SIMCom

A company of SIM Tech

SIM800 Series _Bluetooth_ Application Note_V1.07

Document Title	SIM800 Series_Bluetooth_Application Note
Version	1.07
Date	2017-10-31
Status	Release
Document Control ID	SIM800 Series_Bluetooth_Application Note_V1.07

General Notes

Simcom offers this information as a service to its customers, to support application and engineering efforts that use the products designed by Simcom. The information provided is based upon requirements specifically provided to Simcom by the customers. Simcom has not undertaken any independent search for additional relevant information, including any information that may be in the customer's possession. Furthermore, system validation of this product designed by SIMCOM within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

Copyright

Content

Version History

Date	Version	Description	Author
2013-11-07	1.00	Original	Ping Zhang
2014-03-26	1.01	Chapter 1.4, Add "power-saving mode" description Chapter 2.6, AT+BTSCAN add <rssi> parameter Chapter 2.13, Modify AT+BTSPPGET parameter Chapter 2.14, Modify AT+BTSPPSEND parameter Chapter 2.22, Add AT+BTVTS command Chapter 2.23, Add AT+BTCIND command Chapter 2.24, Add AT+BTCLCC command Chapter 2.25, Add AT+BTPBSYNC command Chapter 2.26, Add AT+BTPBF command Chapter 2.27, Add AT+BTAVRCOP command Chapter 2.28, Add AT+BTVIS command Chapter 2.29, Add AT+BTSPPCFG command Chapter 2.30, Add AT+BTPAIRCFG command Chapter 3, Add Error Code 1051,1056--1058,1060 Chapter 4, Add 4.7----4.17	Ping Zhang
2014-06-30	1.02	Chapter 2.13, Modify AT+BTSPPGET and <command> description Chapter 2.31,Add AT+CPBFEX command Chapter 2.32,Add AT+BTRING command Chapter 4.12,Modify demo Chapter 5, Add	Ping Zhang
2015-01-12	1.03	Chapter 2.14, Modify AT+BTSPPSEND usage Chapter 2.25, Modify description of <fail_num> Chapter 2.29, Modify AT+BTSPPCFG command Chapter 2.3, Modify AT+BTPOWER command note Chapter 2.31,Modify AT+CPBFEX command Chapter 2.32,Modify AT+BTRING command Chapter 2.33,Add AT+BTACI command Chapter 2.34,Add AT+BTHFGOP command Chapter 2.35, Add AT+BTSPPURC command Chapter 5.2,Modify CPBF command difference	Chen Yan ZhuDingFen
2015-2-9	1.04	Add SIM800C Chapter 2.36,Add AT+BTCLCCS command	Ping Zhang
2015-8-6	1.05	Add SIM800A,SIM800F Chapter 1.5, Add	Chen Yan

		Chapter 1.6, Add Chapter 2.37,Add AT+BTSPPCFD command Chapter 2.38,Add AT+BTCOD command	
2017-09-28	1.06	Scope Chapter 2.40,Add AT+BLESREG Chapter 2.41, Add AT+BLESDREG Chapter 2.42, Add AT+BLESSAD Chapter 2.43, Add AT+BLESSRM Chapter 2.44, Add AT+BLESSC Chapter 2.45, Add AT+BLESSD Chapter 2.46, Add AT+BLESSSTART Chapter 2.47, Add AT+BLESSSTOP Chapter 2.48, Add AT+BLESLSTART Chapter 2.49, Add AT+BLESLSTOP Chapter 2.50, Add AT+BLEADV Chapter 2.51, Add AT+BLECPU Chapter 2.52, Add AT+BLESIND Chapter 2.53, Add AT+BLESRSP Chapter 2.54, Notify when connection's status change Chapter 2.55, Add AT+BLEFMP Chapter 2.56, Notify when connection's status change comes +BLEFMPCON Chapter 2.57, Notify when a client's write request comes +BLEFMPWREQ Chapter 2.58, Add AT+BLEPXP Chapter 2.59, Notify when connection's status change comes +BLEPXPCON Chapter 2.60, Notify when a Link loss alert comes +BLEPXPLLAT Chapter 2.61, Notify when a a disconnection alert comes +BLEPXPDISAT	Wenjie.lai
2017-10-31	1.07	Chapter 1.6, Add SIM868E Appendix, add BLE profiles	Wenjie.lai

Scope

This document describes how to use the AT command about Bluetooth and some application note.The document can apply to all SIM800 series modules with Bluetooth fuction.

1 Bluetooth Function

1.1 Bluetooth Introduction

Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength radio transmissions in the ISM band from 2400–2480 MHz) from fixed and mobile devices, creating prsonal area networks (PANs) with high levels of security.Bluetooth was standardized as IEEE 802.15.1.
The bluetooth version is BT3.0(all projects) and BLE(only SIM868E).

1.2 Bluetooth Profile

To use Bluetooth wireless technology, a device has to be able to interpret certain Bluetooth profiles, which are definitions of possible applications and specify general behaviors that Bluetooth enabled devices use to communicate with other Bluetooth devices. These profiles include settings to parametrize and to control the communication from start. Adherence to profiles saves the time for transmitting the parameters anew before the bi-directional link becomes effective. There are a wide range of Bluetooth profiles that describe many different types of applications or use cases for devices.

1.3 Bluetooth Device Address

The Bluetooth device address stores the network address of a Bluetooth–enabled device. It is used to identify a particular device during operations such as connecting to, pairing with, or activating the device.

A Bluetooth–enabled device address is a unique, 48 bits address containing the following three fields:

- LAP field: lower part of the address containing 24 bits.
- UAP field: upper part of the address containing 8 bits.
- NAP field: non–significant part of the address containing 16 bits.

The LAP and the UAP represent the significant address part (SAP) of the Bluetooth device address.

1.4 AT Interface for Bluetooth Function

As module solution, we provide series of AT interface to operate Bluetooth function, including pairing, bonding, pushing or receiving file.

Also including interface for SPP service which could communicate between Bluetooth device and others via serial port.

When the module as a Bluetooth headset role, we provide a set of AT commands to control the remote smart phones, such as phone calls, turn on or hang up calls and so on.

By default, the module operates in power-saving mode, which means that the module can be simultaneously connected to a Bluetooth device. When the module to establish a connection with a device, other devices can not be scanned into the module, the module can not get profile, will not be able to establish new connections and modules. If the customer's application scenario, the module needs to be multiple Bluetooth devices (currently up to three) connection, you need to use the AT+BTSPPCFG=1 command to turn off the power saving mode. It should be noted that the power saving mode does not affect the module initiative to connect to other Bluetooth devices.

1.5 Multi Device Connection

For the MTK6260 platform module, by default, the module works in power saving mode, which means that the module can only be connected to a Bluetooth device. When the module is connected with a certain device, other devices can not scan to the module, but also unable to obtain the module's Profile and can not establish a new connection with the module. If the customer's application scenario, the need for the module is connected to a number of Bluetooth devices (currently up to three), then you need to use the AT+BTSPPCFG=1 command to shut down the power saving mode. Note that the power saving mode does not affect the module's initiative to connect to other Bluetooth devices.

1.6 Function Differences

The current Bluetooth module series can be divided into four platforms, these two platforms to support the Bluetooth function will be different, divided as follows:
MTK6260 platforms: SIM800, SIM800M64, SIM800H.
MTK6261 platforms: SIM808, SIM800C, SIM800A, SIM800F.
MTK6261_DS platforms: SIM800C-DS.
MTK2503 platforms：SIM868, SIM868E.

- support Profile

All of the SIM800 series module have four basic profiles, they are OPP, HSP/HFP, SPP.
For the MTK6260 platform module, support A2DP, AVRCP, PBAP all the roles.
For the MTK6261 and MTK2503 platform module, support PBAP all the roles and only supports A2DP, AVRCP mobile role.
For the MTK2503 platform module SIM868E, additionally support BLEFMP, BLEPXP, BLESPP, Customer can also define their own GATT server.

- Multi-device connection

For the MTK6260 and MTK6261_DS platform module, supports simultaneous connection of multiple devices, up to 3.

For the MTK6261 and MTK2503 platform module, only supports the simultaneous connection of 1 device.

- The difference of the AT command

For the MTK6260 and MTK6261_DS platform module, access to the phone call status of the AT command is: AT+BTCLCC; the default SPP server mode is AT channel mode; Bluetooth open state will be saved when shutdown.

For the MTK6261 and MTK2503 platform module, access to the phone call status of the AT command is: AT+BTCLCCS; the default SPP server mode is the APP data mode; Bluetooth open state is not saved when shutdown.

AT commands of BLE are supported on MTK2503 platform module SIM868E.

2 AT Command

Command	Description
AT+BTHOST	Inquiry and set host device name
AT+BTSTATUS	Inquiry current BT device status
AT+BTPOWER	Power on or power off BT radio
AT+BTLPWR	Modify the Bluetooth transmit power
AT+BTPAIR	Pair BT device
AT+BTSCAN	Scan surrounding BT device
AT+BTUNPAIR	Unpair BT device
AT+BTCONNECT	Connect paired BT device
AT+BTDISCONN	Disconnect BT device
AT+BTGETPROF	Get profile provided by paired device
AT+BTACPT	Accept connecting request
AT+BTOPPACPT	Accept OPP service
AT+BTOPPPUSH	Push OPP object to paired device
AT+BTSPPSEND	Send data based on SPP service
AT+BTSPPGET	Get data based on SPP service
AT+BTATA	Answer incoming call
AT+BTATDL	Redial last number
AT+BTATH	Hung up voice call
AT+BTVGS	Configure voice volume
AT+BTVGM	Configure MIC volume
AT+BTATD	Dial up a voice call
AT+BTRSSI	Get RSSI of connected device
AT+BTVTS	Send DTMF tone
AT+BTCIND	Get status of smartphone
AT+BTCLCC	Get call status of smartphone
AT+BTPBSYNC	Sync phonebook from remote by BT
AT+BTPBF	Find name or number from remote by BT
AT+BTAVRCOP	AVRCP operation
AT+BTVIS	Set visibility of BT
AT+BTSPPCFG	SPP’s config
AT+BTPAIRCFG	Set BT pairing mode
AT+CPBFEX	Find name or number in module phonebook
AT+BTRING	Control ring playing transferered from phone
AT+BTACI	Set report mode of BT audio service state change
AT+BTHFGOP	Set action mode of MS when earphone button is pressed during BT link

AT+BTSPPURC	Set the report format of command +BTSPPSEND
AT+BTCLCCS	Get call status of smartphone
AT+BTSPPCFD	Set string of SPP switching work mode
AT+BTCOD	Set the bluthtooth class of device
AT+BLESREG	Register GATT Server
AT+BLESDREG	Derigister GATT Server
AT+BLESSAD	Add a service
AT+BLESSRM	Remove a service
AT+BLESSC	Add a characteristic to an existed service
AT+BLESSD	Add a descriptor to an existed service
AT+BLESSSTART	Start a service
AT+BLESSSTOP	Stop a service
AT+BLESLSTART	Start advertising
AT+BLESLSTOP	Stop advertising
AT+BLEADV	Set adverting parameters
AT+BLECPU	Connection parameters update
AT+BLESIND	Send an indication to a client
AT+BLESRSP	Send a response to a client's read or write operation
AT+BLEFMP	(De)Register a FMP Service
	Notify when connection's status change comes +BLEFMPCON
	Notify when a client's write request comes +BLEFMPWREQ
AT+BLEPXP	(De)Register PXP Service
	Notify when connection's status change comes +BLEPXPCON
	Notify when a Link loss alert comes +BLEPXPLLAT
	Notify when a a disconnection alert comes +BLEPXPDISAT

2.1 AT+BTHOST Inquiry and set host device name

AT+BTHOST Inquiry and set host device name	
Test command **AT+BTHOST=?**	Response **+BTHOST: (1-18)** **OK**
	Parameters See Write Command
Read command **AT+BTHOST?**	Response **+BTHOST: <name>,<address>** **OK**

	Parameters See Write Command
Write command **AT+BTHOST=<name>**	Response **OK**
	Parameters <**name**> device name <**address**> device address
Note	Max length of <name> is 18 bytes, and display in UTF-8 code.

