AT Command

INTRODUCTION

Note: this AT command instruction only applies to UART TTL Bluetooth Module – BC4 or L6. We do not guarantee it can apply to any Bluetooth module.

HHW-SPP embedded Bluetooth serial module has two modes: command response mode and auto connection mode. In the auto connection mode, there are three job roles as the master, the slave and the loopback. When the module works in auto connection mode, it will automatically connect to other Bluetooth devices with SPP agreement as the way set, and thus constitute a transparent Bluetooth serial channel to make the data transfer between the master and the slave Bluetooth devices. When the module works in the command response mode, it can perform all of the following AT commands. So the user can send various AT commands to the module to set parameters or control behaviors. By controlling the input voltage level of the external pin (PIO11) of the module, we can change the work mode of it dynamically.

The definition of the serial port pins of the module:

1. *PIO8 is in connection with LED, which indicates the module's working state. Flashing interval differs in different states.*
2. *PIO9 is in connection with LED, which indicates connecting successfully. LED will be on all the time if connecting successfully with other bluetooth device.*
3. *PIO11, mode switching, high -> AT command response state, low or floating -> Bluetooth regular working state.*
4. *There is already a reset circuit on the board, repower it can reset it.*

The steps to set it as the master device

1. *Set PIO11 high.*
2. *Power it on and the module works in AT Command State.*
3. *Use a serial communication tool to set the parameters of the serial communication as baud rate: 38400, databit:8, stopbit:1, none calibration and none flow control.*

4. *Send characters "AT + ROLE = 1 \ r \ n" via serial port and it will return "OK \ r \ n", where \ r \ n as carriage return line feed character.*

5. *Set PIO11 low, repower it to make it work as the master. It will search the slave module and make the connection automatically.*

AT Commands in Details

AT commands are not case sensitive, and end with carriage return, new line character: \ r \ n.

1. Test command:

Instruction	Response	Parameter
AT	OK	None

2. Module reset (restart):

Instruction	Response	Parameter
AT+RESET	OK	None

Instruction execution results: reset the module (equivalent to repower)

3. To obtain the software version number:

Instruction	Response	Parameter
AT + VERSION?	+ VERSION: <Param> OK	Param: software version number

For example:

```
at + version? \ r \ n

+ VERSION :1.0-20090818

OK
```

4. To restore the default configuration:

Instruction	Response	Parameter
AT + ORGL	OK	none

Factory default state:

①. Equipment number: 0

②. Query Code: 0x009e8b33

③. Module Role: Slave Mode

④. Connection mode: connecting with specified Bluetooth devices

⑤. Serial port parameters: baud rate-38400bits / s; stop bits: 1; parity bit: none

⑥. Pairing code: "1234"

⑦. Device Name: "HHW-SPP-1800-2"

5. To obtain the Bluetooth address of the module:

Instruction	***Response***	***Parameter***
AT + ADDR?	*+ ADDR: <Param>* *OK*	*Param: Module Bluetooth address*

Bluetooth address of Representation: NAP: UAP: LAP (hex)

For example:

Module Bluetooth device address is: 12:34:56: ab: cd: ef

at + addr? \ r \ n

+ ADDR: 1234:56: abcdef

OK

6. Set / query the device name:

Instruction	*Response*	*Parameter*
AT + NAME = <Param>	*OK*	*Param: Bluetooth device name* *Default name: "HHW-SPP-1800-2"*
AT + NAME?	*+ NAME: <Param> OK-successful* *FAIL - Failed*	

For example:

AT + NAME = HHW-SPP-1800-2 \ r \ n ----- set the device name: "HHW-SPP-1800-2"

OK

AT + NAME = "HHW-SPP-1800-2" \ r \ n ----- set the device name: "HHW-SPP-1800-2"

OK

at + name = Beijin \ r \ n ----- set the device name: "Beijin"

OK

at + name = "Beijin" \ r \ n ----- set the device name: "Beijin"

OK

at + name? \ r \ n

+ NAME: Beijin

OK

7. Access to the remote Bluetooth device name:

Instruction	*Response*	*Parameter*
AT + RNAME? <Param1>	*+ RNAME: <Param2>* *OK- Successful* *FAIL - Failed*	*Param1: the remote Bluetooth device address* *Param2: the remote Bluetooth device name*

Bluetooth address of Representation: NAP: UAP: LAP (hex)

For example:

