NAME

ng_hci - Netgraph node type that is also a Bluetooth Host Controller Interface (HCI) layer

SYNOPSIS

#include <sys/types.h>
#include <netgraph/bluetooth/include/ng_hci.h>

DESCRIPTION

The **hci** node type is a Netgraph node type that implements Bluetooth Host Controller Interface (HCI) layer as per chapter H1 of the Bluetooth Specification Book v1.1.

INTRODUCTION TO BLUETOOTH

Bluetooth is a short-range radio link intended to replace the cable(s) connecting portable and/or fixed electronic devices. Bluetooth operates in the unlicensed ISM band at 2.4 GHz. The Bluetooth protocol uses a combination of circuit and packet switching. Bluetooth can support an asynchronous data channel, up to three simultaneous synchronous voice channels, or a channel which simultaneously supports asynchronous data and synchronous voice. Each voice channel supports a 64 kb/s synchronous (voice) channel in each direction. The asynchronous channel can support maximal 723.2 kb/s asymmetric (and still up to 57.6 kb/s in the return direction), or 433.9 kb/s symmetric.

The Bluetooth system provides a point-to-point connection (only two Bluetooth units involved), or a point-to-multipoint connection. In the point-to-multipoint connection, the channel is shared among several Bluetooth units. Two or more units sharing the same channel form a "piconet". One Bluetooth unit acts as the master of the piconet, whereas the other unit(s) acts as slave(s). Up to seven slaves can be active in the piconet. In addition, many more slaves can remain locked to the master in a so-called parked state. These parked slaves cannot be active on the channel, but remain synchronized to the master. Both for active and parked slaves, the channel access is controlled by the master.

Multiple piconets with overlapping coverage areas form a "scatternet". Each piconet can only have a single master. However, slaves can participate in different piconets on a time-division multiplex basis. In addition, a master in one piconet can be a slave in another piconet. The piconets shall not be frequency-synchronized. Each piconet has its own hopping channel.

Time Slots

The channel is divided into time slots, each 625 usec in length. The time slots are numbered according to the Bluetooth clock of the piconet master. The slot numbering ranges from 0 to 2^27 -1 and is cyclic with a cycle length of 2^27. In the time slots, master and slave can transmit packets.

SCO Link

The SCO link is a symmetric, point-to-point link between the master and a specific slave. The SCO link reserves slots and can therefore be considered as a circuit-switched connection between the master and the slave. The SCO link typically supports time-bounded information like voice. The master can support up to three SCO links to the same slave or to different slaves. A slave can support up to three SCO links from the same master, or two SCO links if the links originate from different masters. SCO packets are never retransmitted.

ACL Link

In the slots not reserved for SCO links, the master can exchange packets with any slave on a per-slot basis. The ACL link provides a packet-switched connection between the master and all active slaves participating in the piconet. Both asynchronous and isochronous services are supported. Between a master and a slave only a single ACL link can exist. For most ACL packets, packet retransmission is applied to assure data integrity.

HOST CONTROLLER INTERFACE (HCI)

The HCI provides a command interface to the baseband controller and link manager, and access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband capabilities.

The HCI layer on the Host exchanges data and commands with the HCI firmware on the Bluetooth hardware. The Host Controller Transport Layer (i.e., physical bus) driver provides both HCI layers with the ability to exchange information with each other.

The Host will receive asynchronous notifications of HCI events independent of which Host Controller Transport Layer is used. HCI events are used for notifying the Host when something occurs. When the Host discovers that an event has occurred it will then parse the received event packet to determine which event occurred. The next sections specify the HCI packet formats.

HCI Command Packet

```
#define NG_HCI_CMD_PKT 0x01
typedef struct {
        uint8_t  type;   /* MUST be 0x1 */
        uint16_t opcode; /* OpCode */
        uint8_t  length; /* parameter(s) length in bytes */
} __attribute__ ((packed)) ng_hci_cmd_pkt_t;
```

The HCI command packet is used to send commands to the Host Controller from the Host. When the Host Controller completes most of the commands, a Command Complete event is sent to the Host. Some commands do not receive a Command Complete event when they have been completed. Instead, when the Host Controller receives one of these commands the Host Controller sends a Command Status event back to the Host when it has begun to execute the command. Later on, when the actions associated with the command have finished, an event that is associated with the sent command will be sent by the Host Controller to the Host.

