	Event Parameters
Read Remote Version Information Complete	0x0C	Status, Connection_Handle, LMP_Version, Manufacturer_Name, LMP_Subversion

Description:

The Read Remote Version Information Complete event is used to indicate the completion of the process of the Link Manager obtaining the version information of the remote Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. The LMP_Version event parameter defines the major hardware version of the Bluetooth hardware. This event parameter only changes when new versions of the Bluetooth hardware are produced for new Bluetooth SIG specifications; it is controlled by the SIG. The Manufacturer_Name event parameter indicates the manufacturer of the remote Bluetooth module. The LMP_Subversion event parameter should be controlled by the manufacturer and should be changed as needed. The LMP_Subversion event parameter defines the various revisions that each version of the Bluetooth hardware will go through as design processes change and errors are fixed. This allows the software to determine what Bluetooth hardware is being used and, if necessary, to work around various bugs in the hardware.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Remote_Version_Information command succeeded.
0x01-0xFF	Read_Remote_Version_Information command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xFFFF	Connection Handle which is used for the Read_Remote_Version_Information command. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

LMP_Version:

Size: 1 Byte

Value	Parameter Description
0xXX	Version of the Current LMP in the remote Bluetooth Hardware, see Table 5.2 on page 231 in the Link Manager Protocol for assigned values (VersNr).

Manufacturer_Name:

Size: 2 Bytes

Value	Parameter Description
0xXXXX	Manufacturer Name of the remote Bluetooth Hardware, see Table 5.2 on page 231 in the Link Manager Protocol for assigned values (Compld).

LMP_Subversion:

Size: 2 Bytes

Value	Parameter Description
0xXXXX	Subversion of the Current LMP in the remote Bluetooth Hardware, see Table 5.2 on page 231 in the Link Manager Protocol for assigned values (SubVersNr).

5.2.13 QoS Setup Complete event

Event	Event Code	Event Parameters
QoS Setup Complete	0x0D	Status, Connection_Handle, Flags, Service_Type, Token_Rate, Peak_Bandwidth, Latency, Delay_Variation

Description:

The QoS Setup Complete event is used to indicate the completion of the process of the Link Manager setting up QoS with the remote Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. For more detail see "Logical Link Control and Adaptation Protocol Specification" on page 245.

Event Parameters:

Status: *Size: 1 Byte*

Value	Parameter Description
0x00	QoS_Setup command succeeded.
0x01-0xFF	QoS_Setup command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle which is used for the QoS_Setup command. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Flags: *Size: 1 Byte*

Value	Parameter Description
0x00 – 0xFF	Reserved for Future Use.

Service_Type:

Size: 1 Byte

Value	Parameter Description
0x00	No Traffic Available.
0x01	Best Effort Available.
0x02	Guaranteed Available.
0x03-0xFF	Reserved for Future Use.

Token_Rate:

Size: 4 Bytes

Value	Parameter Description
0XXXXXXXX	Available Token Rate, in bytes per second.

Peak_Bandwidth:

Size: 4 Bytes

Value	Parameter Description
0XXXXXXXX	Available Peak Bandwidth, in bytes per second.

Latency:

Size: 4 Bytes

Value	Parameter Description
0XXXXXXXX	Available Latency, in microseconds.

Delay_Variation:

Size: 4 Bytes

Value	Parameter Description
0XXXXXXXX	Available Delay Variation, in microseconds.

5.2.14 Command Complete event

Event	Event Code	Event Parameters
Command Complete	0x0E	Num_HCI_Command_Packets, Command_Opcode, Return_Parameters

Description:

The Command Complete event is used by the Host Controller for most commands to transmit return status of a command and the other event parameters that are specified for the issued HCI command.

The Num_HCI_Command_Packets event parameter allows the Host Controller to indicate the number of HCI command packets the Host can send to the Host Controller. If the Host Controller requires the Host to stop sending commands, the Num_HCI_Command_Packets event parameter will be set to zero. To indicate to the Host that the Host Controller is ready to receive HCI command packets, the Host Controller generates a Command Complete event with the Command_Opcode 0x0000, and the Num_HCI_Command_Packets event parameter is set to 1 or more. Command_Opcode, 0x0000 is a NOP (No Operation), and can be used to change the number of outstanding HCI command packets that the Host can send before waiting. See each command for the parameters that are returned by this event.

Event Parameters:

Num_HCI_Command_Packets:

Size: 1 Byte

Value	Parameter Description
N = 0xXX	The Number of HCI command packets which are allowed to be sent to the Host Controller from the Host. Range for N: 0 – 255

Command_Opcode:

Size: 2 Bytes

Value	Parameter Description
0xXXXX	Opcode of the command which caused this event.

Return_Parameter(s):

Size: Depends on Command

Value	Parameter Description
0xXX	This is the return parameter(s) for the command specified in the Command_Opcode event parameter. See each command's definition for the list of return parameters associated with that command.

5.2.15 Command Status event

Event	Event Code	Event Parameters
Command Status	0x0F	Status, Num_HCI_Command_Packets, Command_Opcode

Description:

The Command Status event is used to indicate that the command described by the Command_Opcode parameter has been received, and that the Host Controller is currently performing the task for this command. This event is needed to provide mechanisms for asynchronous operation, which makes it possible to prevent the Host from waiting for a command to finish. If the command can not begin to execute (a parameter error may have occurred, or the command may currently not be allowed), the Status event parameter will contain the corresponding error code, and no complete event will follow since the command was not started. The Num_HCI_Command_Packets event parameter allows the Host Controller to indicate the number of HCI command packets the Host can send to the Host Controller. If the Host Controller requires the Host to stop sending commands, the Num_HCI_Command_Packets event parameter will be set to zero. To indicate to the Host that the Host Controller is ready to receive HCI command packets, the Host Controller generates a Command Status event with Status 0x00 and Command_Opcode 0x0000, and the Num_HCI_Command_Packets event parameter is set to 1 or more. Command_Opcode, 0x0000 is a NOP (No Operation) and can be used to change the number of outstanding HCI command packets that the Host can send before waiting.

Event Parameters:

Status: *Size: 1 Byte*

Value	Parameter Description
0x00	Command currently in pending.
0x01-0xFF	Command failed. See Table 6.1 on page 745 for list of Error Codes.

Num_HCI_Command_Packets: *Size: 1 Byte*

Value	Parameter Description
N = 0xXX	The Number of HCI command packets which are allowed to be sent to the Host Controller from the Host. Range for N: 0 – 255

Command_Opcode: *Size: 2 Bytes*

Value	Parameter Description
0xXXXX	Opcode of the command which caused this event and is pending completion.

5.2.16 Hardware Error event

Event	Event Code	Event Parameters
Hardware Error	0x10	Hardware_Code

Description:

The Hardware Error event is used to indicate some type of hardware failure for the Bluetooth device. This event is used to notify the Host that a hardware failure has occurred in the Bluetooth module.

Event Parameters:

Hardware_Code:

Size: 1 Byte

Value	Parameter Description
0x00	These Hardware_Codes will be implementation-specific, and will be assigned to indicate various hardware problems.

5.2.17 Flush Occurred event

Event	Event Code	Event Parameters
Flush Occurred	0x11	Connection_Handle

Description:

The Flush Occurred event is used to indicate that, for the specified Connection Handle, the current user data to be transmitted has been removed. The Connection_Handle will be a Connection_Handle for an ACL connection. This could result from the flush command, or be due to the automatic flush. Multiple blocks of an L2CAP packet could have been pending in the Host Controller. If one baseband packet part of an L2CAP packet is flushed, then the rest of the HCI data packets for the L2CAP packet must also be flushed.

Event Parameters:*Connection_Handle:**Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle which was flushed. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

5.2.18 Role Change event

Event	Event Code	Event Parameters
Role Change	0x12	Status, BD_ADDR, New_Role

Description:

The Role Change event is used to indicate that the current Bluetooth role related to the particular connection has changed. This event only occurs when both the remote and local Bluetooth devices have completed their role change for the Bluetooth device associated with the BD_ADDR event parameter. This event allows both affected Hosts to be notified when the Role has been changed.

Event Parameters:

Status: *Size: 1 Byte*

Value	Parameter Description
0x00	Role change has occurred.
0x01-0xFF	Role change failed. See Table 6.1 on page 745 for list of Error Codes.

BD_ADDR: *Size: 6 Bytes*

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR of the Device for which a role change has completed.

New_Role: *Size: 1 Byte*

Value	Parameter Description
0x00	Currently the Master for specified BD_ADDR.
0x01	Currently the Slave for specified BD_ADDR.

5.2.19 Number Of Completed Packets event

Event	Event Code	Event Parameters
Number Of Completed Packets	0x13	Number_of_Handles, Connection_Handle[i], HC_Num_Of_Completed_Packets[i]

Description:

The Number Of Completed Packets event is used by the Host Controller to indicate to the Host how many HCI Data Packets have been completed (transmitted or flushed) for each Connection Handle since the previous Number Of Completed Packets event was sent to the Host. This means that the corresponding buffer space has been freed in the Host Controller. Based on this information, and the HC_Total_Num_ACL_Data_Packets and HC_Total_Num_SCO_Data_Packets return parameter of the Read_Buffer_Size command, the Host can determine for which Connection Handles the following HCI Data Packets should be sent to the Host Controller. The Number Of Completed Packets event must not be sent before the corresponding Connection Complete event. While the Host Controller has HCI data packets in its buffer, it must keep sending the Number Of Completed Packets event to the Host at least periodically, until it finally reports that all the pending ACL Data Packets have been transmitted or flushed. The rate with which this event is sent is manufacturer specific.

Note that Number Of Completed Packets events will not report on SCO connection handles if SCO Flow Control is disabled. (See Read/Write_SCO_Flow_Control_Enable on page 658 and page 659.)

Event Parameters:

Number_of_Handles:

Size: 1 Byte

Value	Parameter Description
0xXX	The number of Connection Handles and Num_HCI_Data_Packets parameters pairs contained in this event. Range: 0-255

*Connection_Handle[i]: Size: Number_of_Handles * 2 Bytes(12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

HC_Num_Of_Completed_Packets [i]: *Size: Number_of_Handles * 2 Bytes*

Value	Parameter Description
N = 0xXXXX	The number of HCI Data Packets that have been completed (transmitted or flushed) for the associated Connection Handle since the previous time the event was returned. Range for N: 0x0000-0xFFFF

5.2.20 Mode Change event

Event	Event Code	Event Parameters
Mode Change	0x14	Status, Connection_Handle, Current_Mode, Interval

Description:

The Mode Change event is used to indicate when the device associated with the Connection Handle changes between Active, Hold, Sniff and Park mode. The Connection_Handle will be a Connection_Handle for an ACL connection. The Connection_Handle event parameter is used to indicate which connection the Mode Change event is for. The Current_Mode event parameter is used to indicate which state the connection is currently in. The Interval parameter is used to specify a time amount specific to each state. Each Host Controller that is associated with the Connection Handle which has changed Modes will send the Mode Change event to its Host.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	A Mode Change has occurred.
0x01-0xFF	Hold_Mode, Sniff_Mode, Exit_Sniff_Mode, Park_Mode, or Exit_Park_Mode command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes(12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Current_Mode:

Size: 1 Byte

Value	Parameter Description
0x00	Active Mode.
0x01	Hold Mode.
0x02	Sniff Mode.
0x03	Park Mode.
0x04-0xFF	Reserved for future use.

Interval:

Size: 2 Bytes

Value	Parameter Description
N = 0xXXXX	<p>Hold: Number of Baseband slots to wait in Hold Mode. Hold Interval = $N * 0.625$ msec (1 Baseband slot) Range for N: 0x0000-0xFFFF Time Range: 0-40.9 sec</p> <p>Sniff: Number of Baseband slots between sniff intervals. Time between sniff intervals = 0.625 msec (1 Baseband slot) Range for N: 0x0000-0xFFFF Time Range: 0-40.9 sec</p> <p>Park: Number of Baseband slots between consecutive beacons. Interval Length = $N * 0.625$ msec (1 Baseband slot) Range for N: 0x0000-0xFFFF Time Range: 0-40.9 Seconds</p>

5.2.21 Return Link Keys event

Event	Event Code	Event Parameters
Return Link Keys	0x15	Num_Keys, BD_ADDR [i], Link_Key[i]

Description:

The Return Link Keys event is used by the Host Controller to send the Host one or more stored Link Keys. Zero or more instances of this event will occur after the Read_Stored_Link_Key command. When there are no link keys stored, no Return Link Keys events will be returned. When there are link keys stored, the number of link keys returned in each Return Link Keys event is implementation specific.

Event Parameters:

Num_Keys: *Size: 1 Byte*

Value	Parameter Description
0xXX	Number of Link Keys contained in this event. Range: 0x01 – 0xFF

BD_ADDR [i]: *Size: 6 Bytes * Num_Keys*

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR for the associated Link Key.

Link_Key[i]: *Size: 16 Bytes * Num_Keys*

Value	Parameter Description
0XXXXXXXXXXXXX XXXXXXXXXXXXX XXXXXXXXXXXXX	Link Key for the associated BD_ADDR.

5.2.22 PIN Code Request event

Event	Event Code	Event Parameters
PIN Code Request	0x16	BD_ADDR

Description:

The PIN Code Request event is used to indicate that a PIN code is required to create a new link key. The Host must respond using either the PIN Code Request Reply or the PIN Code Request Negative Reply command, depending on whether the Host can provide the Host Controller with a PIN code or not. Note: If the PIN Code Request event is masked away, then the Host Controller will assume that the Host has no PIN Code.

When the Host Controller generates a PIN Code Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of a Create_Connection or Authentication_Requested command from the remote Host), the local Host must respond with either a PIN_Code_Request_Reply or PIN_Code_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 185.)

Event Parameters:

BD_ADDR:

Size: 6 Bytes

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR of the Device which a new link key is being created for.

5.2.23 Link Key Request event

Event	Event Code	Event Parameters
Link Key Request	0x17	BD_ADDR

Description:

The Link Key Request event is used to indicate that a Link Key is required for the connection with the device specified in BD_ADDR. If the Host has the requested stored Link Key, then the Host will pass the requested Key to the Host Controller using the Link_Key_Request_Reply Command. If the Host does not have the requested stored Link Key, then the Host will use the Link_Key_Request_Negative_Reply Command to indicate to the Host Controller that the Host does not have the requested key.

Note: If the Link Key Request event is masked away, then the Host Controller will assume that the Host has no additional link keys.

When the Host Controller generates a Link Key Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of a Create_Connection or Authentication_Requested command from the remote Host), the local Host must respond with either a Link_Key_Request_Reply or Link_Key_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 185.)

Event Parameters:

BD_ADDR:

Size: 6 Bytes

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR of the Device which a stored link key is being requested.