Remote device Bluetooth address: 00:02:72:0 d: 22:24, device name: Bluetooth

at + rname? 0002,72,0 d2224 \ r \ n

+ RNAME: Bluetooth

OK

8. Set / query - Module Role:

Instruction	*Response*	*Parameter*
AT + ROLE = <Param>	*OK*	*Param: parameter values are as follows:* *0 - Slave* *1 - Master* *2 - Slave-Loop* *Default value: 0*
AT + ROLE?	*+ ROLE: <Param>* *OK*	

Module Role Description:

Slave --- passive connection;

Slave-Loop ---
passive connection, receive data from master Bluetooth device and them send the data back to master Bluetooth device;

Master --- detecting SPP slave Bluetooth devices, and initiate connections.

9. Set / Query - Equipment:

Instruction	Response	Parameter
AT + CLASS = <Param>	OK	Param: Equipment Bluetooth device actually has a 32-bit parameter, the parameter indicates device type, and supported services . Default value: 0 Specific settings, see Annex 1: Equipment Description
AT + CLASS?	+ CLASS: <Param> OK – Successful FAIL - Failed	

In order to effectively filter surrounding Bluetooth devices and quickly search or query specific Bluetooth device, the module can be set as non-standard module, such as: 0x1f1f (hex).

10. Set / query - query access code:

Instruction	Response	Parameter
AT +IAC = <Param>	OK – success FAIL - Failed	Param: query access code Default value: 9e8b33 Specific settings, see Annex 2: query access code Description
AT + IAC?	+ IAC: <Param> OK	

Access code is set to GIAC (General Inquire Access Code: 0x9e8b33) General Query access code, which can be used to find all the Bluetooth devices around. In order to effectively locate or be located quickly in a number of Bluetooth devices around, Access the module query can be set as numbers other than GIAC and LIAC code, such as: 9e8b3f.

For example:

AT + IAC = 9e8b3f \ r \ n

OK

AT + IAC? \ R \ n

+ IAC: 9e8b3f

OK

11. Set / query - query access patterns:

Instruction	Response	Parameter

AT + INQM = <Param1>, <Param2>, <Param3>	*OK – success* *FAIL - Failed*	*Param1: query pattern* *0 - inquiry_mode_standard* *1 - inquiry_mode_rssi* *Param2: maximum number of Bluetooth devices to respond* *Param3: maximum query overtime* *Time-out range: 1 ~ 48* *(Converted into time: 1.28 seconds to 61.44 seconds)* *Default value: 1,1,48*
AT + INQM?	*+ INQM: <Param1>, <Param2>, <Param3>* *OK*	

For example:

AT + INQM = 1,9,48 \ r \ n--

Query mode settings: with RSSI signal strength indicator, If more than 9 Bluetooth devices respond, stop inquiry, set overtime to 48x1.28 = 61.44s.

OK

AT + INQM? \ R \ n

+ INQM: 1,9,48

OK

12. Set / Query - matching code:

Instruction	*Response*	*Parameter*
AT + PSWD = <Param1>	*OK*	*Param:matching code* *Default:"1234"*
AT + PSWD?	*+ PSWD: <Param>* *OK*	

13. Set / Query - serial port parameters:

Instruction	*Response*	*Parameter*
AT + UART = <Param1>,<Param2>,<Param3>	*OK*	*Param1:baudrate(bits/s)* *4800,9600,19200,38400,* *57600,115200,230400,* *460800,921600,1382400* *Param2:stopbit* *bit ,1-2 bits* *Param3:check bit* *0-None,1-Odd,2-Even* *Default:9600,0,0*
AT + UART?	*+ UART: <Param1>, <Param2>, <Param3>* *OK*	

For example: set serial port baud rate: 115200,2 stop bit, Even check

```
AT + UART = 115200,1,2 \ r \ n

OK

AT + UART?

+ UART: 115200,1,2

OK
```

14. Set / Query - connection mode:

Instruction	Response	Parameter
AT + CMODE= <Param>	OK	Param: 0 - the specified Bluetooth address connection mode (Bluetooth address specified by the bound instruction) 1 - any Bluetooth address connection mode (Not constrained by bound addresses) Default connection mode: 0
AT + CMODE?	+ CMODE: <Param> OK	

15. Set / Query - binding Bluetooth address:

Instruction	Response	Parameter
AT + BIND= <Param>	OK	Param: the Bluetooth address bounded Default: 00:00:00:00:00:00
AT + BIND?	+ BIND: <Param> OK	

Bluetooth address: NAP: UAP: LAP (hex)

Binding instruction works only in connection mode with the specified Bluetooth address!