2.2 AT+BTSTATUS Inquiry current BT device status

AT+BTSTATUS	**Inquiry current BT device status**
Test Command **AT+BTSTATUS=?**	Response **OK**
	Parameters See Read Command
Read Command **AT+BTSTATUS?**	Response If unpaired before: **+BTSTATUS: <status>** If paired before but unconnected: **+BTSTATUS: <status>** **P: <paired id>,<name>,<address>** If paired and connected: **+BTSTATUS: <status>** **P: <paired id>,<name>,<address>** **C: <connected id>,<name>,<address>,<profile name>** **OK**
	Parameters **<status>** 0 Initial 1 Disactivating 2 Activating 5 Idle 6 Inquiry 7 Inquiry Res Ind 8 Cancelling inquiry 9 Bonding 11 Pairing 12 Connecting 14 Deleting paired device 15 Deleting all paired device 19 Pairing confirm while passive pairing

	20 Waiting for remote confirm while passive pairing 25 Accepting connection 26 SDC refreshing 29 Setting host name **<paired id>** paired device ID **<connected id>** connected device ID <**name**> device name <**address**> device address **<profile name>** profile
Note	Max length of <name> is 18 bytes, 18 bytes in UTF-8 code

2.3 AT+BTPOWER Power on/off BT radio

AT+BTPOWER	**Power on/off BT radio**
Test Command **AT+BTPOWER =?**	Response **+BTPOWER: (**list of supported **<n>**s**)** **OK**
	Parameters See Write Command
Read Command **AT+BTPOWER ?**	Response **+BTPWR: <status>** **OK**
	Parameters See Write Command
Write Command **AT+BTPOWER =<n>**	Response **OK**
	parameter **<n>** 0 power off BT radio 1 power on BT radio
Note	After turning off, the BT radio shall not be re-opened until the status of BT is changed to 0. So wait for some seconds is needed. The status can be obtained by using AT+BTSTATUS.

2.4 AT+BTLPWR Modify the Bluetooth transmit power

AT+BTLPWR	**Modify the Bluetooth transmit power**
Read Command **AT+BTLPWR?**	Response **+BTPWR: <status>**

	OK
	Parameters See Write Command
Test Command **AT+BTLPWR=?**	Response **+BTPOWER:** (0-7) **OK**
	Parameters See Write Command
Write Command **AT+BTLPWR=<n>**	Response **OK**
	parameter **<n>** 0 reset power status to default 1-7 the class of Bluetooth transmit power

2.5 AT+BTPAIR Pair BT device

AT+BTPAIR Pair BT device	
Test Command **AT+BTPAIR=?**	Response **+BTPAIR:** 0,(list of supported **<device ID>**s) **+BTPAIR:** 1,(list of supported **<confirm>**s) **+BTPAIR:** 2,(length of supported **<passkey>**s) **OK**
	Parameters See Write Command
Write Command 1) active **AT+BTPAIR=0, <device ID>** 2) passive with digital key request **AT+BTPAIR=1, <confirm>** 3) passive with passkey request **AT+BTPAIR=2, <passkey>**	Response **OK** If digital key exchanged **+BTPAIRING: <name>,<address>,<passcode>** If passkey exchanged: **+BTPAIRING: <name>,<address>** If passive mode with succees: **+BTPAIR: <id>,<name>,<address>** If passive mode with failure: **+BTPAIR: 0**
	Parameters **<device ID>** BT device ID **<confirm>** 1 accept 0 reject **<passkey>** passkey, length is（4-16）

	<id> 0 paired failed >=1 paired deivce ID <**name**> BT device name <**address**> BT device address **<passcode>** Digital password
	URC If there is incoming request: **+BTPAIRING: <name>,<address>,<passcode>** or **+BTPAIRING: <name>,<address>** Parameters <**name**> device name <**address**> device address **<passcode>** digital password
Note	1. Max length of <name> is 18 bytes, 18 bytes in UTF-8 code 2. Pairing timeout is around 15s each side

2.6 AT+BTUNPAIR Unpair BT device

AT+BTUNPAIR	**Unpair BT device**
Test Command **AT+BTUNPAIR =?**	Response **+BTUNPAIR:** (list of supported **<device ID>**s) **OK**
	Parameter See Write Command
Write Command **AT+BTUNPAIR =<device ID>**	Response **OK**
	Parameter **<device ID>** Paired Device ID. 0 delete all the paired device 1 delete the the paired device corresponding to ID

2.7 AT+BTSCAN Scan surrounding BT device

AT+BTSCAN	**Scan surrounding BT device**
Test Command **AT+BTSCAN=?**	Response **+BTSCAN:** (list of supported **<switch>**s)**,** (list of supported **<Timer>**s)

	OK
	Parameters See Write Command
Wrtie Command **AT+BTSCAN=<switch>[,<Timer>]**	Response **OK** If BT device scanned: **+BTSCAN: <status>,<device ID>,<name>,<address>,<rssi>** If terminate: **+BTSCAN: <status>**
	Parameters **<switch>** 1 start 0 stop **<status>** 0 BT device found 1 scanning finished 2 scanning stop 3 scanning failed **<Timer>** scanning time 10-60s **<device ID>** BT device ID scanned <**name**> BT device name <**address**> BT device address **<rssi>** -127…0 RSSI value of BT device
Note	1. Max length of <name> is 18 bytes, 18 bytes in UTF-8 code 2. If <timer> ommited, the default value is 30s

2.8 AT+BTCONNECT Connect paired BT device

AT+BTCONNECT	**Connect paired BT device**
Test Command **AT+BTCONNECT=?**	Response **+BTCONNECT:** (list of supported **<device ID>**s), (list of supported **<profile ID>**s) **OK**
	Parameters See Write Command
Write Command **AT+BTCONNECT=<device ID>,<profile ID>**	Response **OK** If OK: **+BTCONNECT: <id>,<name>,<address>,<profile name>** If failed: **+BTCONNECT: 0**

	Parameters **<device ID>** ID of paired BT device **<profile ID>** BT profile ID **<id>** ID of connected BT device <**name**> BT device name <**address**> BT device adress **<profile name>** BT device service name
Note	1. Max length of <name> is 18 bytes, 18 bytes in UTF-8 code 2. Connection timeout is around 20s 3. if incoming request, there will be URC **+BTCONNECING: <address>,<profile name>**

2.9 AT+BTDISCONN Disconnect BT connection

AT+BTDISCONN	**Disconnect BT connection**
Test Command **AT+BTDISCONN=?**	Response **+BTDISCONN:** (list of supported **<device ID>**s) **OK**
	Parameters See Write Command
Write Command **AT+BTDISCONN=<device ID>**	Response **OK** **+BTDISCONN: <name>,<address>,<profile name>**
	Parameters **<device ID>** connected device ID <**name**> device name <**address**> devie address **<profile name>** profile service
Note	1. Max length of <name> is 18 bytes, 18 bytes in UTF-8 code 2. If disconnected by remote, there still be URC: +BTDISCONN

2.10 AT+BTGETPROF Get profile provided by paired device

AT+BTGETPROF	**Get profile provided by paired device**
Test Command **AT+BTGETPROF=?**	Response **+BTGETPROF:** (list of supported **<device ID>**s) **OK**
	Parameters See Write Command

Write Command **AT+BTGETPROF=<device ID>**	Response **OK** **+BTGETPROF: <profile ID>,<profile name>**
	Parameters **<device ID>** Paired Device ID **<profile ID>** profile ID **<profile name>** profile name

2.11 AT+BTACPT Accept connecting request

AT+BTACPT Accept connecting request	
Test Command **AT+BTACPT=?**	Response **+BTACPT:** (list of supported **<confirm>**s) **OK**
Write Command **AT+BTACPT=<confirm>**	Response **OK** If connected successfully, then will report: **+BTCONNECT: <id>,<name>,<address>,<profile name>** If connecting failed: **+ BTDISCONN: <name>,<address>,<profile name>**
	Parameters **<confirm>** 1 accept 0 reject **<id>** >0 connected device ID <**name**> device name <**address**> device address **<profile name>** profile name
	URC If incoming connecting request: **+BTCONNECTING: <address>,<profile name>** Parameters <**address**> device address **<profile name>** profile name
Note	Max length of <name> is 18 bytes, 18 bytes in UTF-8 code

2.12 AT+BTOPPACPT Accept OPP service

AT+BTOPPACPT	**Accept OPP service**
Test Command **AT+BTOPPACPT=?**	Response **+BTOPPACPT:** (list of supported **<confirm>**s),(list of supported<**drv**>) **OK**
Write Command **AT+BTOPPACPT=<confirm>[,<drv>]**	Response **OK** **+BTOPPPUSH: <status>**
	Parameters **<confirm>** 1 Accept 0 Reject **<drv>** 0 internal flash memory 1 external memory card **<status>** 0 failed 1 successful
	URC: If there has an incoming opp file, there will be a URC report. **+BTOPPPUSHING: <name>,<file name>** Parameters <**name**> device name **<file name>** file name
Note	1. Max length of <name> is 18 bytes, 18 bytes in UTF-8 code 2．File is stored in path: C:\User\BtReceived\ for internal memory card, D:\BtReceived\ for external memory card. At the first time to use SD card, customer must execute “AT+SD2PCM=0” and “AT&W”, then reboot the module.

2.13 AT+BTOPPPUSH Push OPP object to paired device

AT+BTOPPPUSH	**Push OPP object to paired device**
Test Command **AT+BTOPPPUSH=?**	Response **+BTOPPPUSH:** (list of supported **<device ID>**s), **(**length of supported **<string>**s) **OK**

	Parameters See Write Command
Write Command **AT+BTOPPPUSH=<device ID>,<string>**	Response **OK** +**BTOPPPUSH: <para>**
	Parameters **<device ID>** Paired Device ID **<string>** file name include complete path, lenght（4-259） **<para>** 0 Send failed 1 Send successfully 2 Server issue
Note	

2.14 AT+BTSPPGET Get data based on SPP service

AT+BTSPPGET	**Get data based on SPP service**
Test Command **AT+BTSPPGET=?**	Response **+BTSPPGET: (**list of supported **<command>**s**), (**list of supported **<connectId>**), (list of supported **<reqLength>**s**), (**list of supported **<showWithHex>**s) **OK**
	Parameters See Write Command
Read Command **AT+BTSPPGET?**	Response **+BTSPPGET: <command>** **OK**
	Parameters See Write Command
Write Command 1).If AT+BTSPPCFG=”MC”,2 response 1(Enable multi-connect) **AT+BTSPPGET=<command>[,<connectId>][,<reqLength>][,<showWithHex>]**	Response **OK** **or** **ERROR** If command value is 2,return: **+BTSPPGET: <connectId>,<cnfLen1>** **OK** If command value is 3,return: **+BTSPPGET: <connectId>,<cnfLen1>[,<data string>]**

2).If AT+BTSPPCFG=”MC”,2 response 0(Disable multi-connect) **AT+BTSPPGET=<command>[,<reqLength>][,<showWithHex>]**	**OK**
	Parameters **<command>** 0 Auto mode. Data will be output in decimal system. 1 Manual mode. There will be an indication when first package arrives. 2 Inquiry data length in manual mode.If multi-connect enabled,this command need parameter <**connectId**>. 3 Getting data in manual mode. If multi-connect enabled,this command need parameter <**connectId**>.You can input params of <**reqLength**> and <**showWithHex**> when you need. **<reqLength>** 1-1024 , the length of data requested, only valid in manual mode <**showWithHex**> 1, displayed in hex, only valid in manual mode **<connectId>** connection`s ID **<cnfLen1>** 0-1024, character length **<data string>** string printed
Note	URC When the module receives data by SPP,there will be URC report: 1. Auto mode **+BTSPPDATA: <connectId>,<cnfLen2>,<data string>** 2. Manual mode **+BTSPPMAN: <connectId>** Parameter **<cnfLen2>** 1-1024, length of printed character

2.15 AT+BTSPPSEND Send data based on SPP service

AT+BTSPPSEND	**Send data based on SPP service**
Write Command 1).If AT+BTSPPCFG=”MC”,2 response 1(Enable multi-connect) **AT+BTSPPSEND=<connectId>,<length>** 2).If AT+BTSPPCFG=”MC”,2 response	Response > If successful, **SEND OK** If failed, **SEND FAIL** Or if this connectId is not allowed to send data, **ERROR**
	Parameters **<connectId>** connection`s ID.If disable multi-connection, this param is no need. **<length>** 1-1024, the length of data will be sent.