5.2.24 Link Key Notification event

Event	Event Code	Event Parameters
Link Key Notification	0x18	BD_ADDR, Link_Key

Description:

The Link Key Notification event is used to indicate to the Host that a new Link Key has been created for the connection with the device specified in BD_ADDR. The Host can save this new Link Key in its own storage for future use. Also, the Host can decided to store the Link Key in the Host Controller's Link Key Storage by using the Write_Stored_Link_Key command.

Event Parameters:

BD_ADDR:

Size: 6 Bytes

Value	Parameter Description
0XXXXXXXXXXXXX	BD_ADDR of the Device for which the new link key has been generated.

Link_Key:

Size: 16 Bytes

Value	Parameter Description
XXXXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX	Link Key for the associated BD_ADDR.

5.2.25 Loopback Command event

Event	Event Code	Event Parameters
Loopback Command	0x19	HCI_Command_Packet

Description:

When in Local Loopback mode, the Host Controller loops back commands and data to the Host. The Loopback Command event is used to loop back all commands that the Host sends to the Host Controller with some exceptions. See Section 4.10.1, "Read_Loopback_Mode," on page 696 for a description of which commands that are not looped back. The HCI_Command_Packet event parameter contains the entire HCI Command Packet including the header. Note: the event packet is limited to a maximum of 255 bytes in the payload; since an HCI Command Packet has 3 bytes of header data, only the first 252 bytes of the command parameters will be returned.

Event Parameters:

HCI_Command_Packet:

Size: Depends on Command

Value	Parameter Description
0XXXXXXXX	HCI Command Packet, including header.

5.2.26 Data Buffer Overflow event

Event	Event Code	Event Parameters
Data Buffer Overflow	0x1A	Link_Type

Description:

This event is used to indicate that the Host Controller's data buffers have been overflowed. This can occur if the Host has sent more packets than allowed. The Link_Type parameter is used to indicate that the overflow was caused by ACL or SCO data.

Event Parameters:

Link_Type:

Size: 1 Byte

Value	Parameter Description
0x00	SCO Buffer Overflow (Voice Channels).
0x01	ACL Buffer Overflow (Data Channels).
0x02-0xFF	Reserved for Future Use.

5.2.27 Max Slots Change event

Event	Event Code	Event Parameters
Max Slots Change	0x1B	Connection_Handle, LMP_Max_Slots

Description:

This event is used to notify the Host about the LMP_Max_Slots parameter when the value of this parameter changes. It will be sent each time the value of the LMP_Max_Slots parameter changes, as long as there is at least one connection to another device. The Connection_Handle will be a Connection_Handle for an ACL connection.

Event Parameters:

Connection_Handle: *Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

LMP_Max_Slots: *Size: 1 byte*

Value	Parameter Description
0xXX	Maximum number of slots allowed to use for baseband packets, see "Link Manager Protocol" on page 185.

5.2.28 Read Clock Offset Complete event

Event	Event Code	Event Parameters
Read Clock Offset Complete	0x1C	Status, Connection_Handle, Clock_Offset

Description:

The Read Clock Offset Complete event is used to indicate the completion of the process of the Link Manager obtaining the Clock Offset information of the Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Read_Clock_Offset command succeeded.
0x01-0xFF	Read_Clock_Offset command failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 bits meaningful)

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Clock Offset parameter is returned. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Clock_Offset:

Size: 2 Bytes

Bit format	Parameter Description
Bit 14.0	Bit 16.2 of CLKslave-CLKmaster.
Bit 15	Reserved.

5.2.29 Connection Packet Type Changed event

Event	Event Code	Event Parameters
Connection Packet Type Changed	0x1D	Status, Connection_Handle, Packet_Type

Description:

The Connection Packet Type Changed event is used to indicate that the process has completed of the Link Manager changing which packet types can be used for the connection. This allows current connections to be dynamically modified to support different types of user data. The Packet_Type event parameter specifies which packet types the Link Manager can use for the connection identified by the Connection_Handle event parameter for sending L2CAP data or voice. The Packet_Type event parameter does not decide which packet types the LM is allowed to use for sending LMP PDUs.

Event Parameters:

Status:

Size: 1 Byte

Value	Parameter Description
0x00	Connection Packet Type changed successfully.
0x01-0xFF	Connection Packet Type Changed failed. See Table 6.1 on page 745 for list of Error Codes.

Connection_Handle:

Size: 2 Bytes (12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Packet_Type:

Size: 2 Bytes

For ACL_Link_Type

Value	Parameter Description
0x0001	Reserved for future use.
0x0002	Reserved for future use.
0x0004	Reserved for future use.
0x0008	DM1
0x0010	DH1
0x0020	Reserved for future use.
0x0040	Reserved for future use.
0x0080	Reserved for future use.

Value	Parameter Description
0x0100	Reserved for future use.
0x0200	Reserved for future use.
0x0400	DM3
0x0800	DH3
0x1000	Reserved for future use.
0x2000	Reserved for future use.
0x4000	DM5
0x8000	DH5

For SCO_Link_Type

Value	Parameter Description
0x0001	Reserved for future use.
0x0002	Reserved for future use.
0x0004	Reserved for future use.
0x0008	Reserved for future use.
0x0010	Reserved for future use.
0x0020	HV1
0x0040	HV2
0x0080	HV3
0x0100	Reserved for future use.
0x0200	Reserved for future use.
0x0400	Reserved for future use.
0x0800	Reserved for future use.
0x1000	Reserved for future use.
0x2000	Reserved for future use.
0x4000	Reserved for future use.
0x8000	Reserved for future use.

5.2.30 QoS Violation event

Event	Event Code	Event Parameters
QoS Violation	0x1E	Connection_Handle

Description:

The QoS Violation event is used to indicate the Link Manager is unable to provide the current QoS requirement for the Connection Handle. This event indicates that the Link Manager is unable to provide one or more of the agreed QoS parameters. The Host chooses what action should be done. The Host can reissue QoS_Setup command to renegotiate the QoS setting for Connection Handle. The Connection_Handle will be a Connection_Handle for an ACL connection.

Event Parameters:*Connection_Handle:**Size: 2 Bytes (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle that the LM is unable to provide the current QoS requested for. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

5.2.31 Page Scan Mode Change event

Event	Event Code	Event Parameters
Page Scan Mode Change	0x1F	BD_ADDR, Page_Scan_Mode

Description:

The Page Scan Mode Change event indicates that the remote Bluetooth device with the specified BD_ADDR has successfully changed the Page_Scan_Mode.

Event Parameters:

BD_ADDR:

Size: 6 Bytes

Value	Parameter Description
0XXXXXXXXX XXXX	BD_ADDR of the remote device.

Page_Scan_Mode:

Size: 1 Byte

Value	Parameter Description
0x00	Mandatory Page Scan Mode.
0x01	Optional Page Scan Mode I.
0x02	Optional Page Scan Mode II.
0x03	Optional Page Scan Mode III.
0x04 – 0xFF	Reserved.

5.2.32 Page Scan Repetition Mode Change event

Event	Event Code	Event Parameters
Page Scan Repetition Mode Change	0x20	BD_ADDR, Page_Scan_Repetition_Mode

Description:

The Page Scan Repetition Mode Change event indicates that the remote Bluetooth device with the specified BD_ADDR has successfully changed the Page_Scan_Repetition_Mode (SR).

Event Parameters:*BD_ADDR:**Size: 6 Bytes*

Value	Parameter Description
0XXXXXXXXX XXXX	BD_ADDR of the remote device.

*Page_Scan_Repetition_Mode:**Size: 1 Byte*

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03 – 0xFF	Reserved.

6 LIST OF ERROR CODES

6.1 LIST OF ERROR CODES

This section of the document lists the various possible error codes. When a command fails, Error codes are returned to indicate the reason for the error. Error codes have a size of one byte, and the possible range of failure codes is 0x01-0xFF. Section 6.2 provides an error code usage description for each error code.

Error Code	Description
0x01	Unknown HCI Command.
0x02	No Connection.
0x03	Hardware Failure.
0x04	Page Timeout.
0x05	Authentication Failure.
0x06	Key Missing.
0x07	Memory Full.
0x08	Connection Timeout.
0x09	Max Number Of Connections.
0x0A	Max Number Of SCO Connections To A Device.
0x0B	ACL connection already exists.
0x0C	Command Disallowed.
0x0D	Host Rejected due to limited resources.
0x0E	Host Rejected due to security reasons.
0x0F	Host Rejected due to remote device is only a personal device.
0x10	Host Timeout.
0x11	Unsupported Feature or Parameter Value.
0x12	Invalid HCI Command Parameters.
0x13	Other End Terminated Connection: User Ended Connection.
0x14	Other End Terminated Connection: Low Resources.
0x15	Other End Terminated Connection: About to Power Off.
0x16	Connection Terminated by Local Host.
0x17	Repeated Attempts.

Error Code	Description
0x18	Pairing Not Allowed.
0x19	Unknown LMP PDU.
0x1A	Unsupported Remote Feature.
0x1B	SCO Offset Rejected.
0x1C	SCO Interval Rejected.
0x1D	SCO Air Mode Rejected.
0x1E	Invalid LMP Parameters.
0x1F	Unspecified Error.
0x20	Unsupported LMP Parameter Value.
0x21	Role Change Not Allowed
0x22	LMP Response Timeout
0x23	LMP Error Transaction Collision
0x24	LMP PDU Not Allowed
0x25-0xFF	Reserved for Future Use.

Table 6.1: List of Possible Error Codes

6.2 HCI ERROR CODE USAGE DESCRIPTIONS

The purpose of this section is to give descriptions of how the error codes specified in Table 6.1 on page 745 should be used. It is beyond the scope of this document to give detailed descriptions of all situations where error codes can be used – especially as this may also, in certain cases, be implementation-dependent. However, some error codes that are to be used only in very special cases are described in more detail than other, more general, error codes.

The following error codes are only used in LMP messages, and are therefore not described in this section:

- Unknown LMP PDU (0x19)
- SCO Offset Rejected (0x1B)
- SCO Interval Rejected (0x1C)
- SCO Air Mode Rejected (0x1D)
- Invalid LMP Parameters (0x1E)

Some of the following error code descriptions describe as implementation-dependent whether the error should be returned using a Command Status event or the event associated with the issued command (following a Command Status event with Status=0x00). In these cases, the command can not start executing because of the error, and it is therefore recommended to use the Command Status event. The reason for this suggested course of action is that it is not possible to use the Command Status event in all software architectures.

6.3 UNKNOWN HCI COMMAND (0X01)

The 'Unknown HCI Command' error code is returned by the Host Controller in the Status parameter in a Command Complete event or a Command Status event when the Host Controller receives an HCI Command Packet with an OpCode that it does not recognize. The OpCode given might not correspond to any of the OpCodes specified in this document, or any vendor-specific OpCodes, or the command may not have been implemented. If a Command Complete event is returned, the Status parameter is the only parameter contained in the Return_Parameters event parameter. Which of the two events is used is implementation-dependent.

6.4 NO CONNECTION (0X02)

The 'No Connection' error code is returned by the Host Controller in the Status parameter in an event when the Host has issued a command which requires an existing connection and there is currently no connection corresponding to the specified Connection Handle or BD Address. If the issued command is a command for which a Command Complete event should be returned, the event containing the error code is a Command Complete event. Otherwise, the event containing the error code is a Command Status event or the event associated with the issued command (following a Command Status event with Status=0x00), depending on the implementation.

6.5 HARDWARE FAILURE (0X03)

The 'Hardware Failure' error code is returned by the Host Controller in the Status parameter in an event when the Host has issued a command and this command can not be executed because of a hardware failure. If the issued command is a command for which a Command Complete event should be returned, the event containing the error code is a Command Complete event. Otherwise, the event containing the error code is a Command Status event or the event associated with the issued command (following a Command Status event with Status=0x00) depending on the implementation.

6.6 PAGE TIMEOUT (0X04)

The 'Page Timeout' error code is returned by the Host Controller in the Status parameter of the Connection Complete event when the Host has issued a Create_Connection command and the specified device to connect to does not respond to a page at baseband level before the page timer expires (a page timeout occurs). The error code can also be returned in the Status parameter of a Remote Name Request Complete event when the Host has issued a Remote_Name_Request command and a temporary connection needs to be established but a page timeout occurs. (The page timeout is set using the Write_Page_Timeout command.)

6.7 AUTHENTICATION FAILED (0X05)

The 'Authentication Failed' error code is returned by the Host Controller in the Status parameter in a Connection Complete event or Authentication Complete event when pairing or authentication fails due to incorrect results in the pairing/authentication calculations (because of incorrect PIN code or link key).

6.8 KEY MISSING (0X06)

The 'Key Missing' error code is returned by the Host Controller in the Status parameter in a Connection Complete event or Authentication Complete event when pairing fails because of missing PIN code(s).

6.9 MEMORY FULL (0X07)

The 'Memory Full' error code is returned by the Host Controller in the Status parameter in a Command Complete event when the Host has issued a command that requires the Host Controller to store new parameters and the Host Controller does not have memory capacity for this. This may be the case after the Set_Event_Filter command has been issued. Note that for the Write_Stored_Link_Key command, no error is returned when the Host Controller can not store any more link keys. The Host Controller stores as many link keys as there is free memory to store in, and the Host is notified of how many link keys were successfully stored.

6.10 CONNECTION TIMEOUT (0X08)

Note: this error code is used to indicate a reason for disconnection. It is normally returned in the Reason parameter of a Disconnection Complete event. It is therefore called reason code in the following description.

The 'Connection Timeout' reason code is sent by the Host Controller in an event when the link supervision timer (see "Baseband Timers" on page 993) expires and the link therefore is considered to be lost. The link supervision timeout is set using Write_Link_Supervision_Timeout. The event that returns this reason code will most often be a Disconnection Complete event (in the Reason parameter). The event will be returned on both sides of the connection, where one Disconnection Complete event will be sent from the Host Controller to the Host for each Connection Handle that exists for the physical link to the other device.

(It is possible for a link loss to be detected during connection set up, in which case the reason code would be returned in a Connection Complete event.)

6.11 MAX NUMBER OF CONNECTIONS (0X09)

The 'Max Number Of Connections' error code is returned by the Host Controller in the Status parameter of a Command Status event, a Connection Complete event or a Remote Name Request Complete event when the Bluetooth module can not establish any more connections. It is implementation specific whether the error is returned in a Command Status event or the event following the Command Status event (where Status=0x00 in the Command Status event). The reason for this error may be hardware or firmware limitations. Before the error is returned, the Host has issued a Create_Connection, Add_SCO_Connection or Remote_Name_Request command. The error can be returned in a Remote Name Request Complete event when a temporary connection needs to be established to request the name.

6.12 MAX NUMBER OF SCO CONNECTIONS TO A DEVICE (0X0A)

The 'Max Number Of SCO Connections To A Device' error code is returned by the Host Controller in the Status parameter of a Command Status event or a Connection Complete event (following a Command Status event with Status=0x00) when the maximum number of SCO connections to a device has been reached. Which of the two events that is used depends on the implementation. The device is a device that has been specified in a previously issued Add_SCO_Connection command.