For example:

Bluetooth address of the specified connection mode, the binding Bluetooth device address: 12:34:56: ab: cd: ef

Command and response are as follows:

```
AT + BIND = 1234,56, abcdef \ r \ n

OK

AT + BIND? \ R \ n
```

```
+ BIND: 1234:56: abcdef

OK
```

16. Set / Query - LED drive and the output polarity:

Instruction	Response	Parameter
AT + POLAR= <Param1>,<Param2>	OK	Param1: 0-LED is on when PIO8 is low 1-LED is on when PIO8 is high Param2: 0-PIO9 is low to indicate successful connection. 1- PIO9 is high to indicate successful connection.
AT + POLAR?	+ POLAR: <Param1>,<Param2> OK	

HHW-SPP-1800-2, HHW-SPP-100-2, HHW-SPP-10-2 Bluetooth module definition: PIO8 output drives LED to indicate working state; PIO9 output indicates connection state.

For example:

PIO8 output low means light LED, PIO9 output high means a successful connection.

Command and response are as follows:

```
AT + POLAR = 0, 1 \ r \ n

OK

AT + POLAR? \ R \ n

+ POLAR: 0, 1

OK
```

17. Set PIO single-port output:

Instruction	Response	Parameter
AT + PIO <Param1>,<Param2>	OK	Param1: PIO port number (decimal number) Param2: PIO port output state 0-low,1-high

HHW-SPP-1800-2 or HHW-SPP-100-2 Bluetooth module provides PIO port resources: PIO2 ~ PIO7 and PIO10;

HHW-SPP-10-2 Bluetooth module provides PIO port resources: PIO0 ~ PIO7 and PIO10, which can be used to extend input and output ports.

For example:

1, Set PIO10 port to output high

AT + PIO = 10, 1 \ r \ n

OK

2, Set PIO10 port to output low

AT + PIO = 10, 0 \ r \ n

OK

18. Set PIO multi-port output:

Instruction	*Response*	*Parameter*
AT +MPIO =<Param>	*OK*	*Param: PIO port number mask combinations (hexadecimal number)*

HHW-SPP-1800-2 or HHW-SPP-100-2 Bluetooth module provides PIO port resources: PIO2 ~ PIO7 and PIO10;

HHW-SPP-10-2 Bluetooth module provides PIO port resources: PIO0 ~ PIO7 and PIO10, which can be used to extend input and output ports.

PIO port number mask = (1 <<port number)

PIO port number mask combinations = (PIO port number mask 1 | PIO port number mask 2 | ... ...)

Such as:

PIO2 port mask = (1 <<2) = 0x004

PIO10 port mask = (1 <<10) = 0x400

PIO2 and PIO10 port mask combinations = (0x004 | 0x400) = 0x404

For example:

1. PIO10 and PIO2 port output high

AT + MPIO = 404 \ r \ n

OK

2. PIO4 port output high

AT + PIO = 004 \ r \ n

OK

3. PIO10 port output high

AT + PIO = 400 \ r \ n

OK

4. All the port output low

AT + MPIO = 0 \ r \ n

OK

19. Query PIO port type:

Instruction	Response	Parameter
AT +MPIO?	+ MPIO: <Param> OK	Param -- PIO port value (16bits) Param [0] = PIO0 Param [1] = PIO1 Param [2] = PIO2 ... ... Param [10] = PIO10 Param [11] = PIO11

HHW-SPP-1800-2 and HHW-SPP-100-2 Bluetooth module provide the users with PIO resources: PIO2~PIO7 and PIO10~PIO11;

HHW-SPP-10-2 provides the users with PIO0~PIO7 and PIO10~PIO11, users can use them for input or output extension.

20. Set / Query - paging scan, inquiry scan parameters:

Instruction	Response	Parameter
AT + IPSCAN= <Param1>,<Param2>, <Param3>,<Param4>	OK	Param1: query time interval Param2: query duration Param1: paging time interval Param2: paging duration All in decimal Default:1024,512,1024,512
AT + IPSCAN?	+ IPSCAN: <Param1>,<Param2>, <Param3>,<Param4> OK	

For example:

at + ipscan = 1234,500,1200,250 \ r \ n

OK

at + ipscan?