0(Disable multi-connect) **AT+BTSPPSEND=<length>**	When the length of inputing data is up to <length> specified, the package will be sent out automatically.
Execute Command **AT+BTSPPSEND**	Response > If successful, **SEND OK** Or failed, **SEND FAIL** Or if this connectId is not allowed to send data, **ERROR**
	1.If multi-connection function is enabled, this command will be disabled. 2.In this mode, <Ctrl+z> will send the package immediately, and ESC will quit the process.

2.16 AT+BTATA Answer incoming call

AT+BTATA Answer incoming call	
Execute Command **AT+BTATA**	Response **OK**
	URC If there is incoming Call on remote phone, will report below: **BTRING**
Note	When module connected with smartphone as an earphone,if here comes incoming call,the call would be answered through this command

2.17 AT+BTATDL Redial last number

AT+BTATDL Redial last number	
Execute Command **AT+BTATDL**	Response **OK**
Note	When module connected with smartphone as an earphone,would redial last number through this command

2.18 AT+BTATH Hung up voice call

AT+BTATH Hung up voice call	
Execute Command **AT+BTATH**	Response **OK**

Note	When module connected with smartphone as an earphone, the incoming call would be hung up through this command

2.19 AT+BTVGS Configure voice volume

AT+BTVGS Configure voice volume	
Test Command **AT+BTVGS=?**	Response **+BTVGS: (<gain>** range**)** **OK**
	Parameters See Write Command
Read Commnad **AT+BTVGS?**	Response **+BTVGS: <gain>** **OK**
	Parameters See Write Command
Write Command **AT+BTVGS=<gain>**	Response **OK**
	Parameter <**gain**> volume This command is used configure call volume when the module is connected with smartphone as an earphone
Note	For some smartphone,after connected with BT earphone,the current call volume may not be transmitted to earphone,thus the return value of the read command may be 0.But after setting once,the value would be correct.

2.20 AT+BTVGM Configure MIC gain level

AT+BTVGM Configure MIC gain level	
Test Command **AT+BTVGM=?**	Response **+BTVGM: (<gain>** range**)** **OK**
Read Command **AT+BTVGM?**	Response **+BTVGM: <gain>** **OK**

Write Command **AT+BTVGM=<gain>**	Response **OK**
	Parameter <**gain**> MIC gain level This command is used set MIC volume when the module is connected with smartphone as an earphone
Note	For some smartphone,after connected with BT earphone,the current MIC volume may not be transmitted to earphone,thus the return value of the read command may be 0.But after setting once,the value would be correct.

2.21 AT+BTATD Dial voice call

AT+BTATD Dial voice call	
Test Command **AT+BTATD=?**	Response **+BTATD: (<number>** length range**)** **OK**
Write Command **AT+BTATD=<number>**	Response **OK**
	Parameter <**number**> phone number Module as earphone connected to smartphone, this command could make an outgoing call
Note	

2.22 AT+BTRSSI Get RSSI of connected BT device

AT+BTRSSI Get RSSI of connected BT device	
Test Command **AT+BTRSSI=?**	Response **+BTRSSI:** (list of supported **<device ID>**s**)** **OK**
Write Command **AT+BTRSSI=<device ID>**	Response **+BTRSSI: <rssi>** **OK**
	Parameters **<device ID>** Connected Device ID **<rssi>** -127…0 RSSI value of BT device

Note	RSSI value is negative, the smaller value represents the worse signal

2.23 AT+BTVTS Send DTMF tone

AT+BTVTS Send DTMF tone	
Test Command **AT+BTVTS=?**	Response **+BTVTS: (<dtmf>**'s cope**)** **OK**
Write Command **AT+BTVTS=<dtmf>**	Response **OK**
	Parameter <**dtmf**> DTMF tone
Note	When module connected with smartphone as an earphone,would send DTMF tone through this command

2.24 AT+BTCIND Get status of smartphone

AT+BTCIND Get status of smartphone	
Test Command **AT+BTCIND=?**	Response **+BTCIND: (0,1)** **OK**
Write Command **AT+BTCIND=<mode>**	Response **OK**
	Parameter **<mode>** 1 auto report open 0 auto report close
	Unsolicited Result Code When **<mode>**=1, any changed in **<service>,<call>,<call_setup>,<held>,<signal>,<roam>,<battchg>** , an unsolicited result code is returnd： **+BTCIND: 1,<service>,<call>,<call_setup>,<held>,<signal>,<roam>,<battchg>**
Read Command **AT+BTCIND?**	Response **+BTCIND:**

	<mode>,<service>,<call>,<call_setup>,<held>,<signal>,<roam>,<battchg> **OK**
	Parameters **<service>** 0 no net service 1 net service is normal **<call>** 0 not active 1 active **<call_setup>** 0 set up complete 1 incoming call 2 outgoing call 3 remote alert **<held>** 0 no held call 1 active calls be placed or switched 2 active calls be palced and no active call **<signal>** 0..5 net work signal **<roam>** 0 no roaming 1 in roaming **<battchg>** 0..5 power level
Note	When module connected with smartphone as an earphone, these statuses can be getted.

2.25 AT+BTCLCC Get call status of smartphone

AT+BTCLCC Get call status of smartphone	
Test Command **AT+BTCLCC=?**	Response **OK**
Read Command **AT+BTCLCC?**	Response **OK** When call is active: **+BTCLCC: <index>,<dir>,<stat>,<mode>,<mpty>,<number>,<type>** **…** When no call: **+BTCLCC: 0**
	Parameters **<idx>** 1..7 Call identification number **<dir>** 0 Mobile originated (MO) call 1 Mobile terminated (MT) call **<stat>** State of the call:

	0 Active 1 Held 2 Dialing(MO call) 3 Alerting (Mo call) 4 Incoming (MT call) 5 Waiting (MT call) **<mode>** Bearer/tele service 0 Voice 1 Data 2 Fax **<mpty>** 0 Call is not one of multiparty (conference) call parties 1 Call is one of multiparty (conference) call parties **<number>** String type (string should be included in quotation marks) phone number in format specified by <type>. **<type>** Type of address
Note	● If there are mulit calls, multi "**+BTCLCC**" will be reported, but **<index>** is different ● MTK_6261 platform does not support this command.

2.26 AT+BTPBSYNC Sync phonebook from remote by BT

AT+BTPBSYNC Sync phonebook from remote by BT	
Test Command **AT+BTPBSYNC=?**	Response **+BTPBSYNC: (0,1),(1-10),(0,1),(0,1),(0,1)** **OK**
Write Command **AT+BTPBSYNC=<mode>,<storage>,<loc>[,<loc_phb>[,<loc_mode>]]**	Response **OK** If sync phonebook succeed in mode 0 **+BTPBSYNC: <mode>,<result>,<length>** If sync phonebook failed in mode 0 **+BTPBSYNC: <mode>,<result>** If in mode 1 **+BTPBSYNC: <mode>,<sync2loc_result>,<succ_num>,<fail_num>** If error is related to ME functionality: **+CME ERROR: <err>**
	Parameterss **<mode>** sync mode

0 Get remote phonebook and save in file system. This file will store phonebook in VCARD format.

1 Add phonebook records to ME or SM phonebook from VCARD file. Should get remote phonebook file by mode 0 first.

<storage> Phonebook storage to sync.

1 phonebook on phone storage
2 incoming call list on phone storage
3 outgoing call list on phone stroage
4 missed call list on phone storage
5 all call list in storage 2, 3, 4
6 phonebook on sim card
7 incoming call list on sim card
8 outgoing call list on sim card
9 missed call list on sim card
10 all call list in storage 7, 8, 9

<loc> file saved in ROM or SD card.

0 saved in ROM
file will be saved in "C:\user\bt\remotePb<n>.txt"
1 saved in SD card
file will be saved in "D:\bt\remotePb<n>.txt"

The 'n' in angle brackets is corresponding with **<storage>**, from 1 to 10.

<result> sync phonebook result

0 sync phonebook succeed
1 fail to get phonebook on remote phone
2 save phonebook fail

<length> file length

<loc_phb> save phb file to ME or SM. Just use in mode 1.

0 SM phonebook
1 ME phonebook

<loc_mode> append or overwrite local phonebook. Just use in mode 1.

0 append mode. Phonebook records in VCARD file will add in not used index of local phonebook.

1 overwrite mode. Local phonebook records will be delete first.

<sync2loc_result> sync result in mode 1

0 sync in mode 1 succeed
1 function has already run
2 local phonebook(ME or SM) full
3 not enough memory
4 error when read VCARD file.
5 error when analyze VCARD file
6 local phonebook not ready
7 sim card not ready

<succ_num> num of phonebook records succeed add to local phonebook

	<fail_num> num of phonebook records failed add to local phonebook. The most common reason of add failed is name and number field of VCARD phonebook record is both empty
Note	

2.27 AT+BTPBF Find name or number from remote by BT

AT+BTPBF Find name or number from remote by BT	
Test Command **AT+BTPBF=?**	Response **+BTPBF: (0,1),(32,64),(1-10),(0-2)** **OK**
Write Command **AT+BTPBF=<mode>,<string>[,<storage>[,<order>]]**	Response **OK** If find name by number succeed **+BTPBF: 1,<phb_total>** **+BTPBF: 1,<phb_index>,<name>** **...** If find number by name succeed **+BTPBF: 0,<phb_total>** **+BTPBF: 0,<phb_index>,<num_total>** **+BTPBF: 0,<phb_index>,<num_index>,<number>,<type>** **...** If find name by number failed or find number by name faild at get list step. **+BTPBF: <mode>,<error>** If find number by name failed at get entry step **+BTPBF: <mode>,<phb_index>,<error>** If error is related to ME functionality: **+CME ERROR: <err>**
	Parameters **<mode>** find mode 0 find number by name 1 find name by number **<string>** string to be searched. If use mode 0, it should be alphanumeric ASCII text string up to 32 characters If use mode 1, it should be ucs2(big endian) value form with

	alphanumeric ASCII text string. Max length is 64 **<storage>** see **AT+BTPBSYNC**. Default value is 1. **<order>** search results order 0 order by indexed 1 order by alpha 2 order by sound **<phb_total>** total number of phonebook record be found. We support max 5 phonebook records. **<phb_index>** index of phonebook record **<name>** The name found by number. It will be ucs2(big endian) value. **<num_total>** total number of <number> in one phonebook record. We support max 4 number in one phonebook record. **<num_index>** index of <number> **<number>** The number found by name. **<type>** type of **<number>** 0 voice 1 cell 2 home 3 work 4 fax **<error>** find error 255 fail to find
Note	The support of this function on different brands of mobile phone is different.