6.13 ACL CONNECTION ALREADY EXISTS (0X0B)

The 'ACL connection already exists' error code is returned by the Host Controller in the Status parameter of a Command Status event or a Connection Complete event (following a Command Status event with Status=0x00) when there already is one ACL connection to a device and the Host tries to establish another one using Create_Connection. Which of the two events that is used depends on the implementation.

6.14 COMMAND DISALLOWED (0X0C)

The 'Command Disallowed' error code is returned by the Host Controller in the Status parameter in a Command Complete event or a Command Status event when the Host Controller is in a state where it is only prepared to accept commands with certain OpCodes and the HCI Command Packet received does not contain any of these OpCodes. The Command Complete event should be used if the issued command is a command for which a Command Complete event should be returned. Otherwise, the Command Status event should be used. The Host Controller is not required to use the 'Unknown HCI Command' error code, since this may require unnecessary processing of the received (and currently not allowed) OpCode. When to use the 'Command Disallowed' error code is mainly implementation-dependent. Certain implementations may, for example, only accept the appropriate HCI response commands after the Connection Request, Link Key Request or PIN Code Request events.

Note: the Reset command should always be allowed.

6.15 HOST REJECTED DUE TO ... (0X0D-0X0F)

Note: these error codes are used to indicate a reason for rejecting an incoming connection. They are therefore called reason codes in the following description.

When a Connection Request event has been received by the Host and the Host rejects the incoming connection by issuing the Reject_Connection_Request command, one of these reason codes is used as value for the Reason parameter. The issued reason code will be returned in the Status parameter of the Connection Complete event that will follow the Command Status event

(with Status=0x00) returned by the Host Controller after the Reject_Connection_Request command has been issued. The reason code issued in the Reason parameter of the Reject_Connection_Request command will also be sent over the air, so that it is returned in a Connection Complete event on the initiating side. Before this, the initiating side has issued a Create_Connection command or Add_SCO_Connection command, and has received a Command Status event (with Status=0x00).

6.16 HOST TIMEOUT (0X10)

Note: this error code is used to indicate a reason for rejecting an incoming connection. It is therefore called reason code in the following description.

Assume that a Connection Request event has been received by the Host and that the Host does not issue the Accept_Connection_Request or Reject_Connection_Request command before the connection accept timer expires (the connection accept timeout is set using Write_Connection_Accept_Timeout). In this case, the 'Host Timeout' reason code will be sent by the Host Controller in the Status parameter of a Connection Complete event. The reason code will also be sent over the air, so that it is returned in a Connection Complete event on the initiating side. The initiating side has before this issued a Create_Connection or Add_SCO_Connection command and has received a Command Status event (with Status=0x00).

6.17 UNSUPPORTED FEATURE OR PARAMETER VALUE (0X11)

The 'Unsupported Feature or Parameter Value' error code is returned by the Host Controller in the Status parameter in an event when the Host Controller has received a command where one or more parameters have values that are not supported by the hardware (the parameters are, however, within the allowed parameter range specified in this document). If the issued command is a command for which a Command Complete event should be returned, the event containing the error code is a Command Complete event. Otherwise, the event containing the error code is a Command Status event or the event associated with the issued command (following a Command Status event with Status=0x00) depending on the implementation.

6.18 INVALID HCI COMMAND PARAMETERS (0X12)

The 'Invalid HCI Command Parameters' error code is returned by the Host Controller in the Status parameter of an event when the total parameter length (or the value of one or more parameters in a received command) does not conform to what is specified in this document.

The error code can also be returned if a parameter value is currently not allowed although it is inside the allowed range for the parameter. One case is when a command requires a Connection Handle for an ACL connection but the

Host has given a Connection Handle for an SCO connection as a parameter instead. Another case is when a link key, a PIN code or a reply to an incoming connection has been requested by the Host Controller by using an event but the Host replies using a response command with a BD_ADDR for which no request has been made.

If the issued command is a command for which a Command Complete event should be returned, the event containing the error code is a Command Complete event. Otherwise, the event containing the error code is a Command Status event or the event associated with the issued command (following a Command Status event with Status=0x00), depending on the implementation.

6.19 OTHER END TERMINATED CONNECTION: ... (0X13-0X15)

Note: these error codes are used to indicate a reason for disconnecting a connection. They are therefore called reason codes in the following description.

When the Host issues the Disconnect command, one of these reason codes is used as value for the reason parameter. The 'Connection Terminated By Local Host' reason code will then be returned in the Reason parameter of the Disconnection Complete event that will follow the Command Status event (with Status=0x00) that is returned by the Host Controller after the Disconnect command has been issued. The reason code issued in the Reason parameter of the Disconnect command will also be sent over the air, so that it is returned in the Reason parameter of a Disconnection Complete event on the remote side.

6.20 CONNECTION TERMINATED BY LOCAL HOST (0X16)

See description in 6.19. This error code is called a reason code, since it is returned in the Reason parameter of a Disconnection Complete event.

6.21 REPEATED ATTEMPTS (0X17)

The 'Repeated Attempts' error code is returned by the Host Controller in the Status parameter in a Connection Complete event or Authentication Complete event when a device does not allow authentication or pairing because too little time has elapsed since an unsuccessful authentication or pairing attempt. See "Link Manager Protocol" on page 185 for a description of how repeated attempts work.

6.22 PAIRING NOT ALLOWED (0X18)

The 'Pairing Not Allowed' error code is returned by the Host Controller in the Status parameter in a Connection Complete event or Authentication Complete event when a device for some reason does not allow pairing. An example may be a PSTN adapter that only allows pairing during a certain time window after a button has been pressed on the adapter.

6.23 UNSUPPORTED REMOTE FEATURE (0X1A)

The 'Unsupported Remote Feature' error code is returned by the Host Controller in the Status parameter of the event associated with the issued command when a remote device that has been specified in the command parameters does not support the feature associated with the issued command. The 'Unsupported Remote Feature' error code can also be used as a value for the Reason parameter in the Disconnect command (as a reason code). The error code will then be sent over the air so that it is returned in the Reason parameter of a Disconnection Complete event on the remote side. In the Disconnection Complete event following a Command Status event (where Status=0x00) on the local side on which the Disconnect command has been issued, the Reason parameter will however contain the reason code 'Connection Terminated By Local Host'. (The 'Unsupported Remote Feature' error code is called 'Unsupported LMP Feature' in the LMP specification, see "Link Manager Protocol" on page 185.)

6.24 UNSPECIFIED ERROR (0X1F)

The 'Unspecified error' error code is used when no other error code specified in this document is appropriate to use.

6.25 UNSUPPORTED LMP PARAMETER VALUE (0X20)

The 'Unsupported LMP Parameter Value' error code is returned by the Host Controller in the Status parameter of the event associated with the issued command when a remote device that has been specified in the command parameters sent back an LMP message containing the LMP error code 0x20, 'Unsupported parameter values' (see "Link Manager Protocol" on page 185).

6.26 ROLE CHANGE NOT ALLOWED (0X21)

The 'Role Change Not Allowed' error code is returned by the Host Controller in the Status parameter in a Connection Complete event or Role Change event when role change is not allowed. If the local Host issues the Switch_Role command and the remote device rejects the role change, the error code will be returned in a Role Change event. If a connection fails because a device accepts an incoming ACL connection with a request for role change and the role change is rejected by the initiating device, the error code will be returned in a Connection Complete event on both sides.

6.27 LMP RESPONSE TIMEOUT (0X22)

The 'LMP Response Timeout' error code is returned by the Host Controller in the Status parameter in a Command Complete event or an event associated with the issued command following a Command Status event with Status=0x00, when the remote device does not respond to the LMP PDUs from

the local device as a result of the issued command within LMP response timeout. (See "Link Manager Protocol" on page 185)

6.28 LMP ERROR TRANSACTION COLLISION (0X23)

The 'LMP Error Transaction Collision' error code is returned by the Host Controller in the Status parameter of the event associated with the issued command when a remote device that has been specified in the command parameters sends back an LMP message containing the LMP error code 0x23, "LMP Error Transaction Collision" (see "Link Manager Protocol" on page 185).

6.29 LMP PDU NOT ALLOWED (0X24)

The 'LMP PDU Not Allowed' error code is returned by the Host Controller in the Status parameter of the event associated with the issued command when a remote device that has been specified in the command parameters sends back an LMP message containing the LMP error code 0x24, "PDU Not Allowed" (see "Link Manager Protocol" on page 185).

7 LIST OF ACRONYMS AND ABBREVIATIONS

Acronym or abbreviation	Complete name
ACL	Asynchronous Connection Less
BD_ADDR	Bluetooth Device Address
DH	Data High rate
DIAC	Dedicated Inquiry Access Code
DM	Data Medium rate
DUT	Device Under Test
DV	Data Voice
GIAC	General Inquiry Access Code
HCI	Host Controller Interface
L2CAP	Logical Link Control and Adaptation Protocol
L_CH	Logical Channel
LAP	Lower Address Part
LC	Link Controller
LM	Link Manager
LMP	Link Manager Protocol
OCF	Opcode Command Field
OGF	OpCode Group Field
RF	Radio Frequency
RSSI	Received Signal Strength Indication
SCO	Synchronous Connection Oriented
TBD	To Be Defined
UA	User Asynchronous
UI	User Isochronous
US	User Synchronous
USB	Universal Serial Bus

Table 7.1: List of Acronyms and Abbreviations

8 LIST OF FIGURES

Figure 1.1:	Overview of the Lower Software Layers	524
Figure 1.2:	End to End Overview of Lower Software Layers to Transfer Data	525
Figure 1.3:	Bluetooth Hardware Architecture Overview.	526
Figure 1.4:	Bluetooth Block Diagram with USB HCI	527
Figure 1.5:	Bluetooth Block Diagram with PC-Card HCI.....	527
Figure 4.1:	HCI Command Packet	534
Figure 4.2:	HCI Event Packet	535
Figure 4.3:	HCI ACL Data Packet	536
Figure 4.4:	HCI SCO Data Packet	538
Figure 4.5:	Local Loopback Mode.....	697
Figure 4.6:	Remote Loopback Mode.....	697
Figure 4.7:	Local Loopback Mode.....	700
Figure 4.8:	Remote Loopback Mode.....	700

9 LIST OF TABLES

Table 5.1:	List of Supported Events	703
Table 6.1:	List of Possible Error Codes.....	746
Table 7.1:	List of Acronyms and Abbreviations.....	755



Part H:2

HCI USB TRANSPORT LAYER

An addendum to the HCI document

This document describes the USB transport layer (between a host and the host controller). HCI commands flow through this layer, but the layer does not decode the commands.



CONTENTS

1	Overview	762
2	USB Endpoint Expectations	764
	2.1 Descriptor Overview	764
	2.2 Control Endpoint Expectations	769
	2.3 Bulk Endpoints Expectations	769
	2.4 Interrupt Endpoint Expectations	769
	2.5 Isochronous Endpoints Expectations	770
3	Class Code	771
4	Device Firmware Upgrade	772
5	Limitations	773
	5.1 Power Specific Limitations	773
	5.2 Other Limitations	773

1 OVERVIEW

This document discusses the requirements of the Universal Serial Bus (USB) interface for Bluetooth hardware. Readers should be familiar with USB, USB design issues, Advanced Configuration Power Interface (ACPI), the overall Bluetooth architecture, and the basics of the radio interface.

The reader should also be familiar with the Bluetooth Host Controller Interface.

Referring to Figure 1.1 below, notice that this document discusses the implementation details of the two-way arrow labelled 'USB Function'.

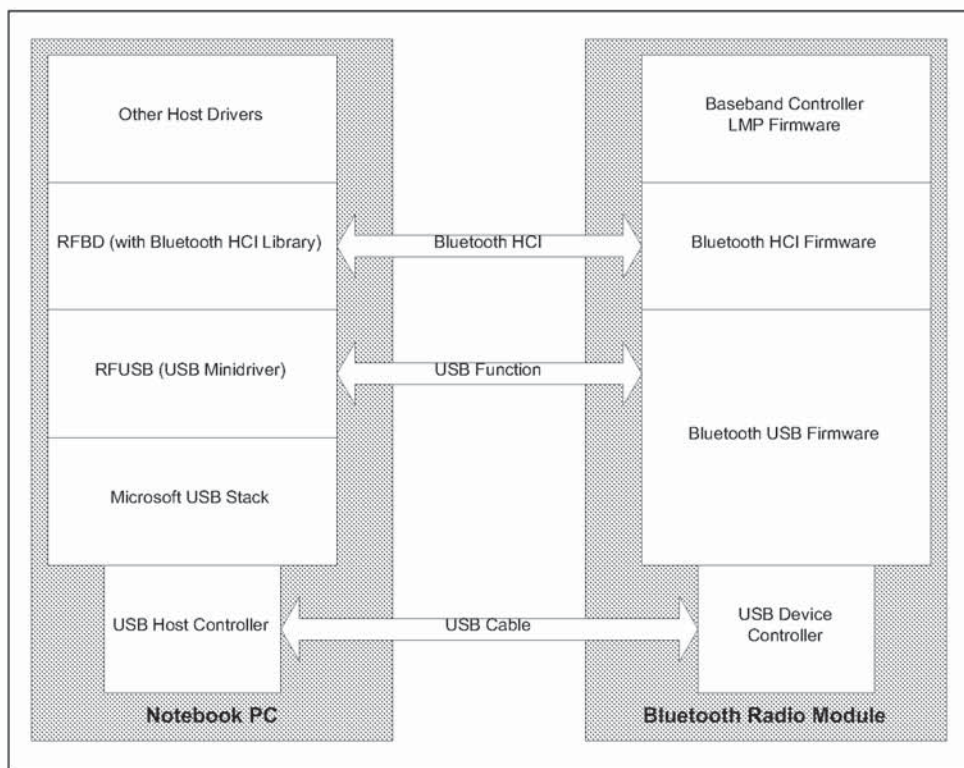


Figure 1.1: The Figure illustrates the relationship between the host and the Bluetooth Radio Module

The USB hardware can be embodied in one of two ways:

1. As a USB dongle, and
2. Integrated onto the motherboard of a notebook PC.

Finally, for an overview of the connection that is established between two Bluetooth devices, reference Figure 1.2, below.

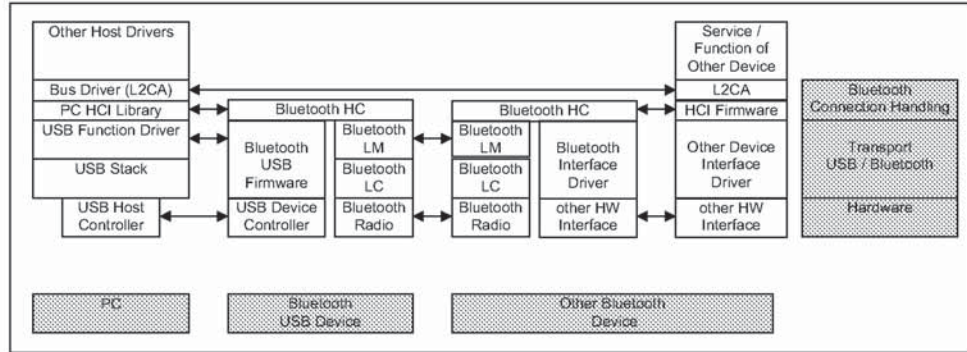


Figure 1.2: The figure illustrates the flow of data from one Bluetooth device to another

2 USB ENDPOINT EXPECTATIONS

This section outlines specific USB endpoints that are required in order to function properly with the host. This section assumes a basic familiarity with USB. The endpoint numbers (labelled 'Suggested Endpoint Address' below) may be dynamically recognized upon driver initialization – this depends on the implementation.