HCI Event Packet

```
#define NG_HCI_EVENT_PKT 0x04
typedef struct {
        uint8_t type;   /* MUST be 0x4 */
        uint8_t event;  /* event */
        uint8_t length; /* parameter(s) length in bytes */
} __attribute__ ((packed)) ng_hci_event_pkt_t;
```

The HCI event packet is used by the Host Controller to notify the Host when events occur.

HCI ACL Data Packet

```
#define NG_HCI_ACL_DATA_PKT 0x02
typedef struct {
        uint8_t  type;       /* MUST be 0x2 */
        uint16_t con_handle; /* connection handle + PB + BC flags */
        uint16_t length;     /* payload length in bytes */
} __attribute__ ((packed)) ng_hci_acldata_pkt_t;
```

HCI ACL data packets are used to exchange ACL data between the Host and Host Controller.

HCI SCO Data Packet

```
#define NG_HCI_SCO_DATA_PKT 0x03
typedef struct {
        uint8_t  type;       /* MUST be 0x3 */
        uint16_t con_handle; /* connection handle + reserved bits */
        uint8_t  length;     /* payload length in bytes */
} __attribute__ ((packed)) ng_hci_scodata_pkt_t;
```

HCI SCO data packets are used to exchange SCO data between the Host and Host Controller.

HCI INITIALIZATION

On initialization, HCI control application must issue the following HCI commands (in any order).

Read_BD_ADDR
  To obtain BD_ADDR of the Bluetooth unit.

Read_Local_Supported_Features

To obtain the list of features supported by Bluetooth unit.

Read_Buffer_Size
: To determine the maximum size of HCI ACL and SCO HCI data packets (excluding header) that can be sent from the Host to the Host Controller. There are also two additional return parameters that specify the total number of HCI ACL and SCO data packets that the Host Controller can have waiting for transmission in its buffers.

As soon as HCI initialization has been successfully performed, HCI control application must turn on "inited" bit for the node. Once HCI node has been initialized all upstream hooks will receive a NGM_HCI_NODE_UP Netgraph message defined as follows.

```
#define NGM_HCI_NODE_UP 112 /* HCI -> Upper */
typedef struct {
        uint16_t  pkt_size; /* max. ACL/SCO packet size (w/o hdr) */
        uint16_t  num_pkts; /* ACL/SCO packet queue size */
        uint16_t  reserved; /* place holder */
        bdaddr_t  bdaddr;   /* bdaddr */
} ng_hci_node_up_ep;
```

HCI FLOW CONTROL

HCI layer performs flow control on baseband connection basis (i.e., ACL and SCO link). Each baseband connection has "connection handle" and queue of outgoing data packets. Upper layers protocols are allowed to send up to (num_pkts - pending) packets at one time. HCI layer will send NGM_HCI_SYNC_CON_QUEUE Netgraph messages to inform upper layers about current queue state for each connection handle. The NGM_HCI_SYNC_CON_QUEUE Netgraph message is defined as follows.

```
#define NGM_HCI_SYNC_CON_QUEUE 113 /* HCI -> Upper */
typedef struct {
        uint16_t con_handle; /* connection handle */
        uint16_t completed;  /* number of completed packets */
} ng_hci_sync_con_queue_ep;
```

HOOKS

This node type supports the following hooks:

drv Bluetooth Host Controller Transport Layer hook. Single HCI packet contained in single *mbuf* structure.

acl Upper layer protocol/node is connected to the hook. Single HCI ACL data packet contained in single *mbuf* structure.

sco Upper layer protocol/node is connected to the hook. Single HCI SCO data packet contained in single *mbuf* structure.

raw
Raw hook. Every HCI frame (including HCI command frame) that goes in or out will be delivered to the hook. Usually the Bluetooth raw HCI socket layer is connected to the hook. Single HCI frame contained in single *mbuf* structure.