2.28 AT+BTAVRCOP AVRCP operation

AT+BTAVRCOP	**AVRCP operation**
Test Command **AT+BTAVRCOP=?**	Response **+BTAVRCOP: (0-STOP,1-PLAY,2-PAUSE,3-FORWARD,4-BACKWARD,5-VOL_UP,6-VOL_DOWN)** **OK**
Write Command **AT+BTAVRCOP=<operator>**	Response **OK** If error is related to ME functionality: **+CME ERROR: <err>**
	Parameters **<operator>** 0 stop the music 1 play the music

	2 pause the music 3 play the next song 4 play the back song 5 increase the volume 6 decrease the volume
Note	

2.29 AT+BTVIS Set visibility of BT

AT+BTVIS Set visibility of BT	
Test Command **AT+BTVIS=?**	Response **+BTVIS: (0,1)** **OK**
Read Commnad **AT+BTVIS?**	Response **+BTVIS: <visibility>** **OK**
	Response See Write Command
Write Command **AT+BTVIS=<visibility>**	Response **OK**
	Parameters <**visibility**> visibility of BT 1 open visibility 0 close visibility
Note	

2.30 AT+BTSPPCFG SPP configuration

AT+BTSPPCFG SPP configuration	
Test Command **AT+BTSPPCFG=?**	Response **+BTSPPCFG: (list of supported <btSppCfg>s)** **OK**
Write Command **AT+BTSPPCFG=<btSppCfg>,<m**	Response **OK** Or

ode>	**ERROR**
	Parameters **<btSppCfg>** “MC” Multi-connection, enable this function to make the module support to connect double SPP’s client at the same time. “TT” Transparent transmission mode, this function makes the module automatically enter the data mode after the SPP connection is established. <**mode**> 0 Disable 1 Enable 2 Query
Read Command **AT+BTSPPCFG ?**	Response Every SPP’s link has been connected as server,output: **+BTSPPCFG: S,<connectId>,<serverMode>** Every SPP’s link has been connected as client,output: **+BTSPPCFG: C,<connectId>** **OK**
	Parameters **<connectId>** connection`s ID **<serverMode>** 0 AT mode 1 APP mode
Note	In AT mode, module of server can’t execute AT+BTSPPSEND and AT+BTSPPGET commands. In APP mode, module of server can execute AT+BTSPPSEND and AT+BTSPPGET commands.

2.31 AT+BTPAIRCFG Set BT pairing mode

AT+BTPAIRCFG	**Set BT pairing mode**
Test Command **AT+BTPAIRCFG=?**	Response **+BTPAIRCFG: (**list of supported **<mode>**s**)** **OK**
	Parameters See Write Command
Read Command **AT+BTPAIRCFG?**	Response If **mode**=1, the notification information is: **+BTPAIRCFG: <mode>,<pin_code>** **OK** If **mode**=0 or 2, the notification information is: **+BTPAIRCFG: <mode>**

	OK
	Parameters See Write Command
Write Command 1) if PIN-Code inputted by manual while pairing **AT+BTPAIRCFG=1[,<pin_code>]** 2）if using random PIN-Code while pairing **AT+BTPAIRCFG=<mode>**	Response **OK**
	Parameters **<mode>** 0 random PIN-Code, and need confirm the pairing request 1 PIN-Code inputted by manual 2 random PIN-Code, and response the pairing request automatic **<pin_code>** PIN-Code, the length is four. default value is 0000
Note	When **mode** is 0 or 2, it is random PIN-Code When mode is 2, it has no **+BTPAIRING** information, and response the pairing request automatic; When mode is 0, it has **+BTPAIRING** informtioan, and need input **AT+BTPAIR=1,1** to confirm pairing request. The setting will be valid after reboot.

2.32 AT+CPBFEX Find name or number in module phonebook

AT+CPBFEX Find name or number in module phonebook	
Test Command **AT+CPBFEX=?**	Response **+CPBFEX: (0,1),40** **OK**
Write Command **AT+CPBFEX=<mode>,<value>**	Response TA returns phone book entries, which contains alphanumeric string <text>. **[+CPBFEX: <text>]** **OK**
	Parameters <**mode**> find mode 0 find name by number 1 find number by name

	<value> String type field of maximum length 40. When select <**mode**> 1, <**value**> should set in current TE character set specified by +CSCS. **<text>** String type field. When select <**mode**> 0, <**text**> will return in current TE character set specified by +CSCS.
Note	AT+CPBFEX will only return the first find result. AT+CPBFEX could find name or number which CPBFEX could not display when use BTPBSYNC sync PHB to ME phonebook.

2.33 AT+BTRING Control ring playing transferred from phone

AT+BTRING Control ring playing transferered from phone	
Test Command **AT+BTRING=?**	Response **+BTRING: (0,1)** **OK**
Read Command **AT+BTRING?**	Response **+BTRING: <mode>** **OK**
	Parameters See Write Command
Write Command **AT+BTRING=<mode>**	Response **OK**
	Parameters <**mode**> 0 not play ring transferred from mobile phone 1 play ring transferred from mobile phone
Note	● This command takes effect when module acts as earphone in BT link. ● This command doesn't support power off save.

2.34 AT+BTACI Set report mode of BT audio service state change

AT+BTACI Set report mode of BT audio service state change	
Test Command **AT+BTACI=?**	Response **+BTACI: (0,1)** **OK**
Read Command	Response

AT+BTACI?	**+BTACI: <mode>,<state>** **OK**
	Parameters See Write Command
Write Command **AT+BTACI=<mode>**	Response **OK**
	Parameters <**mode**> set URC report or not when audio service state change 0 no URC report when audio service state change 1 URC report when audio service state change **<state>** BT audio State 0 ilde 1 SCO service 2 A2DP service
	Unsolicited Result Code When <**mode**> is set to 1, URC **+BTACI: <state>** will report when BT audio service state change
Note	This command doesn't support power off save.

2.35 AT+BTHFGOP Set action mode of MS when earphone button is pressed during BT link

AT+BTHFGOP	Set action mode of MS when earphone button is pressed during BT link
Test Command **AT+BTHFGOP=?**	Response **+BTHFGOP: (0-2)** **OK**
Read Command **AT+BTHFGOP?**	Response **+BTHFGOP: <mode>,<event>** **OK**
	Parameters See Write Command
Write Command **AT+BTHFGOP=<mode>**	Response **OK**
	Parameters

	<**mode**> Set action mode of MS when earphone button is pressed during BT link 0 MS acts normally 1 URC is reported and RI pin will be pulled down for 120ms,MS will suspend earphone events and take no action. 2 Clear event to 0,mode not change **<event>** Earphone event 0 No event 1 Call redial 2 Answer incoming call 3 Call hang up
	Unsolicited Result Code When <**mode**> is set to 1, URC **+BTHFGOP: <event>** will report when earphone event has been changed.
Execute Command **AT+BTHFGOP**	Execute command will restore earphone events of MS. Execute command can't execute when no event. Response **OK**
Note	This command doesn't support power off save.

2.36 AT+BTSPPURC Set the report format of command +BTSPPSEND

AT+BTSPPURC	Set the report format of command +BTSPPSEND
Test Command **AT+BTSPPURC =?**	Response **+BTSPPURC: (0-1)** **OK**
Read Command **AT+BTSPPURC ?**	Response **+BTSPPURC: <mode>,<succ_str>,<fail_str>** **OK**
	Parameters See Write Command
Write Command **AT+BTSPPURC =<mode>**	Response **OK**
	Parameters <**mode**> Set the report format of command +BTSPPSEND 0 Common URC of data mode

	1 Special URC of Bluetooth data mode <**succ_str**> SEND OK Common URC for success BT SEND OK Special URC for success **<fail_str>** SEND FAIL Common URC for failure BT SEND FAIL Special URC for failure
Note	This command doesn't support power off save. The default value of <mode> is 0.

2.37 AT+BTCLCCS Get call status of smartphone

AT+BTCLCCS Get call status of smartphone	
Test Command **AT+BTCLCCS=?**	Response **+BTCLCCS: (0,1)** **OK**
	Parameters See Write Command
Write Command **AT+BTCLCCS=<mode>**	Response **OK**
	Parameters **<mode>** Auto report state 1 Active 0 Deactive
	Unsolicited Result Code When **<mode>** is set to 1， URC will report when call state change: **+BTCLCCS: 1,<call_stat>,<number>,<call_id>**
Read Command **AT+BTCLCCS?**	Response **+BTCLCCS: <mode>** OK
	Parameters See Write Command
Excute Command **AT+BTCLCCS**	Response **OK** When call is active: **+BTCLCCS: <mode>,<call_stat>,<number>,<call_id>** **…** When no call:

	+BTCLCCS: <mode>,0,,0
	Parameters **<mode>** Auto report state 1 Active 0 Deactive **<call_stat>** state of call 0 Idle 1 Dialing(MO call) 2 Incoming (MT call) 4 Active 8 Hold **<number>** String type (string should be included in quotation marks) phone number in format specified by <type>. **<call_id>** 1..7 Call identification number
Note	● If there are mulit calls, multi "**+BTCLCCS**" will be reported, but **<index>** is different ● Only MTK_6261 platform support this command.

2.38 AT+BTSPPCFD Set string of SPP switching work mode

AT+BTSPPCFD	**Set string of SPP switching work mode**
Test Command **AT+BTSPPCFD=?**	Response **+BTSPPCFD: (**list of supported **<switchStr>)** **OK**
	Parameters See Write Command
Write Command **AT+BTSPPCFD=<switchStr>**	**OK** or **ERROR**
	Parameters **<switchStr>** String used to switch work mode from AT mode to data mode
Read Command **AT+BTSPPCFD?**	Response **+BTSPPCFD: <switchStr>** **OK**
	Parameters See Write Command
Note	The usage of this command depends on the model of modules: 1. When any module except SIM800C acts as the SPP server, the default

	connection type is AT mode. User needs to input special strings in order to switch to data mode. If the string is null (AT+BTSPPCFD=“”), SPP server will directly enter data mode after any data is received from client during the next connection. 2. When SIM800C acts as the SPP server, the default connection type is APP data mode. User needs to input special strings in order to switch to the AT mode. If the string is null (AT+BTSPPCFD=“”), SPP server will never enter into the data mode.

2.39 AT+BTCOD Set the Bluetooth Class of Device

AT+BTCOD Set the Bluthtooth Class of Device	
Test Command **AT+BTCOD=?**	Response **OK**
	Parameters See Write Command
Write Command **AT+BTCOD=<en>[,<mjr_srv>[,<mjr_cls>[,<mnr_cls>]]]**	Response **OK** or **ERROR**
	Parameters **<en>** 0 Disable customized COD 1 Enable customized COD **<mjr_srv>** Major service code **<mjr_cls>** Major class code **<mnr_cls>** Minor class code
Read Command **AT+BTCOD?**	Response **+BTCOD: <en>,<mjr_srv>,<mjr_cls>,<mnr_cls>** **OK**
	Parameters See Write Command
Note	The setting does not support power-off preservation. This command only be used when the Bluetooth is power down.