2.1 DESCRIPTOR OVERVIEW

The USB device is expected to be a high-speed device.

The firmware configuration consists of two interfaces. The first interface (interface zero) has no alternate settings and contains the bulk and interrupt endpoints. The second interface (interface one) provides scalable isochronous bandwidth consumption. The second interface has four alternate settings that provide different consumption based on the required isochronous bandwidth. The default interface is empty so that the device is capable of scaling down to no isochronous bandwidth.

An HCI frame - consisting of an HCI header and HCI data - should be contained in one USB transaction. A USB transaction is defined as one or more USB frames that contain the data from one IO request. For example, an ACL data packet containing 256 bytes (both HCI header and HCI data) would be sent over the bulk endpoint in one IO request. That IO request will require four 64-byte USB frames - and forms a transaction.

The endpoints are spread across two interfaces so, when adjusting isochronous bandwidth consumption (via select interface calls), any pending bulk and/or interrupt transactions do not have to be terminated and resubmitted.

The following table outlines the required configuration

Interface Number	Alternate Setting	Suggested Endpoint Address	Endpoint Type	Suggested Max Packet Size
HCI Commands				
0	0	0x00	Control	8/16/32/64
HCI Events				
0	0	0x81	Interrupt (IN)	16
ACL Data				
0	0	0x82	Bulk (IN)	32/64
0	0	0x02	Bulk (OUT)	32/64
No active voice channels (for USB compliance)				
1	0	0x83	Isoch (IN)	0
1	0	0x03	Isoch (OUT)	0
One voice channel with 8-bit encoding				
1	1	0x83	Isoch (IN)	9
1	1	0x03	Isoch (OUT)	9
Two voice channels with 8-bit encoding & One voice channel with 16-bit encoding				
1	2	0x83	Isoch (IN)	17
1	2	0x03	Isoch (OUT)	17
Three voice channels with 8-bit encoding				
1	3	0x83	Isoch (IN)	25
1	3	0x03	Isoch (OUT)	25
Two voice channels with 16-bit encoding				
1	4	0x83	Isoch (IN)	33
1	4	0x03	Isoch (OUT)	33
Three voice channels with 16-bit encoding				
1	5	0x83	Isoch (IN)	49
1	5	0x03	Isoch (OUT)	49

The following two examples are used to demonstrate the flow of data given the describe endpoints.

Number of voice channels	Duration of voice data	Encoding
One	3 ms per IO Request	8-bit

Time (ms)	USB data (header refers to HCI header) (Receive & Send from the host)	Queued data (read / write)	Time (ms)	Air data	Amount Received / Sent (ms)
0	Receive 0 bytes Send 9 bytes (3 header, 6 data)	0 / 6	0	Send 0	0 / 0
		10 / 6	0.625	Receive 10	1.25 / 0
1	Receive 0 bytes Send 9 bytes (9 bytes HCI data)	10 / 15	1.25	Send 0	1.25 / 0
		20 / 15	1.875	Receive 10	2.50 / 0
2	Receive 0 bytes Send 9 bytes (9 bytes HCI data)	20 / 24	2.50	Send 0	2.50 / 0
		30 / 24	3.125	Receive 10	3.75 / 0
3	Receive 9 bytes (3 header, 6 data) Send 9 bytes (3 header, 6 data)	24 / 20	3.75	Send 10	3.75 / 1.25
4	Receive 9 bytes (9 bytes data) Send 9 bytes (9 bytes HCI data)	25 / 29	4.375	Receive 10	5.0 / 1.25
5	Receive 9 bytes (9 bytes data) Send 9 bytes (9 bytes HCI data)	16 / 28	5.0	Send 10	5.0 / 2.50
		26 / 28	5.625	Receive 10	6.25 / 2.50
6	Receive 9 bytes (3 header, 6 data) Send 9 bytes (3 header, 6 data)	20 / 24	6.25	Send 10	6.25 / 3.75
		30 / 24	6.875	Receive 10	7.5 / 3.75
7	Receive 9 bytes (9 bytes data) Send 9 bytes (9 bytes HCI data)	21 / 23	7.5	Send 10	7.5 / 5.0
8	Receive 9 bytes (9 bytes data) Send 9 bytes (9 bytes HCI data)	22 / 32	8.125	Receive 10	8.75 / 5.0
		22 / 22	8.75	Send 10	8.75 / 6.25
9	Receive 9 bytes (3 header, 6 data) Send 9 bytes (3 header, 6 data)	26 / 28	9.375	Receive 10	10.0 / 6.25

Table 2.1:

Time (ms)	USB data (header refers to HCI header) (Receive & Send from the host)	Queued data (read / write)	Time (ms)	Air data	Amount Received / Sent (ms)
10	Receive 9 bytes (9 bytes data) Send 9 bytes (9 bytes HCI data)	17 / 27	10	Send 10	10.0 / 7.5
		27 / 27	10.625	Receive 10	11.25 / 7.5
11	Receive 9 bytes (9 bytes data) Send 9 bytes (9 bytes HCI data)	18 / 26	11.25	Send 10	11.25 / 8.75

Table 2.1:

Convergence is expected because the radio is sending out an average of 8 bytes of voice data every 1 ms and USB is sending 8 bytes of voice data every 1 ms.

Number of voice channels	Duration of voice data	Encoding
Two	3 ms per IO Request	8-bit

Time (ms)	USB data (header refers to HCI header) (Receive & Send from the host)	Queued data (read / write)	Time (ms)	Air data	Amount Received / Sent (ms)
0	Receive 0 bytes for Channel #1 Send 17 bytes (3 header, 14 data) for Channel #1	C1- 0/14 C2- 0/0	0	Send 0 for C1	C1- 0/0 C2- 0/0
		C1- 20/14 C2- 0/0	0.625	Receive 20 for C1	C1- 2.5/0 C2- 0/0
1	Receive 0 bytes for Channel #1 Send 17 bytes (17 bytes HCI data) for Channel #1	C1- 20/31 C2- 0/0	1.25	Send 0 for C2	C1- 2.5/0 C2- 0/0
		C1- 20/31 C2- 20/0	1.875	Receive 20 for C2	C1- 2.5/0 C2- 2.5/0
2	Receive 0 bytes for Channel #1 Send 17 bytes (17 bytes HCI data) for Channel #1	C1- 20/28 C2- 20/0	2.50	Send 20 for C1	C1- 2.5/2.5 C2- 2.5/0
		C1- 40/28 C2- 0/0	3.125	Receive 20 for C1	C1- 5.0/2.5 C2- 2.5/0

Table 2.2:

Time (ms)	USB data (header refers to HCI header) (Receive & Send from the host)	Queued data (read / write)	Time (ms)	Air data	Amount Received / Sent (ms)
3	Receive 0 bytes for Channel #2 Send 17 bytes (3 header, 14 data) for Channel #2	C1- 40/28 C2- 20/14	3.75	Send 0 for C2	C1- 5.0/2.5 C2- 2.5/0
4	Receive 0 bytes for Channel #2 Send 17 bytes (17 bytes HCI data) for Channel #2	C1- 40/28 C2- 40/31	4.375	Receive 20 for C2	C1- 5.0/2.5 C2- 5.0/0
5	Receive 0 bytes for Channel #2 Send 17 bytes (17 bytes HCI data) for Channel #2	C1- 40/8 C2- 40/48	5.0	Send 20 for C1	C1- 5.0/5.0 C2- 5.0/0
		C1- 60/8 C2- 40/48	5.625	Receive 20 for C1	C1- 7.5/5.0 C2- 5.0/0
6	Receive 17 bytes (3 header, 14 data) for Channel #1 Send 17 bytes (3 header, 14 data) for Channel #1	C1- 46/22 C2- 40/28	6.25	Send 20 for C2	C1- 7.5/5.0 C2- 5.0/2.5
		C1- 46/22 C2- 60/28	6.875	Receive 20 for C2	C1- 7.5/5.0 C2- 7.5/2.5
7	Receive 17 bytes (17 bytes data) for Channel #1 Send 17 bytes (17 bytes HCI data) for Channel #1	C1- 29/19 C2- 60/28	7.5	Send 20 for C1	C1- 7.5/7.5 C2- 7.5/2.5
8	Receive 17 bytes (17 bytes data) for Channel #1 Send 17 bytes (17 bytes HCI data) for Channel #1	C1- 32/36 C2- 60/28	8.125	Receive 20 for C1	C1- 10/7.5 C2- 7.5/2.5
		C1- 32/36 C2- 60/8	8.75	Send 20 for C2	C1- 10/7.5 C2- 7.5/5.0
9	Receive 17 bytes (3 header, 14 data) for Channel #2 Send 17 bytes (3 header, 14 data) for Channel #2	C1- 32/36 C2- 54/22	9.375	Receive 20 for C2	C1- 10/7.5 C2- 10/5.0
10	Receive 17 bytes (17 bytes data) for Channel #2 Send 17 bytes (17 bytes HCI data) for Channel #2	C1- 32/16 C2- 37/39	10	Send 20 for C1	C1- 10/10 C2- 10/5.0
		C1- 52/16 C2- 37/39	10.625	Receive 20 for C1	C1- 12.5/10 C2- 10/5.0

Table 2.2:

Time (ms)	USB data (header refers to HCI header) (Receive & Send from the host)	Queued data (read / write)	Time (ms)	Air data	Amount Received / Sent (ms)
11	Receive 17 bytes (17 bytes data) for Channel #2 Send 17 bytes (17 bytes HCI data) for Channel #2	C1- 52/16 C2- 20/36	11.25	Send 20 for C2	C1- 12.5/10 C2- 10/7.5

Table 2.2:

2.2 CONTROL ENDPOINT EXPECTATIONS

Endpoint 0 is used to configure and control the USB device. Endpoint 0 will also be used to allow the host to send HCI-specific commands to the host controller. When the USB firmware receives a packet over this endpoint that has the Bluetooth class code, it should treat the packet as an HCI command packet.

2.3 BULK ENDPOINTS EXPECTATIONS

Data integrity is a critical aspect for ACL data. This, in combination with bandwidth requirements, is the reason for using a bulk endpoint. Multiple 64-byte packets can be shipped, per millisecond, across the bus.

Suggested bulk max packet size is 64 bytes. Bulk has the ability to transfer multiple 64-byte buffers per one millisecond frame, depending on available bus bandwidth.

Bulk has the ability to detect errors and correct them. Data flowing through this pipe might be destined for several different slaves. In order to avoid starvation, a flow control model similar to the shared endpoint model is recommended for the host controller.

2.4 INTERRUPT ENDPOINT EXPECTATIONS

An interrupt endpoint is necessary to ensure that events are delivered in a predictable and timely manner. Event packets can be sent across USB with a guaranteed latency.

The interrupt endpoint should have an interval of 1 ms.

The USB software and firmware requires no intimate knowledge of the events passed to the host controller.

2.5 ISOCHRONOUS ENDPOINTS EXPECTATIONS

These isochronous endpoints transfer SCO data to and from the host controller of the radio.

Time is the critical aspect for this type of data. The USB firmware should transfer the contents of the data to the host controllers' SCO FIFOs. If the FIFOs are full, the data should be overwritten with new data.

These endpoints have a one (1) ms interval, as required by Chapter 9 of the USB Specification, Versions 1.0 and 1.1.

The radio is capable of three (3) 64Kb/s voice channels (and can receive the data coded in different ways – 16-bit linear audio coding is the method that requires the most data). A suggested max packet size for this endpoint would be at least 64 bytes. (It is recommended that max packet sizes be on power of 2 boundaries for optimum throughput.) However, if it is not necessary to support three voice channels with 16-bit coding, 32 bytes could also be considered an acceptable max packet size.

3 CLASS CODE

A class code will be used that is specific to all USB Bluetooth devices. This will allow the proper driver stack to load, regardless of which vendor built the device. It also allows HCI commands to be differentiated from USB commands across the control endpoint.

The class code (bDeviceClass) is 0xE0 – Wireless Controller.

The SubClass code (bDeviceSubClass) is 0x01 – RF Controller.

The Protocol code (bDeviceProtocol) is 0x01 – Bluetooth programming.

4 DEVICE FIRMWARE UPGRADE

Firmware upgrade capability is not a required feature. But if implemented, the firmware upgrade shall be compliant with the "Universal Serial Bus Device Class Specification for Device Firmware Upgrade" (version 1.0 dated May 13, 1999) available on the USB Forum web site at <http://www.usb.org>.

5 LIMITATIONS

5.1 POWER SPECIFIC LIMITATIONS

Today, the host controller of USB-capable machines resides inside a chip known as PIIX4. Unfortunately, because of errata, the USB host controller will not receive power while the system is in S3 or S4. This means that a USB wake-up can only occur when the system is in S1 or S2.

Another issue with the USB host controller is that, while a device is attached, it continually snoops memory to see if there is any work that needs to be done. The frequency that it checks memory is 1ms. This prevents the processor from dropping into a low power state known as C3. Because the notebook processor is not able to enter the C3 state, significant power loss will occur. This is a real issue for business users – as a typical business user will spend almost 90% of their time in the C3 state.

5.2 OTHER LIMITATIONS

Data corruption may occur across isochronous endpoints. Endpoints one and two may suffer from data corruption.

USB provides 16-CRC on all data transfers. The USB has a bit error rate of 10^{-13} .

Note that when a dongle is removed from the system, the radio will lose power (assuming this is a bus-powered device). This means that devices will lose connection.



Part H:3

HCI RS232 TRANSPORT LAYER

An addendum to the HCI document

This document describes the RS232 transport layer (between the Host and the Host Controller). HCI command, event and data packets flow through this layer, but the layer does not decode them.



CONTENTS

1	General.....	778
2	Overview	779
3	Negotiation Protocol.....	780
4	Packet Transfer Protocol.....	784
5	Using delimiters with COBS for synchronization	785
5.1	Using Delimiters with COBS and CRC, Protocol Mode 0x13...	785
5.2	Frame Format	786
5.3	Error Message Packet.....	786
5.4	Consistent Overhead Byte Stuffing	787
6	Using RTS/CTS for Synchronization	788
6.1	Using RTS/CTS for Sync without CRC, Protocol Mode 0x14 ..	788
6.2	Error Message Packet.....	789
6.3	Example of Signalling.....	790
6.4	Control Flow Examples	791
6.4.1	Case 1, Normal Recovery Process	791
6.4.2	Case 2, Both sides detect an error simultaneously	791
6.4.3	Case 3, Error Message with an error	792
7	References.....	794

1 GENERAL

The objective of the HCI RS232 Transport Layer is to make it possible to use the Bluetooth HCI over one physical RS232 interface between the Bluetooth Host and the Bluetooth Host Controller.