+ IPSCAN: 1234,500,1200,250

OK

21. Set / Query - SNIFF energy parameters:

Instruction	Response	Parameter
AT + SNIFF= <Param1>,<Param2>,<Param3>,<Param4>	OK	Param1: maximum time Param2: minimum duration Param1: trying time Param2: overtime time All in decimal Default:0,0,0,0
AT + SNIFF?	+ SNIFF: <Param1>,<Param2>,<Param3>,<Param4> OK	

22. Set / check the security, encryption mode:

Instruction	Response	Parameter
AT + SENM= <Param1>,<Param2>,	OK –Successful FAIL-Failure	Param1: safe mode, values are as follows: 0 - sec_mode0_off 1 - sec_mode1_non_secure 2 - sec_mode2_service 3 - sec_mode3_link 4 - sec_mode_unknown Param2: encryption mode, values are as follows: 0 - hci_enc_mode_off 1 - hci_enc_mode_pt_to_pt 2 -hci_enc_mode_pt_to_pt_and_bcast Default value: 0,0
AT + SENM?	+ SENM: <Param1>,<Param2>, OK	

23. Remove the specified Authenticated Device from the paired list:

Instruction	Response	Parameter
AT +RMSAD =<Param>	OK	Param: Bluetooth device address

For example:

Remove the Bluetooth address from paired list: 12:34:56: ab: cd: ef equipment

at + rmsad = 1234,56, abcdef \ r \ n

OK - deleted successfully

Or

at + rmsad = 1234,56, abcdef \ r \ n

FAIL - address does not exist 12:34:56: ab: cd: ef

24. Remove all Authenticated Device from the Bluetooth pairing list:

Instruction	*Response*	*Parameter*
AT +RMAAD =<Param>	*OK*	*None*

For example:

Remove all the paired Bluetooth devices addresses

at + rmaad \ r \ n

OK

25. Find the specified Authenticated Device from the Bluetooth pairing list:

Instruction	*Response*	*Parameter*
AT +FSAD =<Param>	*OK – exist* *FAIL – not exist*	*Param:Bluetooth device address*

For example:

Find the Bluetooth devices from paired list: 12:34:56: ab: cd: ef

at + fsad = 1234,56, abcdef \ r \ n

OK - address12:34:56: ab: cd: ef exists in the list.

at + fsad = 1234,56, abcde0 \ r \ n

FAIL - address does not exist

26. Get Authenticated Device Count in the Bluetooth pairing list:

Instruction	*Response*	*Parameter*
AT + ADCN ?	*+ ADCN: <Param>* *OK*	*Param: Number of matching the list of Bluetooth devices*

For example:

at + adcn?

+ ADCN: 0 - did not match the list of Bluetooth devices Trust

OK

27. Get Most Recently Used Authenticated Device:

Instruction	*Response*	*Parameter*
AT + MRAD ?	*+ MRAD: <Param>* *OK*	*Param: recently used Bluetooth device address*

For example:

at + mrad?

+ MRAD: 0:0:0 - not recently used Bluetooth device trust

OK

28. Get the Bluetooth module state:

Instruction	*Response*	*Parameter*
AT + STATE ?	*+ STATE: <Param>* *OK*	*Param: Module state* *Return values are as follows:* *"INITIALIZED"* *"READY"* *"PAIRABLE"* *"PAIRED"* *"INQUIRING"* *"CONNECTING"* *"CONNECTED"* *"DISCONNECTED"* *"NUKNOW"*

For example:

at + state?

+ STATE: INITIALIZED

OK

29. Initialize the SPP profile lib:

Instruction	*Response*	*Parameter*
AT + INIT	*OK-Successful* *FAIL-Failure*	*None*

30. Check Bluetooth device

Instruction	*Response*	*Parameter*

AT + INQ	*+ INQ: <Param1>, <Param2>, <Param3>……* *OK*	*Param1: Bluetooth Address* *Param2: Equipment* *Param3: RSSI signal strength*

Example 1:

at + init \ r \ n

- Initialize the SPP library (can not repeat the initialization)

OK

at + iac = 9e8b33 \ r \ n - check Bluetooth device with any access code

OK

at + class = 0 \ r \ n

- Access various class of Bluetooth device

OK

at + inqm = 1,9,48 \ r \ n - Query mode: strength indicator with RSSI signal, stop the query if more than nine Bluetooth devices respond, set overtime 48x1.28 = 61.44s.