2.40 AT+BLESREG Register GATT Server

AT+BLESREG Register GATT Server	
Test Command **AT+BLESREG=?**	Response **OK**
Execute Command	Response

AT+BLESREG	**+BLESREG: <server_index>,<user_id>** **OK** or **ERROR**
	Parameterss <**server_index**> Server index <**user_id**> User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’ }. Max length of it is 32.
Read Command **AT+BLESREG?**	Response **+BLESREG: <server_index>,<user_id>** **OK**
	Parameters See Execute Command
Note	The user id will be generated automatically.

2.41 AT+BLESDREG Deregister GATT Server

AT+BLESDREG	**Deregister GATT Server**
Test Command **AT+BLESDREG =?**	Response **OK**
Write Command **AT+BLESDREG =<server_index>**	Response **+BLESDREG: <server_index>,<user_id>** **OK** or **ERROR**
	Parameters <**server_index**> Server index <**user_id**> User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’ }. Max length of it is 32.
Read Command **AT+BLESDREG ?**	Response **OK**
	Parameters See Write Command
Note	

2.42 AT+BLESSAD Add a service

AT+BLESSAD Add a service	
Test Command **AT+BLESSAD=?**	Response **OK**
Write Command **AT+BLESSAD=<server_index>,<uuid>,<num_handles>,<is_primary>,<inst>**	Response **+BLESSAD: <service_index>,<user_id>,<uuid>,<is_primary>,<inst>,<service_handle>** **OK** or **ERROR**
	Parameters <**server_index**> Server index <**service_index**> Service index <**user_id**> user id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { '0'~'9','a'~'f','A'~'F' }. Max length of it is 32. <**uuid**> The UUID of the service, a string with hex value, max length is 32, min length is 4. <**num_handles**> Number of handles of this service. Dec format. 1~30. Should be larger than num of services + 2* num of Chars + num of descriptor. <**is_primary**> 0 Not primary service 1 Primary service <**inst**> Instance id of this UUID. Dec format. <**service_handle**> The handle of this service. Dec format.
Read Command **AT+BLESSAD?**	Response **+BLESSAD: <service_index>,<user_id>,<uuid>,<is_primary>,<inst>,<service_handle>** **OK**
	Parameters See Write Command
Note	

2.43 AT+BLESSRM Remove a service

AT+BLESSRM Remove a service

Test Command **AT+BLESSRM=?**	Response **OK**
Write Command **AT+BLESSRM=<service_index>**	Response **+BLESSRM: <service_index>,<user_id>,<uuid>,<service_handle>** **OK** or **ERROR**
	Parameters <**service_index**> Service index <**user_id**> User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’ }.Max length of it is 32. <**uuid**> The UUID of the service, a string with hex value, max length is 32, min length is 4. <**service_handle**> The handle of this service. Dec format.
Read Command **AT+BLESSRM?**	Response **OK**
	Parameters See Write Command
Note	

2.44 AT+BLESSC Add a characteristic to an existed service

AT+BLESSC Add a characteristic to an existed service	
Test Command **AT+BLESSC=?**	Response **OK**
Write Command **AT+BLESSC=<service_index>,<char_uuid>,<inst>,<prop>,<permission>**	Response **+BLESSC: <char_index>,<user_id>,<service_handle>,<char_uuid>,<inst>,<char_handle>** **OK** or **ERROR**
	Parameters <**service_index**> Service index <**char_index**> Characteristic index <**user_id**> user id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’ }. Max length of it is 32. <**service_handle**> The handle of this service. Dec format.

	<**char_uuid**> The UUID of characteristic, a string with hex value, max length is 32, min length is 4. <**inst**> Instance id of this UUID. Dec format. <**prop**> Properties of this characteristic. Dec format. (0 - 4294967295) Default 0 Broadcast 1 Read 2 Write without response 4 Write 8 Notify 16 Indicate 32 Signed write 64 Extended properties 128 <**permission**> Permission of this characteristic. Dec format. (0 - 4294967295) Read 1 Read with encrypted protection 2 Read with MITM protection 4 Write 8 Write with encrypted protection 16 Write with MITM protection 32 Signed write 64 Signed write with MITM protection 128 <**char_handle**> The handle of this Characteristic. Dec format.
Read Command **AT+BLESSC?**	Response **+BLESSC: <char_index>,<user_id>,<service_handle>,<char_uuid>,<inst>,<prop>,<permission>,<char_handle>** **OK**
	Parameters See Write Command
Note	

2.45 AT+BLESSD Add a descriptor to an existed service

AT+BLESSD Add a descriptor to an existed service	
Test Command **AT+BLESSD=?**	Response **OK**
Write Command **AT+BLESSD=<service_index>,<d**	Response **+BLESSD: <desc_index>,<user_id>,<service_handle>,<desc_uuid>,<inst>,<desc_handle>**

esc_uuid>,<inst>, **<permission>**	**OK** or **ERROR**
	Parameters <**service_index**> Service index <**desc_index**> descriptor index <**user_id**> user id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { '0'~'9','a'~'f','A'~'F' }. Max length of it is 32. <**service_handle**> The handle of this service. Dec format. <**desc _uuid**> The UUID of the descriptor, a string with hex value, max length is 32, min length is 4. <**inst**> Instance id of this UUID. Dec format. <**permission**> Permission of this descriptor. Dec format. (0 - 4294967295) <**desc_handle**> Handle of this descriptor. Dec format.
Read Command **AT+BLESSD?**	Response **+BLESSD: <desc_index>,<user_id>,<service_handle>,<desc_uuid>,<inst>,<permission>,<desc_handle>** **OK**
	Parameters See Write Command
Note	

2.46 AT+BLESSSTART Start a service

AT+BLESSSTART Start a service	
Test Command **AT+BLESSSTART=?**	Response **OK**
Write Command **AT+BLESSSTART=<service_index>,<transport>**	Response **+BLESSSTART: <service_index>,<user_id>,<service_handle>** **OK** or **ERROR**
	Parameters <**service_index**> Service index <**transport**> Transport way to start service. <u>0</u> LE

	1 BR/EDR 2 Dual mode **<user_id>** User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { '0'~'9','a'~'f','A'~'F' }. Max length of it is 32. **<service_handle>** The handle of this service. Dec format.
Read Command **AT+BLESSSTART?**	Response **+BLESSSTART: <service_index>,<user_id>,<service_handle>** **OK**
	Parameters See Write Command
Note	

2.47 AT+BLESSSTOP Stop a service

AT+BLESSSTOP	**Stop a service**
Test Command **AT+BLESSSTOP=?**	Response **OK**
Write Command **AT+BLESSSTOP=<service_index>**	Response **+BLESSSTOP: <service_index>,<user_id>,<service_handle>** **OK** or **ERROR**
	Parameters **<service_index>** Service index **<transport>** Transport way to start service. 0 LE 1 BR/EDR 2 Dual mode **<user_id>** User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { '0'~'9','a'~'f','A'~'F' }. Max length of it is 32. **<service_handle>** The handle of this service. Dec format.
Read Command **AT+BLESSSTOP?**	Response **OK**
Note	

2.48 AT+BLESLSTART Start advertising

AT+BLESLSTART Start advertising	
Test Command **AT+BLESLSTART=?**	Response **OK**
Write Command **AT+BLESLSTART=<server_index>**	Response **+BLESLSTART: <server_index>,<user_id>** **OK** or **ERROR**
	Parameters <**server_index**> Server index <**user_id**> User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’}. Max length of it is 32.
Read Command **AT+BLESLSTART?**	Response **+BLESLSTART: <server_index>,<user_id>** **OK**
	Parameters See Write Command
Note	**The advertising is started automatically while the server registers successfully by default.**

2.49 AT+BLESLSTOP Stop advertising

AT+BLESLSTOP Stop advertising	
Test Command **AT+BLESLSTOP=?**	Response **OK**
	Parameters See Write Command
Write Command **AT+BLESLSTOP=<server_index>**	Response **+BLESLSTOP: <server_index>,<user_id>** **OK** or **ERROR**
	Parameters <**server_index**> Server index

	<**user_id**> User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’}. Max length of it is 32.
Read Command **AT+BLESLSTOP?**	Response **OK**
Note	

2.50 AT+BLEADV Set Adverting Parameters

AT+BLEADV	**Set Adverting Parameters**
Test Command **AT+BLEADV=?**	Response **OK**
	Parameters See Write Command
Write Command **AT+BLEADV=<server_index>,<scan_rsp>,<include_name>,<include_txpower>,<appearance>,<manufacturer_data>,<service_data>,<service_uuid>**	Response **+BLEADV: <user_id>** **OK** or **ERROR**
	Parameters <**server_index**> Server index <**user_id**> User id of GATT server, or the name of the GATT server. A Hex value string. <**scan_rsp**> include flag parameter or not 0 Not include 1 Include <**include_name**> include bt name 0 Not include 1 Include <**include_txpower**> include Tx power Level 0 Not include 1 Include **<appearance>** set appearance，0~16384. **<manufacturer_data>** set manufacturer, A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’}.Max length of it is 56. <**service_data**> set service_data uuid, A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’}.The length of it should be 0 or 4~32. <**service_uuid**> set complete services uuid, A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’}. The length of it should be 0 or

	4~32.
Note	**AT+BLEADV** will return error when broadcast packet size is over 31 bytes： scan_rsp = 1 3 bytes include_name = 1 characterastic number of bthost name + 2 include_txpower = 1 3 bytes appearance = 0 0 bytes(else will take 4 bytes space) manufacturer_data （Hex value number+1）/2 + 2 service_data （Hex value number+1）/2 + 2 service_uuid （Hex value number+1）/2 + 2 manufacturer_data, service_data and service_uuid won't take any space when Corresponding param is NULL.

2.51 AT+BLESTATUS Inquiry current ble connect status

AT+BLESTATUS	**Inquiry current ble connect status**
Test Command **AT+BLESTATUS=?**	Response **OK**
	Parameters See Write Command
Read Command **AT+BLESTATUS?**	Response If unopened btpower : **+BLESTATUS: <status>** **OK** If btpower opened and connected: **+BLESTATUS: <status>** **+BLESTATUS: <conn_id>,<gatts_type>,<userid>,<addr>** **OK**
	Parameters **<status>** 0 Unopened btpower 1 Btpower opened <**conn_id**> The connection id of current connection <**gatts_type**> 0 custom gatt server 1 FMP server 2 PXP server 3 SPP server <**userid**> User id of GATT server, or the name of the GATT server. A Hex value string

	<addr> Address of the peer device.

2.52 AT+BLEADDR Inquiry current ble address

AT+BLEADDR	**Inquiry current ble address**
Test Command **AT+BLEADDR=?**	Response **OK**
	Parameters See Write Command
Read Command **AT+BLEADDR?**	Response **+BLEADDR: <status>,<addr>** **OK**
	Parameters **<status>** 0 Success 1 Unsuccess **<addr>** Address of current device.

2.53 AT+BLEDISCONN Disconnect BLE connection

AT+BLEDISCONN	**Disconnect BLE connection**
Test Command **AT+BLEDISCONN=?**	Response **OK**
	Parameters See Write Command
Write Command **AT+BLEDISCONN=<conn_id>**	Response **+BLESCON: <op>,<user_id>,<addr>,<conn_id>** **OK** or **ERROR**
	Parameters **<op>** 0 Disconnect 1 Connect **<user_id>** User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’}. Max length of it is 32. **<addr>** Address of the peer device.

	<conn_id> The connection id of current connection.
Read Command **AT+BLEDISCONN?**	Response **OK**
	Parameters See Write Command
Note	When use BLEDISCONN to disconnect server, FMP and PXP ， SPP will have its own URC report (Refer to the BLEFMP and BLEPXP BLESPP disconnection reports).