Figure 1.1:

2 OVERVIEW

There are four kinds of HCI packets that can be sent via the RS232 Transport Layer; i.e. HCI Command Packet, HCI Event Packet, HCI ACL Data Packet and HCI SCO Data Packet (see "Host Controller Interface Functional Specification" on page 517). HCI Command Packets can only be sent to the Bluetooth Host Controller, HCI Event Packets can only be sent from the Bluetooth Host Controller, and HCI ACL/SCO Data Packets can be sent both to and from the Bluetooth Host Controller.

However, HCI does not provide the ability to differentiate the four HCI packet types. Therefore, if the HCI packets are sent via a common physical interface, a HCI packet indicator has to be added according to the Table 2.1 below.

HCI packet type	HCI packet indicator
HCI Command Packet	0x01
HCI ACL Data Packet	0x02
HCI SCO Data Packet	0x03
HCI Event Packet	0x04
Error Message Packet*	0x05
Negotiation Packet*	0x06

Table 2.1: HCI RS232 Packet Header

In addition to those four HCI packet types, two additional packet types are introduced to support dynamic negotiation and error reporting. The Error Message Packet (0x05) is used by the receiver to report the nature of error to the transmitting side. The Negotiation Packet (0x06) is used to negotiate the communication settings and protocols.

The HCI packet indicator shall be followed by an 8-bit sequence number that is incremented by 1 every time any of the above packets are sent, except when the retransmission packets are sent as a part of the error recovery. The HCI packet shall immediately follow the sequence number field. All four kinds of HCI packets have a length field, which is used to determine how many bytes are expected for the HCI packet. The Error Message Packet and Negotiation Packet are fixed-length packets, although the negotiation packet can be extended up to 7 more bytes, based on the number in the extension field.

The frame of the basic RS232 Transport Packet is shown below.



The least significant byte is transmitted first.

3 NEGOTIATION PROTOCOL

Before sending any bytes over the RS232 link, the baud rate, parity type, number of stop bit and protocol mode should be negotiated between the Host Controller and the Host. Tdetect is the maximum time required for the transmitter to detect the CTS state change, plus the time it takes to flush the transmit buffer if RTS/CTS is used for error indication and re-synchronization. Otherwise, Tdetect represents the local-side interrupt latency. Host will first send a negotiation packet with the maximum suggested values, plus Host's Tdetect value with Ack code = 000b at the default UART settings specified below, using protocol mode = 0x13. At the same time, the Host Controller side also sets its UART settings to the same initiating parameters and waits for the negotiation packet from the Host.

If the Host Controller side can accept the suggested values from the Host, it sends back the negotiation packet with the same UART setting values, plus Host Controller's Tdetect value with Ack code = 001b. Then, the Host sends back the negotiation packet with the same UART setting values, plus Host's Tdetect with Ack code = 001b as the final acknowledgment, and then sets its Host's UART to the new value. After it has received the final acknowledgment packet from the Host, the Host Controller also changes its UART setting to the new values.

On the other hand, if the Host Controller side cannot accept the suggested value, it should send a set of new suggested values and its own Tdetect value with Ack code = 010b. Each side should continue these steps until both sides receive the accepted Ack code value. Error detection and error recovery during the initial negotiation are performed in the same manner as described in Section 5 on page 785 (Protocol Mode 0x13)

The negotiation phase can be initiated again by either side at any time in order to renegotiate the new values, or just to inform the new Tdetect time. When the negotiation is reinitiated during the data transfer, it should use the previously negotiated settings to exchange the new parameters rather than using the default values.

The initiating parameters:

baud rate: 9600 bps

parity type: no parity

number of data bit: 8 (Note: Only 8-bit data length is allowed.)

number of stop bit: 1

protocol mode: 0x13 (HDLC like framing with COBS/CCITT-CRC)

The negotiation packet format:

LSB

MSB

Packet Type header 0x06 (8 bits)	SEQ No (8 bits)	UART Settings and ACK (8 bits)	Baud Rate (16 bits)	Tdetect Time (16 bits)	Protocol Mode (8 bit)
----------------------------------	-----------------	--------------------------------	---------------------	------------------------	-----------------------

SEQ No:

This is an 8-bit number that is incremented by 1 each time a packet is transmitted, excluding the retransmission packet. The unsigned Little Endian format is used.

UART Settings and ACK Field

Bit 0-1	Bit 2	Bit 3	Bit 4	Bit 5-7
Reserved	Stop bit (1 bit)	Parity Enable (1 bit)	Parity Type (1 bit)	Ack Code (3 bits)

Stop Bit:

- 0: 1 stop bit
- 1: 2 stop bits

Parity Enable:

- 0: No parity
- 1: Parity

Parity Type:

- 0: Odd Parity
- 1: Even Parity

Ack Code:

- 000b: Request
- 001b: Accepted
- 010b: Not accepted with new suggested values
- 011b-111b: Reserved

Baud Rate:

N should be entered for baud rate where

$$\text{Baud rate} = 27,648,000 / N$$

N=0 is invalid

Maximum possible rate is therefore 27.648Mbps

Minimum possible rate is therefore 421.88bps

The unsigned Little Endian format is used, and the least significant byte should be transmitted first.

Tdetect Time:

This 16-bit field should be filled with the maximum required time for the transmitter to detect the CTS change, plus the time it takes to flush the transmit FIFO if RTS and CTS are used for resynchronization. Otherwise, it should be filled with the local interrupt latency.

The unit of time should be specified in 100 microseconds.

The unsigned Little Endian format is used, and the least significant byte should be transmitted first.

Protocol Mode

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
CRC Used	Delimiter Used	RTS /CTS used	RTS/CTS Mode	Error Recovery Used	Ext0	Ext1	Ext2

CRC Used:

- 0: CRC-CCITT is not attached at the end of the packet.
- 1: CRC-CCITT is attached at the end of the packet. (Default)

16-bit CRC can be used with either RTS/CTS or delimiters, although this specification only describes a case when it is used with delimiters.

Generator Polynomial = $x^{16}+x^{12}+x^5+1$

Delimiter Used:

- 0: Delimiter, 0x7E, is not used.
- 1: Delimiter, 0x7E, is used with COBS. (Default)

RTS/CTS Used:

- 0: RTS/CTS is not used. (Default)
- 1: RTS/CTS is used.

RTS/CTS Mode:

- 0: RTS/CTS is used for Error indication and resynchronization. (Default)
- 1: RTS/CTS is used for hardware flow control. Please refer to "HCI UART Transport Layer" on page 795 for details.

Error Recovery Used:

- 0: Error Recovery is not supported.
Even if Error Recovery is not supported, Error Message has to be sent.
- 1: Error Recovery is supported. (Default)
Error Recovery retransmits the packet with error and all subsequent packets if RTS/CTS are used for synchronization. On the other hand, if 0x7E is used as a delimiter with COBS as a synchronization mechanism, then the error recovery retransmits only the packet with error. Please refer to following sections for details.

Ext2,Ext1,Ext0:

These three bits indicate the number of extra bytes attached to the negotiation packet for future expansion.

4 PACKET TRANSFER PROTOCOL

The packet can be transferred with parity enabled or disabled, and with or without CRC – depending on the environment – as a mechanism to detect the error.

As a synchronization mechanism, one can select either RTS/CTS, or delimiters. Usage of RTS/CTS reduces the computation time for COBS encoding, but it requires two extra copper wires which may not be suitable in some applications. If three-wire cable must be used, or programmable RTS and CTS are not available, delimiter, 0x7E, can be used with COBS.

However, error recovery for these two alternatives may differ slightly. If the RTS/CTS is used for resynchronization, it would be simpler to retransmit all the packets, starting with the packet that had an error. If delimiters are used, the transmitter should retransmit only the packet with an error. The error recovery can be disabled, but the error message packet should still be sent to the transmitter side when the receiver side detects an error.

The HCI RS232 transport layer always uses a data length of 8 bits, and this specification assumes the Little Endian format. Furthermore, the least significant byte should be transmitted first.

The Host Controller may choose to support only one protocol mode, but the Host should be able to support any combination.

Two common schemes (Protocol mode = 0x13 and 0x14) are defined in the following sections to illustrate the usage of each mode.

5 USING DELIMITERS WITH COBS FOR SYNCHRONIZATION

This section illustrates how delimiters with COBS are used for synchronization, and how error recovery procedure is performed if delimiters are used as a mechanism to synchronize. This is described using protocol mode 0x13.

5.1 USING DELIMITERS WITH COBS AND CRC, PROTOCOL MODE 0X13

In case RTS/CTS are not available, or if they are hard-wired to be used as a hardware flow control, the HDLC-like framing with the 16-bit CRC (CRC-CCITT) and delimiter 0x7E with COBS (Consistent Overhead Byte Stuffing) [2] are used as a means to detect an error and to resynchronize.

The CRC-CCITT uses the following generator polynomial for 16-bit checksum: $x^{16}+x^{12}+x^5+1$. The 16-bit CRC should be attached to the end of the packet, but right before the ending delimiter, 0x7E. The beginning delimiter, 0x7E, should be followed by the packet type indicator field.

The Consistent Overhead Byte Stuffing is a recent proposal to PPP that yields less than 0.5% overhead, regardless of the data pattern. It uses two steps to escape the delimiter, 0x7E. The first step is eliminating zeros and then replacing all 0x7E with 0x00 between the beginning and ending delimiters.

A simple error recovery scheme is adapted here to minimize the overhead of supporting the error recovery. When the receiving end detects any error, it should send the error message packet with an error type back to the transmitting side. This error message packet will contain a Sequence Number with Error field (SEQ No with Error) indicating in which packet the error was detected. The Sequence Number field (SEQ No) that is on every packet is an 8-bit field that is incremented by 1 each time any type of packet is transmitted, except for the retransmission packets. The retransmitted packets should contain the original sequence number in the SEQ No field.

The transmitting side should retransmit only the HCI packets that had an error, which is indicated by the SEQ No with Error field. It is the responsibility of the receiving end to reorder the packets in the right order. If the transmitting side doesn't have the packet with the correct sequence number in the retransmission holding buffer, it should send the error message packet with the Error Type equal to 0x81 and SEQ No with Error field with the missing sequence number for the retransmission packet, so that the receiving end can detect missing packets. In this case the full error recovery cannot be performed. However, the receiving side can at least detect the loss of packets.

The receiving side should wait at least 4 times the sum of remote Tdetect, local Tdetect and the transmission time of the error message packet, plus the retransmission packet, before it times out when it is waiting for the retransmission packet. When it times out, the receiver has an option of re-requesting it by

sending another error message packet with error type = 0x09, or simply dropping it and reporting it to the higher layer.

5.2 FRAME FORMAT

BOF(0x7E), CRC-CCITT, and EOF(0x7E) are added as shown below to those basic packets described in this document. When the CRC is transmitted, the least significant byte should be transmitted first.

<i>LSB</i>						<i>MSB</i>
0x7E BOF (8 bits)	Packet Type (8 bits)	SEQ No (8 bits)Payload....	CRC (16 bits)	0x7E EOF (8 bits)	

5.3 ERROR MESSAGE PACKET

The error-message packet format is the following:

<i>LSB</i>				<i>MSB</i>
Packet Type, 0x05 (8-bit field)	Sequence No (8-bit field)	Error Type (8-bit field)	SEQ No with Error (8-bit field)	

Error Type	Description
0x00	Reserved
0x01	Overrun Error
0x02	Parity Error
0x03	Reserved
0x04	Framing Error
0x05-0x07	Reserved
0x08	CRC Error
0x09	Missing SEQ No
0x0A-0x80	Reserved
0x81	Missing Retransmission Packet
0x82- 0xFF	Reserved

Table 5.1: Error Type available

5.4 CONSISTENT OVERHEAD BYTE STUFFING

Code(n)	Followed by	Description
0x00		Unused.
0x01-0xCF	n-1 data bytes	The n-1 data bytes plus implicit trailing zero.
0xD0	n-1 data bytes	The n-1 data bytes without trailing zero.
0xD1		Unused.
0xD2		Reserved for future.
0xD3-0xDF	none	A run of n-0xD0 zeros.
0xE0-0xFE	n-E0 data bytes	The data bytes with two trailing zeros.
0xFF		Unused.

Table 5.2:

The COBS requires two step encodes.

The first step is the zero-elimination step. This step takes place after attaching the 16-bit CRC if CRC is enabled, but before adding the beginning and ending delimiters, 0x7E. Each COBS code block consists of the COBS code followed by zero or more data bytes. Code bytes 0x00, 0xD1, 0xD2 and 0xFF are never used. The COBS zero-elimination procedure searches the packet for the first occurrence of value zero. To simplify the encoding, a zero is added temporarily at the end of the packet, after the CRC, as a temporary place holder. The number of octets up to and including the first zero determines the code to be used. If this number is 207 or less, then the number itself is used as a COBS code byte, followed by the actual non-zero data bytes themselves, excluding the last byte, which is zero. On the other hand, if the number is more than 207, then the code byte 0xD0 is used, followed by the first 207 non-zero bytes. This process is repeated until all of the bytes of the packet, including the temporary place-holding zero at the end, have been encoded. If a pair of 0x00 is detected after 0 to 30 non-zero octets, the count of octets plus 0xE0 is used as the COBS code, followed by the non-zero octets, excluding the pair of zeros. If a run of three to fifteen 0x00 octets are detected, then the count of these 0x00 octets, plus 0xD0, is used as the code, followed by no other bytes.

The second step is replacing 0x7E with 0x00. The two steps can be done together in a loop, to reduce the encoding time.

For more details and a reference code, please refer to “PPP Consistent Overhead Byte Stuffing (COBS)” by J. Carlson et al [2].

6 USING RTS/CTS FOR SYNCHRONIZATION

This section illustrates how RTS and CTS are used to resynchronize, and how error-recovery procedure is performed if RTS and CTS are used as a mechanism to synchronize. This is described using protocol mode 0x14.

6.1 USING RTS/CTS FOR SYNC WITHOUT CRC, PROTOCOL MODE 0X14

The flow of HCI packet transfer is handled by two MODEM control/status signals, -RTS and -CTS. -RTS and -CTS are connected in a null MODEM fashion, meaning that the local-side -RTS should be connected to the remote-side -CTS, and the local-side -CTS should be connected to the remote-side -RTS. These MODEM control/status signals are used to notify the detection of an error to the other side, as well as to resynchronize the beginning of the packet after an error is detected. A very simple error-recovery scheme is adapted here to minimize the overhead of supporting this.