At + inq \ r \ n

- Check the surrounding Bluetooth devices

+ INQ: 2:72: D2224, 3E0104, FFBC

+ INQ: 1234:56:0,1 F1F, FFC1

+ INQ: 1234:56:0,1 F1F, FFC0

+ INQ: 1234:56:0,1 F1F, FFC1

+ INQ: 2:72: D2224, 3E0104, FFAD

+ INQ: 1234:56:0,1 F1F, FFBE

+ INQ: 1234:56:0,1 F1F, FFC2

+ INQ: 1234:56:0,1 F1F, FFBE

+ INQ: 2:72: D2224, 3E0104, FFBC

OK

Example 2:

at + iac = 9e8b33 \ r \ n - check Bluetooth device with any access code

OK

at + class = 1f1f \ r \ n - check the device's Bluetooth device of class 0x1f1f

OK

at + inqm = 1,9,48 \ r \ n - Query mode:strength indicator with RSSI signal, stop the query if more than nine Bluetooth devices respond, set overtime 48x1.28 = 61.44s.

At + inq \ r \ n

- Filter, check the surrounding Bluetooth devices

+ INQ: 1234:56:0,1 F1F, FFC2

+ INQ: 1234:56:0,1 F1F, FFC1

+ INQ: 1234:56:0,1 F1F, FFC1

+ INQ: 1234:56:0,1 F1F, FFC1

+ INQ: 1234:56:0,1 F1F, FFC2

+ INQ: 1234:56:0,1 F1F, FFC1

+ INQ: 1234:56:0,1 F1F, FFC1

+ INQ: 1234:56:0,1 F1F, FFC0

+ INQ: 1234:56:0,1 F1F, FFC2

OK

Example 3:

at + iac = 9e8b3f \ r \ n - check the Bluetooth device with access code 0x9e8b3f

OK

at + class = 1f1f \ r \ n - check the device's Bluetooth device of class 0x1f1f

OK

at + inqm = 1,1,20 \ r \ n - Query mode:strength indicator with RSSI signal, stop the query if more than nine Bluetooth devices respond, set overtime 48x1.28 = 61.44s.

At + inq \ r \ n

- Filter, check the surrounding Bluetooth devices

+ INQ: 1234:56: ABCDEF, 1F1F, FFC2

OK

31. Cancel checking Bluetooth device:

Instruction	*Response*	*Parameter*
AT + INQC	*OK*	*None*

32. Device pairing:

Instruction	*Response*	*Parameter*
AT + PAIR = <Param1>, <Param2>	*OK – success* *FAIL - Failed*	*Param1: remote device Bluetooth Address* *Param2: Connection overtime (second)*

For example:

Pair with the remote Bluetooth device: 12:34:56: ab: cd: ef , with the biggest pairing overtime 20 seconds.

At + pair = 1234,56, abcdef, 20 \ r \ n

OK

33. Device connection:

Instruction	*Response*	*Parameter*
AT + LINK = <Param>	*OK – success* *FAIL - Failed*	*Param: remote device Bluetooth Address*

For example:

Initialise connection with the remote Bluetooth device: 12:34:56: ab: cd: ef

at + fsad = 1234,56, abcdef \ r \ n - check if Bluetooth device 12:34:56: ab: cd: ef is tin he matching list

OK

at + link = 1234,56, abcdef \ r \ n - Bluetooth device 12:34:56: ab: cd: ef is in the match list and without connection can be initialised without query

OK

34. Disconnection:

Instruction	*Response*	*Parameter*
AT + DISC	*+ DISC: SUCCESS - Disconnect success* *OK* *+ DISC: LINK_LOSS - connection lost* *OK* *+ DISC: NO_SLC - no SLC connection* *OK* *+ DISC: overtime - overtime disconnect* *OK* *+ DISC: ERROR - disconnect error* *OK*	*None*

35. Enter energy-saving mode:

Instruction	*Response*	*Parameter*
AT + ENSNIFF = <Param>	*OK*	*Param: Bluetooth device address*

36. Exit energy-saving mode

Instruction	*Response*	*Parameter*
AT + EXSNIFF = <Param>	*OK*	*Param: Bluetooth device address*