2.54 AT+BLESIND Send an indication to a client

AT+BLESIND Send an indication to a client	
Write Command **AT+BLESIND=<char_index>,<value>**	Response **+BLESIND: <result>,<user_id>,<conn_id>,<attr_handle>** **OK** or **ERROR**
	Parameters **<char_index>** Characteristic index **<user_id>** User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’ }. Max length of it is 32. **<conn_id>** The connection id of current connection. **<attr_handle>** The handle of the characteristic value. Dec format. **<value>** The value need to be notified. Hex format. **<result>** 0 Success Other un-success

2.55 AT+BLESRSP Send a response to a client’s read or write operation

AT+BLESRSP Send a response to a client’s read or write operation	
Write Command **AT+BLESRSP=<switch>[,<value>]**	Response **+BLESRSP: <result>,<user_id>,<conn_id>,<attr_handle>** **OK** or **ERROR**
	Parameters

	<switch> Read or write 0 Read 1 Write **<user_id>** User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’ }. Max length of it is 32. **<conn_id>** The connection id of current connection. **<attr_handle>** The handle of the characteristic value. Dec format. **<value>** The value need to be notified. Hex format. If <**switch**> is 0, <**value**> is mandatory. **<result>** 0 Success Other Un-success
AT+BLESRSP will be used when read or write URC is reported.	
	URC if there is incoming a read request: **+BLESRREQ: <user_id>,<conn_id>,<trans_id>,<addr>,<attr_handle>,<is_long>,<offset>**
	Parameters **<user_id>** User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’ }. Max length of it is 32. **<conn_id>** The connection id of current connection. **<trans_id>** The id of current transaction.0~65535 **<addr>** Address of the peer device. **<attr_handle>**Handle of attribute. **<is_long>** Tell server that the request is one or several requests. **<offset>** Offset of the request.0~65535
	URC if there is incoming a write request: **+BLESWREQ: <user_id>,<conn_id>,<trans_id>,<addr>,<attr_handle>,<value>,<need_rsp>,<is_prep>,<offset>**
	Parameters **<user_id>** user id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { ‘0’~‘9’,‘a’~‘f’,‘A’~‘F’ }. Max length of it is 32. **<conn_id>** The connection id of current connection. **<trans_id>** The id of current transaction.0~65535 **<addr>** Address of the peer device. **<attr_handle>** Handle of attribute. **<value>** The value need to be write,Hex format

	<need_rsp> Whether client need server's response 1 Yes 0 No **<is_prep>** Whether or not server excute request immediately 0 No 1 Yes **<offset>** Offset of the request.0~65535

2.56 Notify when a connection's status change

Notify when connection's status change	
	Response **+BLESCON: <op>,<user_id>,<addr>,<conn_id>**
	Parameters **<op>** 0 Disconnect 1 Connect **<user_id>** User id of GATT server, or the name of the GATT server. A Hex value string, each char of it should in set { '0'~'9','a'~'f','A'~'F' }. Max length of it is 32. **<addr>** Address of the peer device. **<conn_id>** The connection id of current connection.
Note	

2.57 AT+BLEFMP (De)Register a FMP Service

AT+BLEFMP (De)Register a FMP Service	
Test Command **AT+BLEFMP=?**	Response **+BLEFMP: (0-1)** **OK**
Execute Command **AT+BLEFMP=<op>**	Response **OK** or **ERROR**
	Parameterss <**op**> 0 Deregister 1 Register
Read Command	Response

AT+BLEFMP?	**+BLEFMP: <op>** **OK**
	Parameters See Execute Command

2.58 Notify when a connection's status change comes +BLEFMPCON

Notify when connection's status change comes +BLEFMPCON	
	Response **+BLEFMPCON: <connect_state>,<addr>**
	Parameters **<connect_state>** 0 Disconnect 1 Connect **<addr>** Address of the peer device.

2.59 Notify when a client's write request comes +BLEFMPWREQ

Notify when a client's write request comes +BLEFMPWREQ	
	Response **+BLEFMPWREQ: <addr>,<alert_level>**
	Parameters **<addr>** Address of the peer device. **<alert_level>** Value of Alert Level characteristic. HEX format.

2.60 AT+BLEPXP (De)Register PXP Service

AT+BLEPXP (De)Register PXP Service	
Test Command **AT+BLEPXP=?**	Response **+BLEPXP: (0-1)** **OK**
Execute Command **AT+BLEPXP=<op>**	Response **OK** or **ERROR**
	Parameterss <op>

	0 Deregister 1 Register
Read Command **AT+BLEPXP?**	Response **+BLEPXP: <op>** **OK**
	Parameters See Execute Command

2.61 Notify when a connection comes +BLEPXPCON

Notify when connection's status change comes +BLEPXPCON	
	Response **+BLEPXPCON: <connect_state>,<addr>**
	Parameters **<connect_state>** 0 Disconnect 1 Connect **<addr>** Address of the peer device.

2.62 Notify when a Write request comes +BLEPXPWREQ

Notify when a Write request comes +BLEPXPWREQ	
	Response **+BLEPXPWREQ: <addr>,<alert_level>**
	Parameters **<addr>** Address of the peer device. **<alert_level>** Value of Alert Level characteristic. HEX format.

2.63 Notify when a disconnection alert comes +BLEPXPCON

Notify when a a disconnection alert comes +BLEPXPCON	
	Response **+BLEPXPCON: <addr>,<alert_level>**
	Parameters **<addr>** Address of the peer device. **<alert_level>** Value of Alert Level characteristic. HEX format.

2.64 AT+BLESPP (De)Register a SPP Service

AT+BLESPP (De)Register a SPP Service	
Test Command **AT+BLESPP=?**	Response **+BLESPP: (0-1)** **OK**
Execute Command **AT+BLESPP=<op>**	Response **OK** or **ERROR**
	Parameterss <**op**> 0 Deregister 1 Register
Read Command **AT+BLESPP?**	Response **+BLESPP: <op>** **OK**
	Parameters See Execute Command

2.65 Notify when a connection's status change comes +BLESPPCON

Notify when connection's status change comes +BLESPPCON	
	Response **+BLESPPCON: <connect_state>,<addr>**
	Parameters **<connect_state>** 0 Disconnect 1 Connect **<addr>** Address of the peer device.

2.66 Notify when a client's write request comes +BLESPPWREQ

Notify when a client's write request comes +BLESPPWREQ	
	Response **+BLESPPWREQ: <addr>,<value>**

	Parameters **<addr>** Address of the peer device. **<value>** Value from peer device.

2.67 AT+BLESPPSIND Send an indication to SPP server

AT+BLESPPSIND Send an indication to SPP server	
Write Command **AT+BLESPPSIND=<value>**	Response **OK** or **ERROR**
	Parameters **<value>** The value need to be notified. Hex format,1~40.

3 CME Error Code

The following error message is associated with the Bluetooth operation following format: +CME ERROR: <err>, the specific error code and error message in the following table:

Code	Description
1000	Return fail
1002	Not power on
1003	State not idle
1004	Malloc error
1010	Scan fail
1011	scan return error
1020	Out of scanning count
1021	Out of profile id count
1025	Out of pairing count
1026	Bond error
1027	Device has Bonded
1030	Debond error
1031	Get device info error
1032	Service refresh error
1033	Profile connect error
1034	HF attach error
1040	OPP handle error
1041	OPP send error
1042	OPP received path error
1043	SD card not exist
1044	OPP file path error
1045	OPP send error by server
1046	Get index by profile error
1047	Connect not support
1048	Disconnect not support
1049	Active or address error
1050	Only connect one device
1051	Out of max connection
1055	SPP is not connect
1056	Spp server isn’t work at send mode
1057	Input data length beyond
1058	SPP port is not create
1060	Pls connect A2DP first

1061	Connected device exceed max
1099	BTAUD attach error
1997	GATT server write error
1998	GATT server read error
1999	GATT server connect error
2000	GATT server register error
2001	GATT server deregister error
2002	GATT no server error
2003	GATT add service error
2004	GATT remove service error
2005	GATT add characteristic error
2006	GATT start service error
2007	GATT stop service error
2008	GATT start/stop advertising error
2009	GATT add descriptor error
2010	GATT server exceed the max number

4 Examples

There are some examples to explain how to use these commands.

In the "Grammar" columns of following tables, inputs of AT commands are in black, module return values are in blue.

4.1 Accept request from other BT device

Command	Description
AT+BTPOWER=1 OK	Power on BT radio
+BTPAIRING: "PC-NS130100361",34:c7:31:aa:37:5b,763191	Incoming digital key request from other BT device
AT+BTPAIR=1,1 OK +BTPAIR: 1,"PC-NS130100361",34:c7:31:aa:37:5b	Accept pairing request, and paired successfully
+BTPAIRING: "Jabra BT160",00:16:8f:0d:65:82	Incoming passkey request from other BT device
AT+BTPAIR=2,0000 OK +BTPAIR: 2,"LBH505",50:5b:0b:0a:10:32	Accept pairing request, and paired successfully.Default passkey of other BT device is 0000.If not, please change this value according to other device's passkey.

4.2 Send pairing request to other BT device

Command	Description
AT+BTPOWER=1 OK	Power on BT radio
AT+BTSCAN=1,20 OK +BTSCAN: 0,1,"PC-NS130100361",34:c7:31:aa:37:5b,-34 +BTSCAN: 0,2,"ADMIN-9A6E040AC",68:5d:43:ec:fe:72,-44 +BTSCAN: 0,3,"LIB-PC",c8:f7:33:43:48:e6,-54	Inquiring surrounding BT device

+BTSCAN: 0,4,"MK-FUJIANJUN",88:53:2e:e8:9d:0f,-33 +BTSCAN: 0,5,"MTKBTDEVICE",45:8c:96:3e:66:01,-56 +BTSCAN: 0,6,"MK-ZHANZHIMIN",00:1a:7d:da:71:10,-67 +BTSCAN: 0,7,"Jabra BT160",00:16:8f:0d:65:82,-55 +BTSCAN: 1	
AT+BTPAIR=0,6 OK	Try to pair the sixth BT device in the view list
+BTPAIRING: "MK-ZHANZHIMIN",00:1a:7d:da:71:10,763191 AT+BTPAIR=1,1 OK +BTPAIR: 1,"MK-ZHANZHIMIN",00:1a:7d:da:71:10	Answer to the pairing request in digital key mode
AT+BTPAIR=0,7 OK	Try to pair the seventh BT device in the view list
+BTPAIRING: "Jabra BT160",00:16:8f:0d:65:82 AT+BTPAIR=2,0000 OK +BTPAIR: 2,"Jabra BT160",00:16:8f:0d:65:82	Answer to the pairing request in passkey mode

4.3 Get the profile provided by paired device

Command	Description
	Configure based on example 4.2
AT+BTGETPROF=1 +BTGETPROF: 1,"A2DP(Source)" +BTGETPROF: 2,"HFP(AG)" +BTGETPROF: 8,"AVRCP(Target)" +BTGETPROF: 3,"A2DP" +BTGETPROF: 4,"SPP" +BTGETPROF: 6,"HFP"	Get the profile of first paired device in list

+BTGETPROF: 5,"HSP" OK	

4.4 Connect service

Command	Description
	Get Profile based on example 4.3
AT+BTCONNECT=1,2 OK +BTCONNECT: 1,"MK-ZHANZHIMIN",00:1a:7d:da:71:10,"HFP(AG)"	Connect with the second profile service of first paired device, "HFP(AG)"

4.5 Accept file from paired device

Command	Description
	Pairing device based on example 4.2
+BTOPPPUSHING: "MK-ZHANZHIMIN","link.txt"	Incoming opp pushing service from paired device
AT+BTOPPACPT=1 OK +BTOPPPUSH: 1	Accept file(stored in internal memery card by default,input "AT+BTOPPACPT=1,1" if want it stored in external memory

4.6 Send file to other paired BT device

Command	Description
	Pairing device based on example 4.2
AT+BTOPPPUSH=1,c:\User\BtReceived\link.txt OK +BTOPPPUSH: 1	Sending file and waiting for response

4.7 Create SPP's link as a client

Command	Description

	Suppose this device's ID is 12:34:56:78:90:12,name is IT;Another ID is 34:c7:31:aa:37:5b,name is ME.they make pair successfully.
AT+BTCONNECT=1,4 OK +BTCONNECT: 1,"IT",12:34:56:78:90:12,"SPP"	Try to build a SPP's connection to server. If successfully, output these URC.