The HCI packet is transmitted only while CTS bit is 1. If the CTS bit changes to 0 during the HCI packet transfer or after the last byte is transmitted, this indicates that there was some error. The receiving end will deassert RTS as soon as it detects any error, and should send the error packet with an error type back to the transmission side. This error packet will contain a Sequence Number with Error field that indicates in which packet the error was detected. The sequence number field that is on every packet is an 8-bit field that is incremented by 1 each time any type of packet is transmitted, except for the retransmission packets. The retransmitted packets should contain the original sequence number in the SEQ No field.

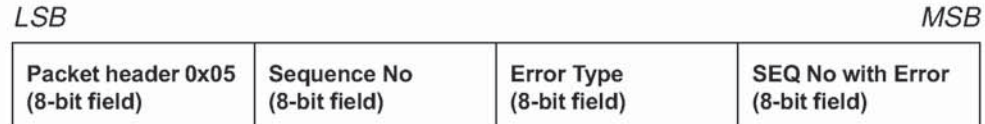
When the transmitting end detects CTS bit changing from 1 to 0 at any time, the transmitting end should hold the transmission and wait until the error packet is received before resuming the transmission. When the receiving end is ready to receive the new data, it should assert RTS after the minimum of Tdetect time. Tdetect time is the maximum time required for the transmit side to detect the state change on CTS bit, plus the time it takes to flush the transmit buffer. The Tdetect value of each side should be informed to the other side during the negotiation phase. The local Tdetect value and the remote side Tdetect value together, along with the baud rate, can also be used to estimate the queue length required for the retransmission holding buffer. Before the receiving side asserts RTS line again, it should flush the RX buffer.

The transmission side should retransmit all of the HCI packets from the packet that had an error, which is indicated by SEQ No with Error field. Before it retransmits, it should flush the transmit buffer that may hold the leftover data from the aborted previous packet. As it retransmits the packets from the retransmission holding buffer, it should start transmitting the packet with the Sequence Number that matches the SEQ No with Error. If the transmitting side doesn't have the packet with the correct sequence number in the retransmission holding buffer, the transmitter should send an error message packet with

error type 0x81, and it should skip to the packet with the sequence number that is available in the buffer. In this case, the full error recovery cannot be performed. However, the receiving side can, at least, detect the loss of packets.

6.2 ERROR MESSAGE PACKET

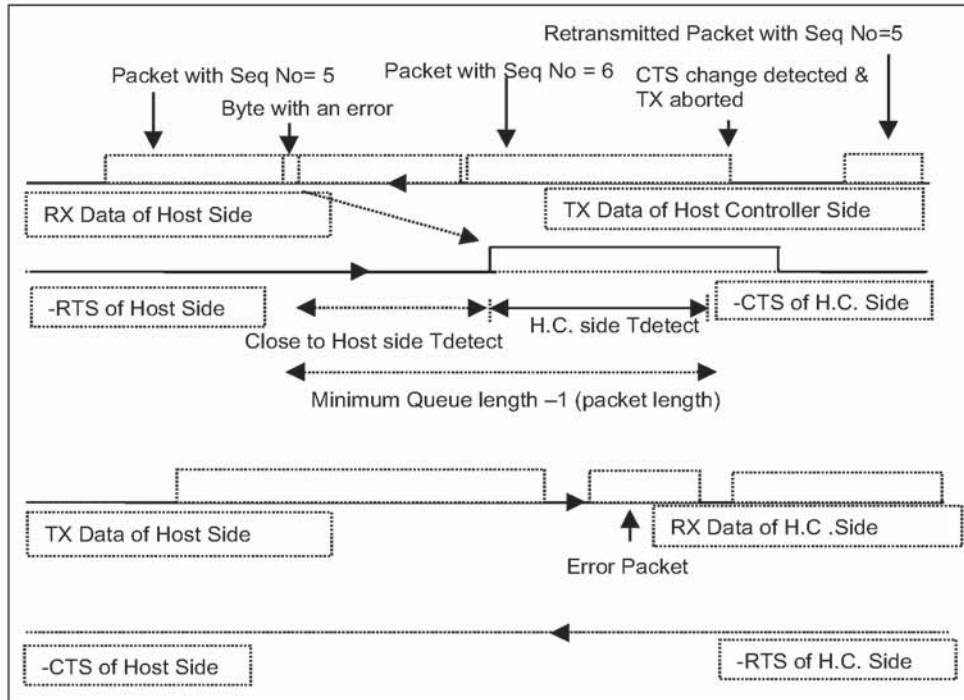
The error-message packet format is the following:



Error Type	Description	Comment
0x00	Reserved	
0x01	Overrun Error	
0x02	Parity Error	
0x03	Reserved	
0x04	Framing Error	
0x05-0x07	Reserved	
0x08	CRC Error*	Not applicable In Mode 0x14
0x09	Missing SEQ No	
0x0A-0x80	Reserved	
0x81	Missing Retransmission Packet	
0x82- 0xFF	Reserved	

Table 6.1: Error Type available

6.3 EXAMPLE OF SIGNALLING



6.4 CONTROL FLOW EXAMPLES

6.4.1 Case 1, Normal Recovery Process

Controller Side	Host Side
0) RTS is asserted and the asserted CTS is detected.	0) RTS is asserted and the asserted CTS is detected.
	1) Ctrl/Data[n] is sent out and Ctrl/Data[n] is stored in the retransmission holding buffer.
2) Ctrl/Data[n] is received with an error.	
3) Deasserts RTS	
4a) Error message for [n] is sent and Error message for [n] is stored in the TX retransmit holding buffer.	4) Detects CTS deasserted.
4b) Empties the RX FIFO and waits for Tdetect (Host) amount of time.	
	5a) Stops further transmission and waits until the TX FIFO is empty (or Flush the FIFO if it can.)
	5b) Error message for [n] is received.
6) Asserts RTS	
.	7) The asserted CTS is detected.
	8) Retransmits Ctrl/Data[n].

6.4.2 Case 2, Both sides detect an error simultaneously

Controller Side	Host Side
0) RTS is asserted and the asserted CTS is detected.	0) RTS is asserted and the asserted CTS is detected.
1) Ctrl/Data[x] is sent and Ctrl/Data[x] is stored in the retransmission holding buffer.	1) Ctrl/Data[n] is sent and Ctrl/Data[n] is stored in the retransmission holding buffer.
2) Ctrl/Data[n] is received with an error.	2) Ctrl/Data[x] is received with an error.
3) Deasserts RTS.	3) Deasserts RTS.
4) Detects CTS deasserted.	4) Detects CTS deasserted.
5a) Stops further transmission and waits until the TX FIFO is empty (or Flush the FIFO if it can).	5a) Stops further transmission and waits until the TX FIFO is empty (or Flush the FIFO if it can).

Controller Side	Host Side
5b) Empties the RX FIFO and waits for Tdetect (Host) amount of time.	5b) Empties the RX FIFO and waits for Tdetect (Controller) amount of time.
6) Asserts RTS.	6) Asserts RTS.
7) The asserted CTS is detected.	7) The asserted CTS is detected.
8) Error message for [n] is sent and Error message for [n] is stored in the TX retransmit holding buffer.	8) Error message for [x] is sent and Error message for [x] is stored in the TX retransmit holding buffer.
9) Error message for [x] is received.	9) Error message for [n] is received.
10) Retransmits Ctrl/Data[x].	10) Retransmits Ctrl/Data[n].

6.4.3 Case 3, Error Message with an error

Controller Side	Host Side
0) RTS is asserted and the asserted CTS is detected.	0) RTS is asserted and the asserted CTS is detected.
2) Ctrl/Data[n] is received with an error.	1) Ctrl/Data[n] is sent and Ctrl/Data[n] is stored in the retransmission holding buffer.
3) Deasserts RTS.	
4a) Error message for [n] (Err[n]) is sent and Err[n] is stored in the TX retransmit holding buffer.	4) Detects CTS deasserted.
4b) Empties the RX FIFO and waits for Tdetect (Host) amount of time.	5a) Stops further transmission and waits until the TX FiFO is empty (or Flush the FIFO if it can.) 5b) Error message for [n] is received with an error.
6) Asserts RTS.	6a) Deasserts RTS. 6b) Empties the RX FIFO and waits for Tdetect (Controller) amount of time.
7) Detects CTS deasserted.	
8) Stops further transmission and waits until the TX FiFO is empty (or Flush the FIFO if it can.)	8) The asserted CTS detected.

Controller Side	Host Side
	9a) Error message for Err[n] is sent and Error message for Err[n] is stored in the retransmission holding buffer. 9b) Asserts RTS.
10a) Error message for Err[n] is received. 10b) The asserted CTS detected.	
11) Retransmits Error message for [n].	
	12) Error message for [n] is received.
	13) Retransmit Ctrl/Data[n].

7 REFERENCES

- [1] Bluetooth Host Controller Interface Function Specification
- [2] J. Carlson, S. Cheshire, M. Baker, draft-ietf-pppext-cobs-00, "PPP Consistent Overhead Byte Stuffing (COBS)", November, 1997
- [3] Bluetooth HCI UART Transport Layer Specification

Part H:4

HCI UART TRANSPORT LAYER

An addendum to the HCI document

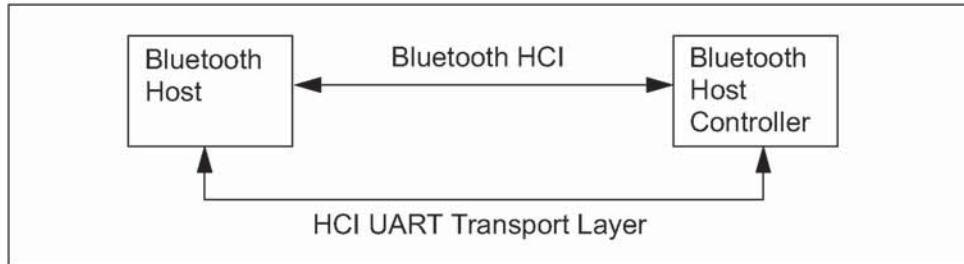
This document describes the UART transport layer (between the Host and the Host Controller). HCI command, event and data packets flow through this layer, but the layer does not decode them.

CONTENTS

1	General.....	798
2	Protocol.....	799
3	RS232 Settings	800
4	Error Recovery	801

1 GENERAL

The objective of this HCI UART Transport Layer is to make it possible to use the Bluetooth HCI over a serial interface between two UARTs on the same PCB. The HCI UART Transport Layer assumes that the UART communication is free from line errors. See also "HCI RS232 Transport Layer" on page 775.



2 PROTOCOL

There are four kinds of HCI packets that can be sent via the UART Transport Layer; i.e. HCI Command Packet, HCI Event Packet, HCI ACL Data Packet and HCI SCO Data Packet (see "Host Controller Interface Functional Specification" on page 517). HCI Command Packets can only be sent to the Bluetooth Host Controller, HCI Event Packets can only be sent from the Bluetooth Host Controller, and HCI ACL/SCO Data Packets can be sent both to and from the Bluetooth Host Controller.

HCI does not provide the ability to differentiate the four HCI packet types. Therefore, if the HCI packets are sent via a common physical interface, a HCI packet indicator has to be added according to Table 2.11 below.

HCI packet type	HCI packet indicator
HCI Command Packet	0x01
HCI ACL Data Packet	0x02
HCI SCO Data Packet	0x03
HCI Event Packet	0x04

Table 2.1: HCI packet indicators

The HCI packet indicator shall be sent immediately before the HCI packet. All four kinds of HCI packets have a length field, which is used to determine how many bytes are expected for the HCI packet. When an entire HCI packet has been received, the next HCI packet indicator is expected for the next HCI packet. Over the UART Transport Layer, only HCI packet indicators followed by HCI packets are allowed.

3 RS232 SETTINGS

The HCI UART Transport Layer uses the following settings for RS232:

Baud rate:	manufacturer-specific
Number of data bits:	8
Parity bit:	no parity
Stop bit:	1 stop bit
Flow control:	RTS/CTS
Flow-off response time:	3 ms

Flow control with RTS/CTS is used to prevent temporary UART buffer overrun. It should not be used for flow control of HCI, since HCI has its own flow control mechanisms for HCI commands, HCI events and HCI data.

If CTS is 1, then the Host/Host Controller is allowed to send.
If CTS is 0, then the Host/Host Controller is not allowed to send.

The flow-off response time defines the maximum time from setting RTS to 0 until the byte flow actually stops.

The RS232 signals should be connected in a null-modem fashion; i.e. the local TXD should be connected to the remote RXD and the local RTS should be connected to the remote CTS and vice versa.

4 ERROR RECOVERY

If the Host or the Host Controller lose synchronization in the communication over RS232, then a reset is needed. A loss of synchronization means that an incorrect HCI packet indicator has been detected, or that the length field in an HCI packet is out of range.

If the UART synchronization is lost in the communication from Host to Host Controller, then the Host Controller shall send a Hardware Error Event to tell the Host about the synchronization error. The Host Controller will then expect to receive an HCI_Reset command from the Host in order to perform a reset. The Host Controller will also use the HCI_Reset command in the byte stream from Host to Host Controller to re-synchronize.

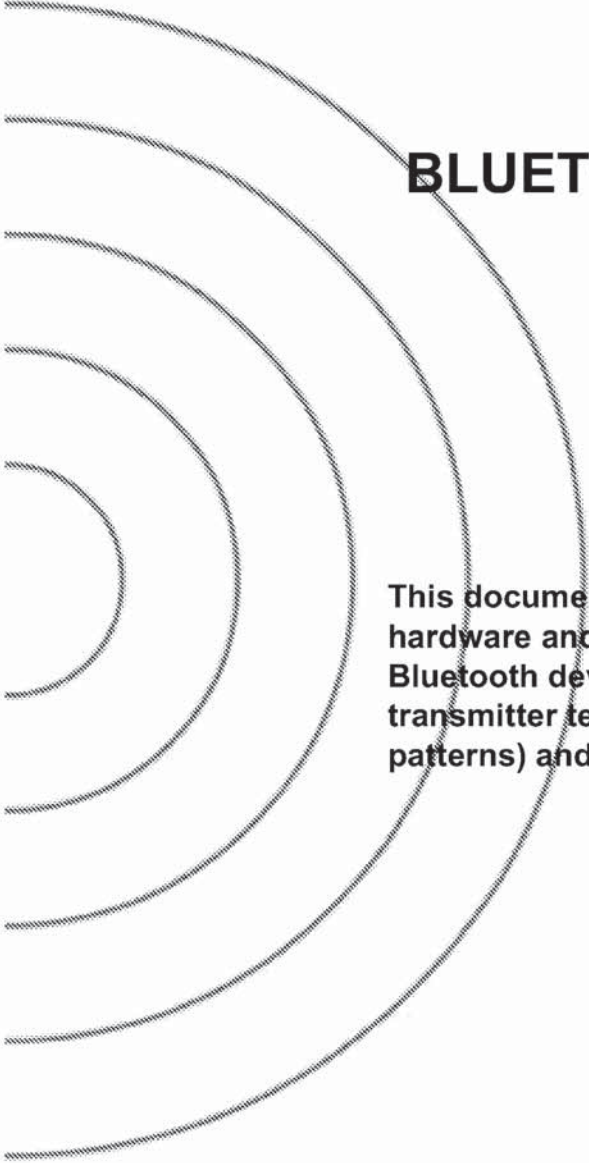
If the UART synchronization is lost in the communication from Host Controller to Host, then the Host shall send the HCI_Reset command in order to reset the Host Controller. The Host shall then re-synchronize by looking for the HCI Command Complete event for the HCI_Reset command in the byte stream from Host Controller to Host.