Appendix 1: AT Error Codes Reply

AT command error codes response ----ERROR: (error code)

Error code (hexadecimal)	*Note*
0	*AT command error*
1	*Instruction response is default*
2	*PSKEY write error*
3	*Device name is too long (more than 32 bytes)*
4	*Device name length of zero*
5	*Bluetooth Address: NAP is too long*
6	*Bluetooth Address: UAP is too long*
7	*Bluetooth Address: LAP is too long*
8	*PIO serial mask length is zero*
9	*Invalid PIO serial number*
A	*Device class length is 0*

B	*Device class number is too long*
B	*Query access code length is zero*
D	*Query access code length is too long*
E	*Invalid query access code*
F	*Pairing code length is zero*
10	*Pairing code is too long (more than 16 bytes)*
11	*Invalid module role*
12	*Invalid baud rate*
13	*Invalid stop bit*
14	*Invalid parity bit*
15	*Pair list does not contain the certified equipment*
16	*SPP library not initialized*
17	*SPP library repeated initialization*
18	*Invalid query state*
19	*Checking overtime is too long*
1A	*Bluetooth address is zero*
1B	*Invalid security mode*
1C	*Invalid encryption mode*

Appendix 2: Equipment Description

The Class of Device / Service (CoD) is a 32 bits number that is made of 3 fields. One field specifies the service supported by the device. Another field specifies the major device class, which broadly corresponds to the type of the device. The third field specifies the minor device class, which describes the device type in more detail.

The Class of Device / Service (CoD) field has a variable format. The format is indicated using the 'Format Type field' within the CoD. The length of the Format Type field is variable and ends with two bits different from '11 '. The version field starts at the least significant bit of the CoD and may extend upwards. In the 'format # 1' of the CoD (Format Type field = 00), 11 bits are assigned as a bit-mask (multiple bits can be set) each bit corresponding to a high level generic category of service class. Currently 7 categories are defined. These are primarily of a 'public service' nature. The remaining 11 bits are used to indicate device type category and other device-specific characteristics. Any reserved but otherwise unassigned bits, such as in the Major Service Class field, should be set to 0.

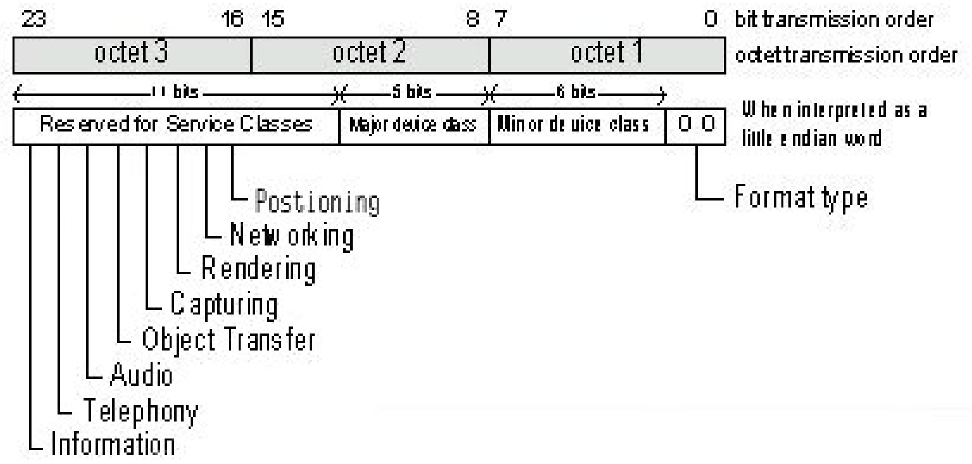

Figure 1.2: The Class of Device / Service field (first format type). Please note the order in which the octets are sent on the air and stored in memory. Bit number 0 is sent first on the air.

1. MAJOR SERVICE CLASSES

Bit no	*Major Service Class*
13	*Limited Discoverable Mode [Ref # 1]*
14	*(Reserved)*
15	*(Reserved)*
16	*Positioning (Location identification)*
17	*Networking (LAN, Ad hoc...)*
18	*Rendering (Printing, Speaker...)*
19	*Capturing (Scanner, Microphone ...)*
20	*Object Transfer (v-Inbox, v-Folder...)*
21	*Audio (Speaker, Microphone, Headset service ...)*
22	*Telephony (Cordless telephony, Modem, Headset service ...)*
23	*Information (WEB-server, WAP-server ...)*

TABLE 1.2: MAJOR SERVICE CLASSES

[Ref # 1 As defined in See Generic Access Profile, Bluetooth SIG]

2. MAJOR DEVICE CLASSES

The Major Class segment is the highest level of granularity for defining a Bluetooth Device.The main function of a device is used to determine the major class grouping. There are 32 different possible major classes. The assignment of this Major Class field is defined in Table 1.3.