4.8 SPP's link be create as a server

Command	Description
	Suppose this device's ID is 12:34:56:78:90:12, name is IT; The other ID is 34:c7:31:aa:37:5b, name is ME.they make pair successfully.
+BTCONNECTING: "34:c7:31:aa:37:5b","SPP" AT+BTACPT=1 OK +BTCONNECT: 1,"ME",34:c7:31:aa:37:5b,"SPP"	Receive a request from client which build a connection. Accept it. Build success.

4.9 Configurate SPP

Command	Description
	Get Profile based on example 4.3. Suppose this device's ID is 12:34:56:78:90:12, and name is IT;The other ID is 34:c7:31:aa:37:5b, and name is ME.This module has had a server-type link of SPP.
AT+BTSPPCFG? +BTSPPCFG: S,1,0 OK AT OK AT OK AT+BTSPPCFG="MC",1 OK	 There is a link.It's a server; Connection's ID is 1; It's not allowed to send data to client. If there is a request from another device which tries to build a connection, no URC will be reported. Because this module disable multi-connection function. Enable multi-connection function.

AT+BTSPPCFG="MC",2 +BTSPPCFG: MC,1 OK +BTCONNECTING: "0c:c5:95:09:62:60","SPP" AT+BTACPT=1 OK +BTCONNECT: 1,"THIRD",0c:c5:95:09:62:60,"SPP" +BTSPPDATA: 2,15,SIMCOMSPPFORAPP AT OK AT+BTSPPCFG? +BTSPPCFG: S,1,0 +BTSPPCFG: S,2,1 OK	Inquire whether the multi-connection is enabled. Enable. There is a request that tries to build a SPP's connection. Build connection successfully. Receive the message of switching mode to APP mode from the second client's link. Allow to send data to second client's link.

4.10 Send data as a SPP's client

A SPP connection has two modules.One is client,and the other is server. Let us see the demo with client module.

Command	**Description**
	Based on example 4.7, as a client.
AT+BTSPPCFG? +BTSPPCFG: C,1 OK AT+BTSPPSEND >AT+CREG?□ SEND OK +BTSPPDATA: 19,1,A +BTSPPDATA: 19,3,T+C +BTSPPDATA: 19,25,REG? +CREG: 0,0 OK AT+BTSPPSEND=10	 There is a link, client-type, and allowed to send data to the server. If the client sends AT command to the server, this command and its response will output to client. "AT+CREG?" are input characters. "+CREG: 0,0" and "OK" are responses. If the multi-connection function is disabled,

>1234567890□ SEND OK	we don't need to input connection's ID. Input data(1234567890) and press Ctrl+Z keys, the data will be sent.

4.11 As a SPP's server worked in AT mode

SPP's connection as a server has two mode.One is AT mode.In this mode, we can't use AT+BTSPPSEND/BTSPPGET commands to send data to the client or get data from the client. We can only receive data from the client.

Command	Description
	Based on example 4.8, as a server.
AT+BTSPPCFG? +BTSPPCFG: S,1,0 OK	There is a link.Server-type; connection's ID is 1; It's not allowed to send data to the client.
AT+BTSPPSEND=10 ERROR	Fail to send.
AT+BTSPPSEND ERROR	Fail to send.

4.12 As a SPP's server worked in APP mode and multi-connection

Another SPP's link mode as a server is the APP mode. In this mode,we can execute AT+BTSPPSEND and AT+BTSPPGET commands.

Command	Description
	Based on example 4.7, as a server.
+BTSPPDATA: 1,15,SIMCOMSPPFORAPP AT OK AT OK AT+BTSPPCFG? +BTSPPCFG: S,1,1 OK AT+BTSPPSEND >12345□ SEND OK AT+BTDISCONN=1 OK	Receive the specified data package from the first client's link which means switching the mode to APP mode(This data package must be the first package recieved).After excuting AT+BTSPPCFD="",client will enter APP mode when sending data package without specified strings. Allow to send data to the client. Send successefully.

+BTDISCONN: "SIM800H",34:c7:31:aa:37:5b,"SPP" AT+BTSPPGET=1 OK +BTCONNECTING: "34:c7:31:aa:37:5b","SPP" AT+BTACPT=1 OK +BTCONNECT: 1,"SIM800H",34:c7:31:aa:37:5b,"SPP" +BTSPPMAN: 1 AT OK AT+BTSPPGET=2,1 +BTSPPGET: 1,15 OK AT+BTSPPGET=3,1,15 +BTSPPGET: 1,15,SIMCOMSPPFORAPP OK AT+BTSPPSEND > 1234567890□ SEND OK	Disconnect this link of client. Switch to manual mode. Recieve the connecting request from the client. Build link successefully. Receive the data from the client whose connection's ID is 1. Connection's ID is 1, and the data length is 15. Get data, length is 15(This data package means switching the mode to APP mode). Send data to the client. Send successefully.
AT+BTSPPGET=? +BTSPPGET: (0-3),(1-6),(1-1024),1 OK	

4.13 Sync phonebook from remote by BT

Command	Description
	Based on example 4.2
AT+BTGETPROF=1 +BTGETPROF: 10,"PBAP" +BTGETPROF: 1,"A2DP(Source)" +BTGETPROF: 2,"HFP(AG)" +BTGETPROF: 8,"AVRCP(Target)" OK	Get the profile of first paired device in list

AT+BTCONNECT=1,10 OK +BTCONNECT: 1,"LG-P705",00:aa:70:23:7d:06,"PBAP(C)"	Connect server Report automatically once ready
AT+BTPBSYNC=0,1,0 OK +BTPBSYNC: 0,0,53786	Sync phonebook Sync succeed. File size is 53786 bytes.

4.14 Find name or number from remote by BT

Command	Description
	Based on example 4.2
AT+BTGETPROF=1 +BTGETPROF: 10,"PBAP" +BTGETPROF: 1,"A2DP(Source)" +BTGETPROF: 2,"HFP(AG)" +BTGETPROF: 8,"AVRCP(Target)" OK	Get the profile of first paired device in list
AT+BTCONNECT=1,10 OK +BTCONNECT: 1,"LG-P705",00:aa:70:23:7d:06,"PBAP(C)"	Connect server Report automatically once ready
AT+BTPBF=1,"135",1 OK +BTPBF: 1,5 +BTPBF: 1,1,00310033003500380035003800380037003700370035 +BTPBF: 1,2,5170621056FD +BTPBF: 1,3,521800206587660E +BTPBF: 1,4,52186021	Find name whose number contain "135". Find succeed. Five names found.

+BTPBF: 1,5,5362592A592A	
AT+BTPBF=0,"0063",1 OK +BTPBF: 0,1 +BTPBF: 0,1,1 +BTPBF: 0,1,1,***********,1	Find number which owner’s name contain char "c" (format with usc2 value is “0063”). Find succeed. One phonebook record found. First phonebook record contain one number

4.15 Play music and so on by AVRCP

Command	Description
	Based on example 4.2
AT+BTGETPROF=1 +BTGETPROF: 1,"A2DP(Source)" +BTGETPROF: 2,"HFP(AG)" +BTGETPROF: 8,"AVRCP(Target)" OK	Get the profile of first paired device in list
AT+BTCONNECT=1,1 OK +BTCONNECT: 1,"Lenovo A780",d8:71:57:2b:02:66,"A2DP" +BTCONNECT: 2,"Lenovo A780",d8:71:57:2b:02:66,"AVRCP" +BTCONNECT: 3,"Lenovo A780",d8:71:57:2b:02:66,"HFP(AG)"	Connect with the first profile service of first paired device, "A2DP", For the service of "AVRCP" depends on the "A2DP". After connected with “A2DP” successfully, the modem will connect to the sevice of "AVRCP" automatically. Report automatically once ready.
AT+BTAVRCOP=1 OK AT+BTAVRCOP=2 OK AT+BTAVRCOP=1 OK AT+BTAVRCOP=3 OK	Play music The sound can be heard form the modem Pause music The music will be paused Play music again The music will be palyed Play the next song The next song will be palyed

AT+BTAVRCOP=4 OK	Play the back song The back song will be palyed
AT+BTAVRCOP=5 OK	Increase the volume The volume of the music will be increased
AT+BTAVRCOP=6 OK	Decrease the volume The volume of the music will be Decreased
AT+BTAVRCOP=0 OK	Stop music The music will be stoped

4.16 Add phonebook records to ME or SM phonebook from VCARD file

Command	Description
	Based on example 4.13
AT+BTPBSYNC=1,1,0,0,1 OK +BTPBSYNC: 1,0,214,67	Sync file "c:\user\bt\remotePb1.txt" to SM phonebook with overwrite mode Sync finished. 214 phonebook records add succeed and 67 records failed.
AT+CPBR=1,250 +CPBR: 1,"",129,"Me" … OK	Read phonebook records.

4.17 Set BT pairing mode

Command	Description
AT+BTPOWER=1 OK AT+BTPAIRCFG=1 OK	Power on BT radio Set paring mode is PIN-Code inputted by manual（mode=1）, and the default PIN-Code value is 0000, if you want to set other PIN-Code, follow it：AT+BTPAIRCFG=1,<pin_code>
	BT reboot
AT+BTSCAN=1 OK	Inquiring surrounding BT device and pair, input PIN-Code by opposite side, the default

+BTSCAN: 0,1,"XT615 ",00:11:94:cb:20:d2,-34 +BTSCAN: 0,2,"LIB-PC",c8:f7:33:43:48:e6,-45 AT+BTPAIR=0,1 OK +BTSCAN: 2 +BTPAIR: 1,"XT615 ",00:11:94:cb:20:d2	value is 0000
AT+BTPAIRCFG=2 OK	Set pairing mode is random PIN-Code(mode = 2). (mode = 0, reference 4.2 section)
	BT reboot
AT+BTSCAN=1 OK +BTSCAN: 0,1,"XT615 ",00:11:94:cb:20:d2,-44 +BTSCAN: 0,2,"MK-ZHANZHIMIN",00:1a:7d:da:71:10,-55 AT+BTPAIR=0,1 OK +BTSCAN: 2 +BTPAIR: 1,"XT615 ",00:11:94:cb:20:d2	Inquiring surrounding BT device and pair, and wait to confirm pairing request by opposite side.