See "Host Controller Interface Functional Specification" on page 517 for HCI commands and HCI events.



Part I:1

BLUETOOTH TEST MODE



This document describes the test mode for hardware and low-level functionality tests of Bluetooth devices. The test mode includes transmitter tests (packets with constant bit patterns) and loop back tests.

CONTENTS

1	General Description	806
1.1	Test Setup	806
1.2	Activation	807
1.3	Control	807
2	Test Scenarios	808
2.1	Transmitter Test	808
2.1.1	Packet Format	808
2.1.2	Pseudorandom Sequence	810
2.1.3	Reduced Hopping Sequence	811
2.1.4	Control of Transmit Parameters	811
2.1.5	Power Control.....	812
2.1.6	Switch between different Frequency Settings	812
2.2	LoopBack Test	812
3	Outline of Proposed LMP Messages	817
4	References	819

1 GENERAL DESCRIPTION

The test mode supports testing of the Bluetooth transmitter and receiver. It is intended mainly for certification/compliance testing of the radio and baseband layer, and may also be used for regulatory approval or in-production and after-sales testing.

A device in test mode must not support normal operation. For security reasons the test mode is designed such that it offers no benefit to the user. Therefore, no data output or acceptance on a HW or SW interface is allowed.

1.1 TEST SETUP

The setup consists of a device under test (DUT) and a tester. Optionally, additional measurement equipment may be used.

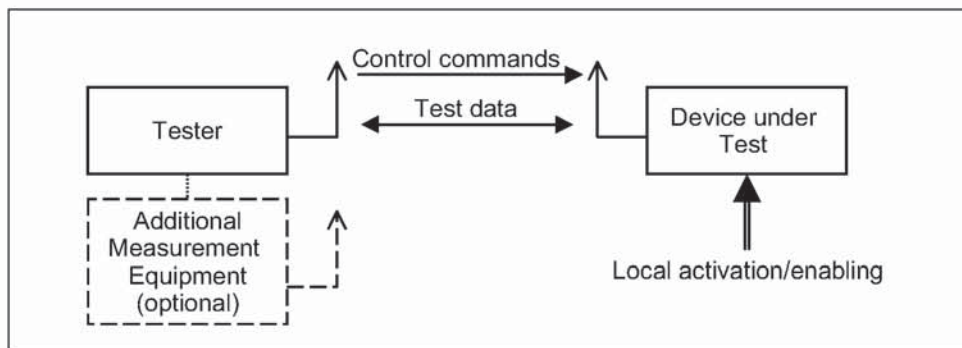


Figure 1.1: Setup for Test Mode

Tester and DUT form a piconet where the tester acts as master and has full control over the test procedure. The DUT acts as slave.

The control is done via the air interface using LMP commands (see Section 3 on page 817 and “Link Manager Protocol” on page 185). Hardware interfaces to the DUT may exist, but are not subject to standardization.

The test mode is a special state of the Bluetooth model. For security and type approval reasons, a device in test mode may not support normal operation. When the DUT leaves the test mode it enters the standby state. After power-off the Bluetooth device must return to standby state.

1.2 ACTIVATION

The activation may be carried out locally (via a HW or SW interface), or using the air interface.

- For activation over the air interface, entering the test mode must be locally enabled for security and type approval reasons. The implementation of this local enabling is not subject to standardization.

The tester sends an LMP command that forces the DUT to enter test mode. The DUT terminates all normal operation before entering the test mode.

The DUT shall return an LMP_Accepted on reception of an activation command. LMP_Not_Accepted shall be returned if the DUT is not locally enabled.

- If the activation is performed locally using a HW or SW interface, the DUT terminates all normal operation before entering the test mode.

Until a connection to the tester exists, the device shall perform page scan and inquiry scan. Extended scan activity is recommended.

1.3 CONTROL

Control and configuration is performed using special LMP commands (see Section 3 on page 817). These commands must be rejected if the Bluetooth device is not in test mode. In this case, an LMP_not_accepted is returned. The DUT shall return an LMP_accepted on reception of a control command when in test mode.

A Bluetooth device in test mode must ignore all LMP commands not related to control of the test mode. LMP commands dealing with power control and the request for LMP features (LMP_features_req) are allowed in test mode; the normal procedures are also used to test the adaptive power control.

The DUT can be commanded to leave the test mode by an LMP_Detach command or by sending an LMP_test_control command with test scenario set to 'exit test mode'.

2 TEST SCENARIOS

2.1 TRANSMITTER TEST

The Bluetooth device transmits a constant bit pattern. This pattern is transmitted periodically with packets aligned to the slave TX timing of the piconet formed by tester and DUT. The same test packet is repeated for each transmission.

The transmitter test is started when the master sends the first POLL packet. In non-hopping mode agreed frequency is used for this POLL packet.

The tester transmits at his TX slots (control commands or POLL packets). The slave starts burst transmission in the following slave TX slot. The master's polling interval is fixed and defined beforehand. The device under test may transmit its burst according to the normal timing even if no packet from the tester was received.

The burst length may exceed the length of a one slot packet. In this case the tester may take the next free master TX slot for polling. The timing is illustrated in Figure 2.1.

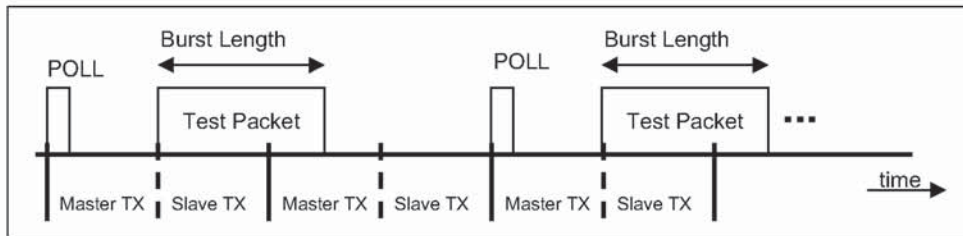


Figure 2.1: Timing for Transmitter Test

2.1.1 Packet Format

The test packet is a normal Bluetooth packet, see Figure 2.2. For the payload itself see below.

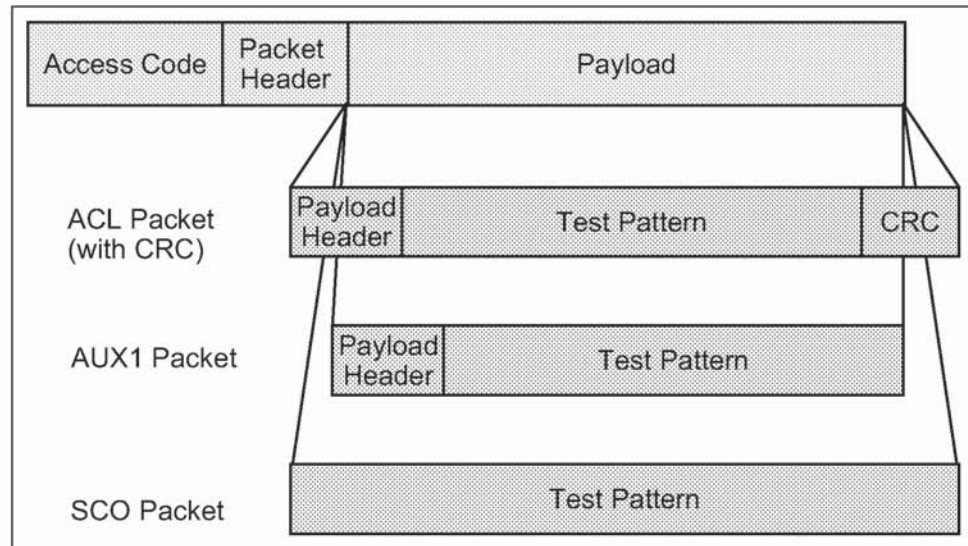


Figure 2.2: General Format of TX Packet

During configuration the tester defines:

- the packet type to be used
- payload length

For the payload length, the restrictions from the baseband specification apply (see "Baseband Specification" on page 33.). In case of ACL packets the payload structure defined in the baseband specification is preserved as well, see Figure 2.2.

For the transmitter test mode, only packets without FEC should be used; i.e. HV3, DH1, DH3, DH5 and AUX1 packets. Support of packet type is only mandatory up to the longest implemented packet type.

In transmitter test mode, the packets exchanged between tester and DUT are not scrambled with the whitening sequence. Whitening is turned off when the DUT has accepted to enter the transmitter test mode, and is turned on when the DUT has accepted to exit the transmitter test mode, see Figure 2.3.¹

1. Note: Implementations must ensure that retransmissions of the LMP_Accepted messages use the same whitening status.

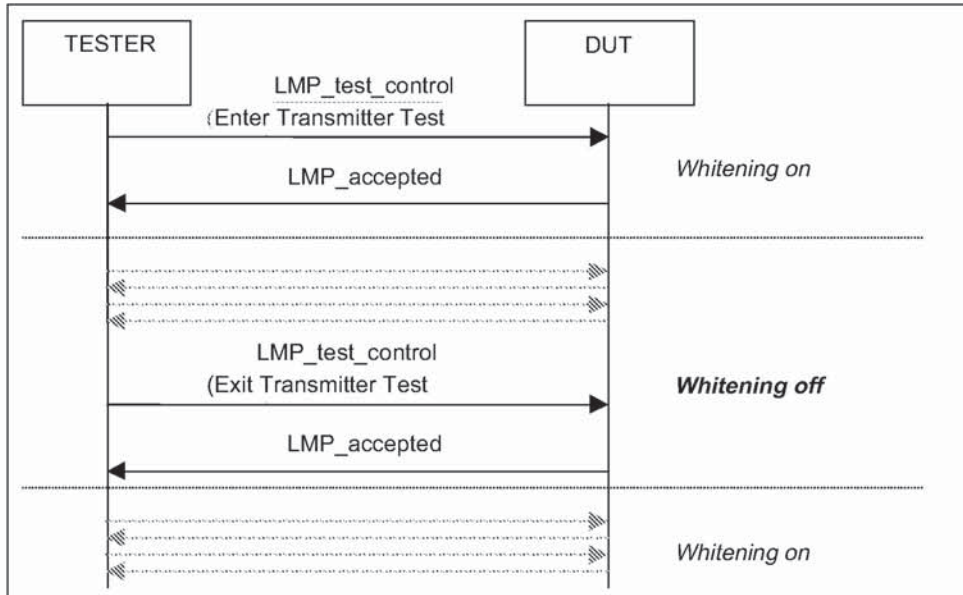


Figure 2.3: Use of whitening in Transmitter mode

2.1.2 Pseudorandom Sequence

In case of pseudorandom bit sequence, the same sequence of bits is used for each transmission (i.e. the packet is repeated, see above). A PRBS-9 Sequence² is used, see [2] and [3].

The properties of this sequence are as follows (see [3]). The sequence may be generated in a nine-stage shift register whose 5th and 9th stage outputs are added in a modulo-two addition stage (see Figure 2.4), and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 9 consecutive ONES; i.e. the shift register is initialized with nine ones.

- Number of shift register stages: 9
- Length of pseudo-random sequence: $2^9 - 1 = 511$ bits
- Longest sequence of zeros: 8 (non-inverted signal)

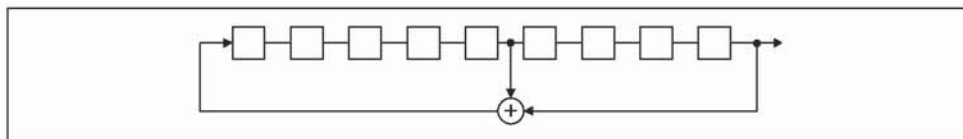


Figure 2.4: Linear Feedback Shift Register for Generation of the PRBS sequence

2. Some uncertainties about Japanese regulatory requirements have been reported. If necessary for regulatory type approval in Japan, some features might be added; e.g. a longer PN sequence.

2.1.3 Reduced Hopping Sequence

To support quick testing of the radio over the complete frequency range, a reduced hopping mode is defined. Implementation of this mode is optional for Bluetooth devices and modules.

Reduced hopping uses only five frequencies, on which a sequential hopping is done (channels 0, 23, 46, 69 and 93 are used³), see Figure 2.5.

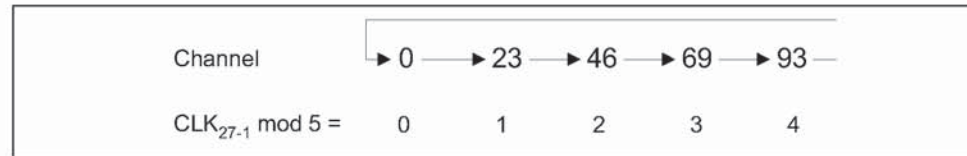


Figure 2.5: Reduced hopping scheme

The timing is based on the Bluetooth clock of the tester. The value of CLK_{27-1} (i.e. not using CLK_0 , representing half slots) modulo 5 is used to determine the transmit frequency.

2.1.4 Control of Transmit Parameters

The following parameters can be set to configure the transmitter test:

1. Bit pattern:
 - Constant zero
 - Constant one
 - Alternating 1010...⁴
 - Alternating 1111 0000 1111 0000...⁴
 - Pseudorandom bit pattern
 - Transmission off
 2. Frequency selection:
 - Single frequency
 - Hopping Europe/USA
 - Hopping Japan
 - Hopping France
 - Hopping Spain
 - Reduced Hopping (implementation in Bluetooth devices and modules is optional)
 3. TX frequency
 - $k \Rightarrow f := (2402 + k)$ MHz
3. The range is chosen to test the whole frequency range, which covers the normal 79 channels, as well as Spanish, French and Japanese hopping schemes. The frequency assignment rule is the same as for the fixed TX frequency: $f = (2402 + k)$ MHz.
4. It is recommended that the sequence starts with a one; but, as this is irrelevant for measurements, it is also allowed to start with a zero.

4. Default poll period in TDD frames ($n * 1.25$ ms)
5. Packet Type
6. Length of Test Sequence (user data of packet definition in Baseband Specification" on page 33.)

2.1.5 Power Control

If adaptive power control is tested, the normal LMP commands will be used. The DUT starts to transmit at the maximum power and reduces/increases its power by one step on every command received.

2.1.6 Switch between different Frequency Settings

A change in the frequency selection becomes effective when the LMP procedure is completed:

The tester switches to a new frequency or hopping pattern after the LMP_Accepted message has been received.