12~8	*Major Device Class*
0 0 0 0 0	*Miscellaneous [Ref # 2]*
0 0 0 0 1	*Computer (desktop, notebook, PDA, organizers ....)*
0 0 0 1 0	*Phone (cellular, cordless, payphone, modem ...)*

0 0 0 1 1	*LAN / Network Access point*
0 0 1 0 0	*0 0 1 0 0 Audio / Video (headset, speaker, stereo, video display, vcr .....*
0 0 1 0 1	*Peripheral (mouse, joystick, keyboards .....)*
0 0 1 1 0	*Imaging (printing, scanner, camera, display ...)*
1 1 1 1 1	*Uncategorized, specific device code not specified*
XXXXX	*All other values reserved*

TABLE 1.3: MAJOR DEVICE CLASSES

[Ref # 2: Used where a more specific Major Device Class code is not suited (but only as specified in this document). Devices that do not have a major class code assigned can use the all-1 code until 'classified']

3. THE MINOR DEVICE CLASS FIELD

The 'Minor Device Class field' (bits 7 to 2 in the CoD), are to be interpreted only in the context of the Major Device Class (but independent of the Service Class field). Thus the meaning of the bits may change, depending on the value of the 'Major Device Class field'. When the Minor Device Class field indicates a device class, then the primary device class should be reported, eg a cellular phone that can also work as a cordless handset should use 'Cellular' in the minor device class field.

4. MINOR DEVICE CLASS FIELD - COMPUTER MAJOR CLASS

7~2	*Minor Device Class* *bit no of CoD*
0 0 0 0 0 0	*Uncategorized, code for device not assigned*
0 0 0 0 0 1	*Desktop workstation*
0 0 0 0 1 0	*Server-class computer*
0 0 0 0 1 1	*Laptop*
0 0 0 1 0 0	*Handheld PC / PDA (clam shell)*
0 0 0 1 0 1	*Palm sized PC / PDA*
0 0 0 1 1 0	*Wearable computer (Watch sized)*
XXXXXX	*All other values reserved*

TABLE 1.4: SUB DEVICE CLASS FIELD FOR THE 'COMPUTER' MAJOR CLASS

5. MINOR DEVICE CLASS FIELD - PHONE MAJOR CLASS

7~2	*Minor Device Class* *bit no of CoD*
0 0 0 0 0 0	*Uncategorized, code for device not assigned*
0 0 0 0 0 1	*Cellular*

0 0 0 0 1 0	Cordless
0 0 0 0 1 1	Smart phone
0 0 0 1 0 0	Wired modem or voice gateway
0 0 0 1 0 1	Common ISDN Access
0 0 0 1 1 0	Sim Card Reader
XXXXXX	All other values reserved

TABLE 1.5: SUB DEVICE CLASSES FOR THE 'PHONE' MAJOR CLASS

6. MINOR DEVICE CLASS FIELD - LAN / NETWORK ACCESS POINT MAJOR CLASS

7~5	Minor Device Class bit no of CoD
0 0 0	Fully available
0 0 1	1 - 17% utilized
0 1 0	17 - 33% utilized
0 1 1	33 - 50% utilized
1 0 0	50 - 67% utilized
1 0 1	67 - 83% utilized
1 1 0	83 - 99% utilized
1 1 1	No service available [REF # 3]
XXX	All other values reserved

TABLE 1.6: THE LAN / NETWORK ACCESS POINT LOAD FACTOR FIELD

[Ref # 3: "Device is fully utilized and cannot accept additional connections at this time, please retry later "]

The exact loading formula is not standardized. It is up to each LAN / Network Access Point implementation to determine what internal conditions to report as a utilization percentage. The only requirement is that the number reflects an ever-increasing utilization of communication resources within the box. As a recommendation, a client that locates multiple LAN / Network Access Points should attempt to connect to the one reporting the lowest load.