4.18 Inquiry current ble address

Command	Description
AT+BTPOWER=1 OK	Power on BT radio
AT+BLEADDR? +BLEADDR: 0,d4:d9:f9:30:88:33 OK	Inquiry current ble address。

4.19 Set Adverting Parameters

Command	Description
AT+BTPOWER=1	Power on BT radio

OK	
AT+BLESREG +BLESREG: 1,ABCDEFF0 OK	Register GATT Server。
AT+BLEADV=1,0,0,0,0,"","","" +BLEADV: ABCDEFF0 OK AT+BLEADV=1,1,1,0,25,"4c00","02291234","2902" +BLEADV: ABCDEFF0 OK	Set Adverting Parameters。 Gradually add parameters to see the changes through the APP。

4.20 Setup GATT server

Command	Description
AT+BTPOWER=1 OK	Power on BT radio
AT+BLESREG +BLESREG: 1,ABCDEFF0 OK	Register GATT Server.
AT+BLESSAD=1,”123456”,15,1,1 +BLESAD: 1,ABCDEFF0,123456,1,1,256 OK	Add a service.
AT+BLESSC=1,”ABCDEF”,1, 10,17 +BLESSC: 1,ABCDEFF0,256,ABCDEF,1,258 OK	Add a R/W characteristic.
AT+BLESSC=1,”ABCDEF”,1, 16,17 +BLESSC: 2,ABCDEFF0,256,ABCDEF,1,260 OK	Add a Notify characteristic.
AT+BLESSD=1,"0229",1,0 +BLESSD: 1,ABCDEFF0,256,0229,1,261 OK	Add a descriptor.

AT+BLESSSTART=1,0 +BLESSSTART: 1,ABCDEFF0,256 OK	Setup service.
AT+BLESLSTART=1 +BLESLSTART: 1,ABCDEFF0 OK	Start advertising.

4.21 Data transmission between module and client

Command	Description
	Start the GATT service as shown in example 4.20.
+BLESCON: 1,ABCDEFF0,7a:16:fc:60:72:40,1	APP connect with module.
+BLESRREQ: ABCDEFF0,1,99,7a:16:fc:60:72:40,258,0,0	APP read data.
AT+BLESRSP=0,A1B2 +BLESRSP: 0,ABCDEFF0,1,258 OK	Answer with A1B2。
+BLESWREQ: ABCDEFF0,1,100,7a:16:fc:60:72:40,258,ABCD,1,0,0	Write with ABCD。
AT+BLESRSP=1 +BLESRSP: 0,ABCDEFF0,1,258 OK	Answer the write request。
AT+BLESIND=2,”9876” +BLESIND: 0,ABCDEFF0,1,260 OK	Module send 9876 to Notify characterastic。
+BLESCON: 0,ABCDEFF0,7a:16:fc:60:72:40,1	APP disconnect wite module.

4.22 Setup FMP server

Command	Description
AT+BTPOWER=1 OK	Power on BT radio.
AT+BLEFMP=1	Setup FMP server.

OK	
+BLEFMPCON: 1,69:e9:06:60:7a:e7	APP connect with module.
+BLEFMPWREQ: 69:e9:06:60:7a:e7,87	APP write data.
+BLEFMPCON: 0,69:e9:06:60:7a:e7	APP disconnect wite module.

4.23 Setup PXP server

Command	Description
AT+BTPOWER=1 OK	Power on BT radio.
AT+BLEPXP=1 OK	Setup PXP server.
+BLEPXPCON: 1,6f:53:17:18:56:15	APP connect with module.
+BLEPXPWREQ: 6f:53:17:18:56:15,78	APP write data.
+BLEPXPCON: 6f:53:17:18:56:15,87	APP disconnect wite module.

4.24 Setup SPP server

Command	Description
AT+BTPOWER=1 OK	Power on BT radio.
AT+BLESPP=1 OK	Setup SPP server.
+BLESPPCON: 1,6f:53:17:18:56:15	APP connect with module.
+BLESPPWREQ: 6f:53:17:18:56:15,78	Module sent data to APP.
AT+BLESPPSIND=”ABCD” OK	APP write data.
+BLESPPCON: 0,66:ee:48:40:e0:64	APP disconnect wite module.

4.25 Inquiry current ble status

	Setup GATT， FMP,PXP,SPP. APP connect with module.

AT+BLESTATUS? +BLESTATUS: 1 +BLESTATUS: 1,0,ABCDEFF0,66:ee:48:40:e0:64 +BLESTATUS: 2,1,ABCDEFFA,66:ee:48:40:e0:64 +BLESTATUS: 3,2,ABCDEFFB,66:ee:48:40:e0:64 +BLESTATUS: 4,3,ABCDEFFC,66:ee:48:40:e0:64 OK	Inquiry current ble status.

4.26 Module disconnect with APP

	Inquiry current ble status first.
AT+BLEDISCONN=1 +BLESCON: 0,ABCDEFF0,49:bb:c7:48:4d:87,1 OK AT+BLEDISCONN=2 +BLEFMPCON: 0,49:bb:c7:48:4d:87 OK	Module disconnect with APP.

4.27 Module disconnect Start or stop advertising

Command	Description
AT+BTPOWER=1 OK	Power on BT radio.
AT+BLESREG +BLESREG: 1,ABCDEFF0 OK	Register GATT Server.
AT+BLESLSTART=1 +BLESLSTART: 1,ABCDEFF0 OK	Start advertising.
AT+BLESLSTOP=1 +BLESLSTOP: 1,ABCDEFF0 OK	Stop advertising.

5 Differences between bluetooth version and standard Version

Note: In this chapter, SIM800 BT indicates SIM800 series BT version, SIM800 indicates SIM800 series standard version. Differences among SIM800 series standard version, please refer to chapter 21 for details in doc "SIM800 Series AT Command Manual".

5.1 ATD<str>

SIM800 BT does not support finding number by name.

5.2 AT+CPBF

SIM800 BT	SIM800
Max length of <findtext> is always 40 bytes.	Max length of <findtext> depends on AT+CSCS
Results will order by phonebook index when select "SM" or "ME" phonebook, from small to large.	Results will order by the order user inputs phonebooks.
<findtext> must match <text> from the leftmost side, when select "SM" or "ME" phonebook	No this limit
Difference	There are multi difference of AT+CPBF between SIM800 BT and SIM800.

5.3 AT+CPBFEX

MTK MMI version can support this command and modem version is the opposite. That is to say, MTK6260 and MTK 6260A platform without BT version and MTK6261A platform cannot support this command.

5.4 AT+CMUX

SIM800 BT does not support MUX function.

5.5 AT+CNUM

SIM800 BT	SIM800
+CNUM: [<alpha>],<number>,<type>,,<service>	+CNUM: <alpha>,<number>,<type>,<speed>,<service>

Difference	**<alpha> of SIM800 BT does not display if length of <alpha> is 0.** **SIM800 BT does not support <speed> field and left blank.**

5.6 AT+CMGS

SIM800 BT does not support sending message by phonebook index or name.

5.7 AT+CMSS

SIM800 BT does not support sending message from storage.

5.8 AT+CPMS

SIM800 BT	**SIM800**
AT+CPMS=? **+CPMS: ("SM","ME","MT"),("SM","ME","MT"),("SM","ME","MT")** **OK**	**AT+CPMS=?** **+CPMS: ("SM","ME","SM_P","ME_P","MT"),("SM","ME","SM_P","ME_P","MT"),("SM","ME","SM_P","ME_P","MT")** **OK**
Difference	**SIM800 BT supports three modes: "SM","ME","MT".** **SIM800 supports "SM","ME","SM_P","ME_P","MT" modes.**

5.9 AT+CHFA

SIM800 BT	**SIM800**
AT+CHFA=? **+CHFA: (0=NORMAL_AUDIO, 1=AUX_AUDIO, 2=HANDFREE_AUDIO, 3=AUX_HANDFREE_AUDIO, 4=PCM_AUDIO,5=BT_CHANNEL)** **OK**	**AT+CHFA=?** **+CHFA: (0=NORMAL_AUDIO, 1=AUX_AUDIO, 2=HANDFREE_AUDIO, 3=AUX_HANDFREE_AUDIO, 4=PCM_AUDIO)** **OK**
Difference	**Value of parameter <n> has BT audio channel in SIM800 BT.** **BT channel can be set when BT link is established and module acts as mobile phone. After switch to BT channel, local sound can be transferred to BT earphone. If BT link is disconnected, audio channel will restore to the original channel and URC +CHFA: <n> is reported. Because the audio service is always on after switch to BT channel, consumption current is bigger than normal.**

5.10 TTS function

SIM800 BT which module memory is 32M does not support TTS function.

Appendix

A. Reference

ID	Document	Remark
[1]	SIM800 Series_AT Command Manual	

B. Profile

Profile	Introduction
SPP	Abbreviation of Serial Port Profile,to implement BT serial port function.Moduel an transimit data to connected BT device throuth AT+BTSPPSEND after successfully applying this profile.The module will receive data report +BTSPPDATA in automatic mode,and +BTSPPMAN in mamual mode.
OPP	Abbreviation of OPP Object Push Profile,to implement pushing BT object.This unction is used between the two paired BT devices,AT+BTOPPPUSH to push file, AT+OPPACPT to receive the pushed file.
HFP/HSP	Abbreviation of Handsfree Profile/Headset Profile, i.e. BT earphone function. HFP is the enhanced version of HSP,so even if the other BT device just supports HSP,SIM800H still can connect the BT device with HFP.Module's call voice would be displayed from BT earphone after this profile being connected.When the module play a role as smart phone,BT earphone could control the call operation(e.g.hang up,answer,redial) .
A2DP	Abbreviation of Advanced Audio Distribution Profile,which is advanced rotocol for audio frequency distribution.Earphone will activate AVRCP connection after the profile being connected.It is mainly used to for BT earphone to transmit Hi-Q audio frequency.If be suffixed with source,it means this device is audio frequency source,i.e. paly a role as smartphone.
AVRCP	Abbreviation of Audio Video Remote Control Profile,is AV remote control protocol.This profile depends on A2DP and only could be connected after the A2DP connection is established.It is mainly used for BT earphone to control the edia function of smartphone.If be suffixed with target,it means this device is controlling target,i.e.paly a role as smart phone.
HFP(AG)	This profile is HFP,i.e. paly a role as BT earphone.After the module connected with smartphone,the call voice of smartphone could be displayed by the module's audil channel.Also the call operation of smartphone can be controlled by those commands such as AT+BTATD， AT+BTATH， AT+BTATA.
HFG	This profile is HFP,but plays a role as smartphone at this moment.After the module connected with smartphone,there will display such information indicates profile being connected successfully.If the module plays a role of earphone,then the information displayed after connection will be HFP(AG).

PBAP	Phone Book Access Profile (PBAP) is a profile that allows exchange of Phone Book Objects between devices.
BLEFMP	Find Me Profile (FMP) , The mobile terminal can send data to the module to identify the current phone calls, SMS, alarm clock or find module location.
BLEPXP	PXP Profile, Support all the functions of FMP, you can set the URC report after the disconnection.
BLESPP	To implement BLE serial port function.

C. Glossary and Abbreviation

Glossary	Discription
EVB	Evaluation Board
BT	Blue tooth
PROFILE	Bluetooth function protocol
SPP	Serial Port Profile
OPP	OPP Object Push Profile
A2DP	Advanced Audio Distribution Profile
AVRCP	Audio Video Remote Control Profile
HSP	BT handset protocol
HFP	HandFree application protocol
URC	Unsolicited Result Code
TE	Terminal Equipment
TA	Terminal Adapter
DTE	Data Terminal Equipment
DCE	Data Communication Equipment
ME	Mobile Equipment
MS	Mobile station
PBAP	Phone Book Access Profile

Contact us:

Shanghai SIMCom Wireless Solutions Co.,Ltd.

Address: Building A, SIM Technology Building, No. 633, Jinzhong Road, Shanghai, P. R. China 200335
Tel: +86 21 3252 3300
Fax: +86 21 3252 3020
URL: www.simcomm2m.com