- | The DUT switches after the LMP_accepted message has been sent.

Note: Loss of the LMP_Accepted packet will eventually lead to a loss of frequency synchronization that cannot be recovered. Similar problems occur in normal operation, when the hopping pattern changes.

2.2 LOOPBACK TEST

The device under test receives normal baseband packets. The received packets are decoded in the DUT, and the payload is sent back using the same packet type. The return packet is sent back in either the TX slot directly following the transmission of the tester, or it is delayed and sent back in the slot after the next transmission of the tester (see Figure 2.7 to Figure 2.9 on page 815).

Alternatively, it is possible to implement a delayed loopback instead. Then the return packet is delayed to the following TX slot. There is no signalling to determine or control the mode. The device behavior must be fixed or adjusted by other means, but must not change randomly.

- | The tester can select, whether whitening is on or off. This setting holds for both up- and downlink. For switching the whitening status, the same rules as in Section 2.1 on page 808 (Figure 2.3) apply.

The following rules apply (for illustration see Figure 2.6 on page 814):

- Clearly, if the synch word was not detected, there will be no reply.
- If the header error check (HEC) fails, the DUT replies with a NULL packet with the ARQN bit set to NAK. It is not mandatory to return a NULL packet in this case; the DUT may send nothing.

- If the packet contains an LMP message relating to the control of the test mode this command is executed and the packet is not returned, though ACK or NAK is still returned as usual procedure. Other LMP commands are ignored and no packet is returned.
- The payload FEC is decoded and the payload is coded again for transmission. This allows testing of the FEC handling. If the pure bit error rate shall be determined the tester chooses a packet type without FEC.
- The CRC is evaluated. In case of a failure, the payload is returned with ARQN = NAK. The CRC for the return packet is calculated for the returned payload.
- If the CRC fails the number of bytes as indicated in the (possibly erroneous) payload header shall be looped back.

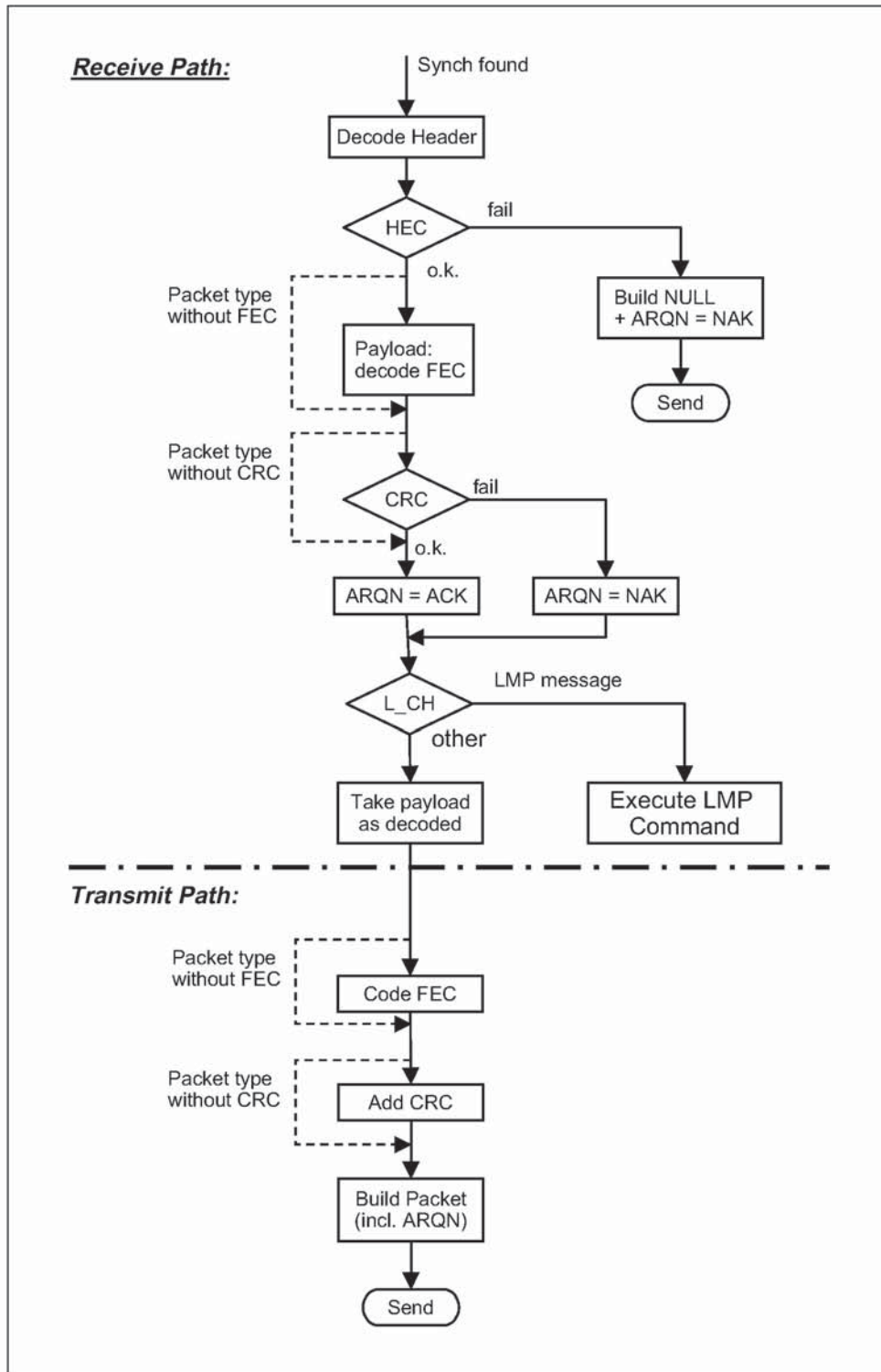


Figure 2.6: DUT Packet Handling in Loop Back Test

The timing for normal and delayed loopback is illustrated in Figure 2.7 to Figure 2.9:

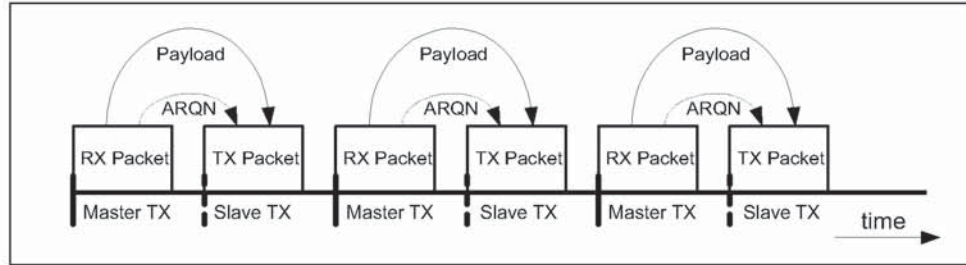


Figure 2.7: Payload & ARQN handling in normal loopback.

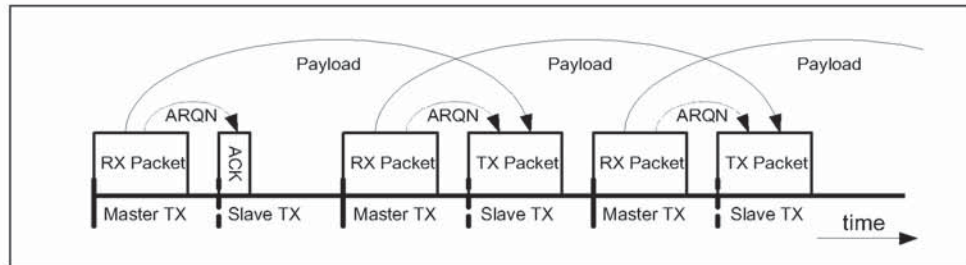


Figure 2.8: Payload & ARQN handling in delayed loopback - start.

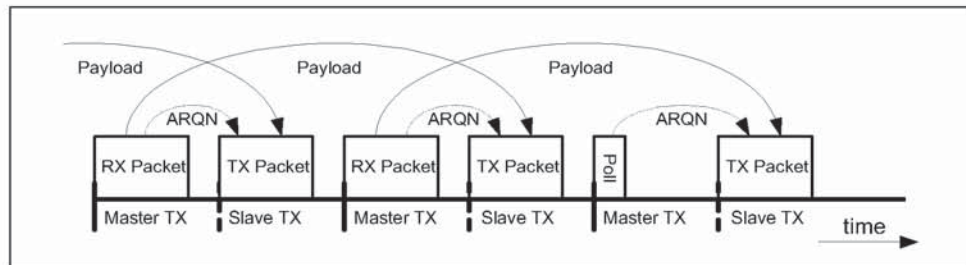


Figure 2.9: Payload & ARQN handling in delayed loopback - end.

The whitening is performed in the same way as it is used in normal active mode.

The following parameters can be set to configure the loop back test:

1. Packet Class⁵
 - ACL Packets
 - SCO Packets
 - ACL Packets without whitening
 - SCO Packets without whitening

5. This is included because, in the future, the packet type numbering may not remain unambiguous.

2. Frequency Selection

- Single frequency (independent for RX and TX)
- Hopping Europe/USA
- Hopping Japan
- Hopping France
- Hopping Spain
- Hopping reduced (optional)

Hopping reduced uses only five frequencies on which a sequential hopping is done on (channel: 0, 23, 46, 69 and 93 is used).

3. Power level: (To be used according radio specification requirements)

- power control or fixed TX power

The switch of the frequency setting is done exactly as for the transmitter test (see Section 2.1.6 on page 812).

3 OUTLINE OF PROPOSED LMP MESSAGES

Table 3.1 lists all LMP messages used for test mode (see Link Manager Protocol, Section 6 on page 237).

LMP PDU	PDU number	Possible Direction	Contents	Position in Payload
LMP_test_activate	56	m → s		
LMP_test_control	57	m → s	test scenario hopping mode TX frequency RX frequency power control mode poll period packet type length of test data	2 3 4 5 6 7 8 9-10
LMP_detach	7	m → s		
LMP_accepted	3	m ← s		
LMP_not_accepted	4	m ← s		

Table 3.1: LMP messages used for Test Mode

Name	Length (bytes)	Type	Unit	Detailed
Test scenario	1	u_int8		0 Pause (TX off) 1 Transmitter test – 0 pattern 2 Transmitter test – 1 pattern 3 Transmitter test – 1010 pattern 4 Pseudorandom bit sequence 5 Closed Loop Back – ACL packets 6 Closed Loop Back – SCO packets 7 ACL Packets without whitening 8 SCO Packets without whitening 9 Transmitter test – 1111 0000 pattern 10–254 reserved 255 Exit Test Mode
Hopping mode	1	u_int8		0 RX/TX on single frequency 1 Hopping Europe/USA 2 Hopping Japan 3 Hopping France 4 Hopping Spain 5 Reduced Hopping (optional) 6–255 reserved
TX frequency (for DUT)	1	u_int8		$f = [2402 + k] \text{ MHz}$

Table 3.2: Parameters used in LMP_Test_Control PDU

Name	Length (bytes)	Type	Unit	Detailed
RX frequency (for DUT)	1	u_int8		f = [2402 + k] MHz
Power control mode	1	u_int8		0 fixed TX output power 1 adaptive power control
Poll period	1	u_int8	1.25 ms	
Packet type	1	u_int8		numbering as in packet header, see Baseband Specification)
length of test sequence (=length of user data in Baseband Specification)	2	u_int16	1 byte	unsigned binary number

Table 3.2: Parameters used in LMP_Test_Control PDU

The control PDU is used for both transmitter and loop back tests. The following restrictions apply for the parameter settings:

Parameter	Restrictions Transmitter Test	Restrictions Loopback Test
TX frequency	$0 \leq k \leq 93$	$0 \leq k \leq 93$
RX frequency	same as TX frequency	$0 \leq k \leq 93$
Poll period		not applicable (set to 0)
Length of test sequence	depends on packet type: DH1: ≤ 28 byte DH3: ≤ 181 byte DH5: ≤ 339 byte AUX1: ≤ 29 Byte HV3: = 30 byte	not applicable (set to 0)

Table 3.3: Restrictions for Parameters used in LMP_Test_Control PDU

4 REFERENCES

- [1] Bluetooth Link Manager Protocol.
- [2] CCITT Recommendation O.153 (1992), Basic parameters for the measurement of error performance at bit rates below the primary rate.
- [3] ITU-T Recommendation O.150 (1996), General requirements for instrumentation for performance measurements on digital transmission equipment.
- [4] Bluetooth Baseband Specification.



Part 1:2

**BLUETOOTH COMPLIANCE
REQUIREMENTS**

This document specifies the requirements for
Bluetooth compliance.



CONTENTS

1	Scope	825
2	Terms used	826
3	Legal aspects	828
4	The value of the Bluetooth Brand.....	829
5	The Bluetooth qualification program	830
6	Bluetooth license requirements for products	832
6.1	Bluetooth radio link requirements.....	832
6.1.1	Requirement description	832
6.1.2	Qualification.....	832
6.2	Bluetooth protocol requirements	832
6.2.1	Qualification.....	833
6.3	Bluetooth profile requirements	833
6.3.1	Requirement description	833
6.3.2	Qualification.....	834
6.4	Bluetooth information requirements	834
6.4.1	Requirement description	834
6.4.2	Qualification.....	834
6.5	Requirements on Bluetooth accessory products.....	834
6.5.1	Definition of 'Bluetooth accessory products'.....	834
6.5.2	Qualification.....	834
6.6	Requirements on Bluetooth components	834
6.6.1	Definition of "Bluetooth components"	834
6.6.2	Requirement description	835
6.6.3	Qualification.....	835
7	Bluetooth license provisions for early products.....	836

8	Bluetooth Brand License provisions for special products & marketing	837
8.1	Bluetooth Development tools and demos	837
8.1.1	Definition of 'Bluetooth Development tools and demos'	837
8.1.2	Requirement description	837
8.1.3	Qualification	837
8.2	Marketing	837
9	Recommendations concerning information about a product's Bluetooth capabilities	838
10	Quality management, configuration management and version control	839
11	Appendix A – Example of a “Bluetooth Capability Statement” ...	840
12	Appendix B - Marketing names of Bluetooth profiles	841

1 SCOPE

The Bluetooth Promoters and the Bluetooth Adopters have signed the Promoters' Agreement and the Adopters' Agreement respectively. These agreements grant Promoters and Adopters a Bluetooth license for "products which comply with the Specification".

This document specifies the requirements which must be met by a Promoter or Adopter to demonstrate that a particular product does "comply with the Specification", thereby qualifying that particular product to be subject to the rights extended by the Promoters' and Adopters' Agreements respectively.

The Bluetooth Qualification Program is the process by which a Promoter or an Adopter demonstrates that a particular product meets the requirements specified herein. This document provides an overview of the requirements and the Bluetooth Qualification Program. Further details are available through the Bluetooth Web site.

Regulatory requirements and governmental type approval requirements are outside the scope of this document.