4~2	Minor Device Class bit no of CoD
0 0 0	Uncategorized (use this value if no other apply)
XXX	All other values reserved

TABLE 1.7: RESERVED SUB-FIELD FOR THE LAN / NETWORK ACCESS POINT

7. MINOR DEVICE CLASS FIELD - AUDIO / VIDEO MAJOR CLASS

7~2	Minor Device Class bit no of CoD
0 0 0 0 0 0	Uncategorized, code not assigned

0 0 0 0 0 1	*Device conforms to the Headset profile*
0 0 0 0 1 0	*Hands-free*
0 0 0 0 1 1	*(Reserved)*
0 0 0 1 0 0	*Microphone*
0 0 0 1 0 1	*Loudspeaker*
0 0 0 1 1 0	*Headphones*
0 0 0 1 1 1	*Portable Audio*
0 0 1 0 0 0	*Car audio*
0 0 1 0 0 1	*Set-top box*
0 0 1 0 1 0	*HiFi Audio Device*
0 0 1 0 1 1	*VCR*
0 0 1 1 0 0	*Video Camera*
0 0 1 1 0 1	*Camcorder*
0 0 1 1 1 0	*Video Monitor*
0 0 1 1 1 1	*Video Display and Loudspeaker*
0 1 0 0 0 0	*Video Conferencing*
0 1 0 0 0 1	*(Reserved)*
0 1 0 0 1 0	*Gaming / Toy [Ref # 4]*
XXXXXX	*All other values reserved*

[Ref # 4: Only to be used with a Gaming / Toy device that makes audio / video capabilities available via Bluetooth]

TABLE 1.8: SUB DEVICE CLASSES FOR THE 'AUDIO / VIDEO' MAJOR CLASS

8. MINOR DEVICE CLASS FIELD - PERIPHERAL MAJOR CLASS

7~6	*Minor Device Class* *bit no of CoD*
0 1	*Keyboard*
1 0	*Pointing device*
1 1	*Combo keyboard / pointing device*
XX	*All other values reserved*

TABLE 1.9: THE PERIPHERAL MAJOR CLASS KEYBOARD / POINTING DEVICE FIELD

Bits 6 and 7 independently specify mouse, keyboard or combo mouse / keyboard devices. These may be combined with the lower bits in a multifunctional device.

5~2	*Minor Device Class* *bit no of CoD*
0 0 0 0	*Uncategorized device*
0 0 0 1	*Joystick*
0 0 1 0	*Gamepad*
0 0 1 1	*Remote control*
0 1 0 0	*Sensing device*
0 1 0 1	*Digitizer tablet*
XXXX	*All other values reserved*

TABLE 1.10: RESERVED SUB-FIELD FOR THE DEVICE TYPE

9. MINOR DEVICE CLASS FIELD - IMAGING MAJOR CLASS

7~4	Minor Device Class bit no of CoD
XXX 1	Display
XX 1 X	Camera
X 1 XX	Scanner
1 XXX	Printer
XXXX	All other values reserved

TABLE 1.11: THE IMAGING MAJOR CLASS BITS 4 TO 7

Bits 4 to 7 independently specify display, camera, scanner or printer. These may be combined in a multifunctional device.

3~2	Minor Device Class bit no of CoD
0 0	Uncategorized, default
XX	All other values reserved

TABLE 1.12: THE IMAGING MAJOR CLASS BITS 2 AND 3

Bits 2 and 3 are reserved

Appendix 3: The Inquiry Access Codes

The General-and Device-Specific Inquiry Access Codes (DIACs)

The Inquiry Access Code is the first level of filtering when finding Bluetooth devices and services. The main purpose of defining multiple IACs is to limit the number of responses that are received when scanning devices within range.

0. 0x9E8B33 - General / Unlimited Inquiry Access Code (GIAC)

1. 0x9E8B00 - Limited Dedicated Inquiry Access Code (LIAC)

2. 0x9E8B01 ~ 0x9E8B32 RESERVED FOR FUTURE USE

3. 0x9E8B34 ~ 0x9E8B3F RESERVED FOR FUTURE USE

The Limited Inquiry Access Code (LIAC) is only intended to be used for limited time periods in scenarios where both sides have been explicitly caused to enter this state, usually by user action. For further explanation of the use of the LIAC, please refer to the Generic Access Profile.

In contrast it is allowed to be continuously scanning for the General Inquiry Access Code (GIAC) and respond whenever inquired.