BLUETOOTH UPPER LAYER PROTOCOLS INTERFACE (LP CONTROL MESSAGES)

NGM_HCI_LP_CON_REQ
Requests the lower protocol to create a connection. If a physical link to the remote device does not exist, this message must be sent to the lower protocol (baseband) to establish the physical connection.

NGM_HCI_LP_DISCON_REQ
Requests the lower protocol (baseband) to terminate a connection.

NGM_HCI_LP_CON_CFM
Confirms success or failure of the NGM_HCI_LP_CON_REQ request to establish a lower layer (baseband) connection. This includes passing the authentication challenge if authentication is required to establish the physical link.

NGM_HCI_LP_CON_IND
Indicates the lower protocol (baseband) has successfully established incoming connection.

NGM_HCI_LP_CON_RSP
A response accepting or rejecting the previous connection indication request.

NGM_HCI_LP_DISCON_IND
Indicates the lower protocol (baseband) has terminated connection. This could be a response to NGM_HCI_LP_DISCON_REQ or a timeout event.

NGM_HCI_LP_QOS_REQ
Requests the lower protocol (baseband) to accommodate a particular QoS parameter set.

NGM_HCI_LP_QOS_CFM
Confirms success or failure of the request for a given quality of service.

NGM_HCI_LP_QOS_IND
Indicates the lower protocol (baseband) has detected a violation of the QoS agreement.

NETGRAPH CONTROL MESSAGES

This node type supports the generic control messages, plus the following:

NGM_HCI_NODE_GET_STATE
Returns current state for the node.

NGM_HCI_NODE_INIT
Turn on "inited" bit for the node.

NGM_HCI_NODE_GET_DEBUG
Returns an integer containing the current debug level for the node.

NGM_HCI_NODE_SET_DEBUG
This command takes an integer argument and sets current debug level for the node.

NGM_HCI_NODE_GET_BUFFER
Returns current state of data buffers.

NGM_HCI_NODE_GET_BDADDR
Returns BD_ADDR as cached in the node.

NGM_HCI_NODE_GET_FEATURES
Returns the list of features supported by hardware (as cached by the node).

NGM_HCI_NODE_GET_NEIGHBOR_CACHE
Returns content of the neighbor cache.

NGM_HCI_NODE_FLUSH_NEIGHBOR_CACHE
Remove all neighbor cache entries.

NGM_HCI_NODE_GET_CON_LIST
Returns list of active baseband connections (i.e., ACL and SCO links).

NGM_HCI_NODE_GET_STAT
Returns various statistic counters.

NGM_HCI_NODE_RESET_STAT

Resets all statistic counters to zero.

NGM_HCI_NODE_SET_LINK_POLICY_SETTINGS_MASK
Sets current link policy settings mask. After the new ACL connection is created the HCI node will try set link policy for the ACL connection. By default, every supported Link Manager (LM) mode will be enabled. User can override this by setting link policy settings mask which specifies LM modes to be enabled.

NGM_HCI_NODE_GET_LINK_POLICY_SETTINGS_MASK
Returns current link policy settings mask.

NGM_HCI_NODE_SET_PACKET_MASK
Sets current packet mask. When new baseband (ACL or SCO) connection is created the HCI node will specify every packet type supported by the device. User can override this by setting packet mask which specifies packet types to be used for new baseband connections.

NGM_HCI_NODE_GET_PACKET_MASK
Returns current packet mask.

NGM_HCI_NODE_SET_ROLE_SWITCH
Sets the value of the role switch. Role switch is enabled when this value is not zero. This is the default state. Note that actual role switch at Bluetooth link level will only be performed if hardware supports role switch and it was enabled.

NGM_HCI_NODE_GET_ROLE_SWITCH
Returns the value of the role switch for the node.

SHUTDOWN

This node shuts down upon receipt of a NGM_SHUTDOWN control message, or when all hooks have been disconnected.

SEE ALSO

netgraph(4), hccontrol(8), ngctl(8)

HISTORY

The **hci** node type was implemented in FreeBSD 5.0.

AUTHORS

Maksim Yevmenkin <*m_evmenkin@yahoo.com*>

BUGS

Most likely. Please report if found.