LY50A02

BLE Dongle 用户手册

Bluetooth Series
BLE Mode

深圳市科名科技有限公司

Shenzhen Keming Technology

- 蓝牙版本: Bluetooth Specification V5.0 BLE
- USB 串口蓝牙设备
- AT 指令集
- 使用简单
- 主机功能，连接从机设备
- 支持 BLE 5.0 BEL 4.2, BLE 4.1, BLE 4.0
- 支持 16 bits UUID
- 支持 128 bits UUID
- 每包最大有效数据: 244 字节
- 最大支持 10K 字节每秒的数据传输(测试条件：在 115200 波特率的情况每包 10000 字节，需要从机也支持这个速度)
- 速度: 6-10K 字节/秒
- 可通过 MAC 地址连接 BLE 从机设备
- 通过名字连接 BLE 从机设备
- 连接加密 BLE 设备
- 支持工厂产线产品测试
- 读取 RSSI

目录

产品介绍：

本产品利用 TI 公司 蓝牙芯片做数据收发，本适配器集成 USB 转串口及蓝牙控制器，在 PC 端，蓝牙适配器为串口设备，所有操作都通过串口实现，支持AT 指令设置参数，支持16位UUID,支持128位 UUID。适用各种标准 BLE 蓝牙设备。向下兼容 BLE4.2 ,BLE4.1,BLE4.0

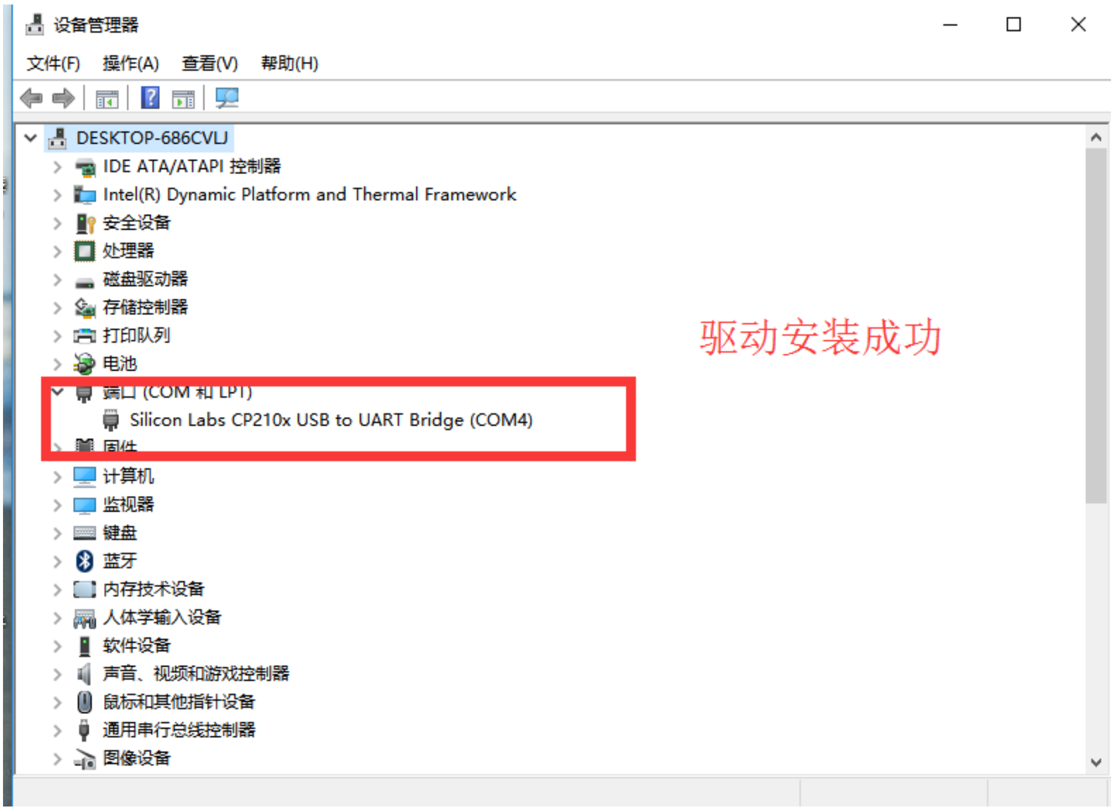
AT 指令列表

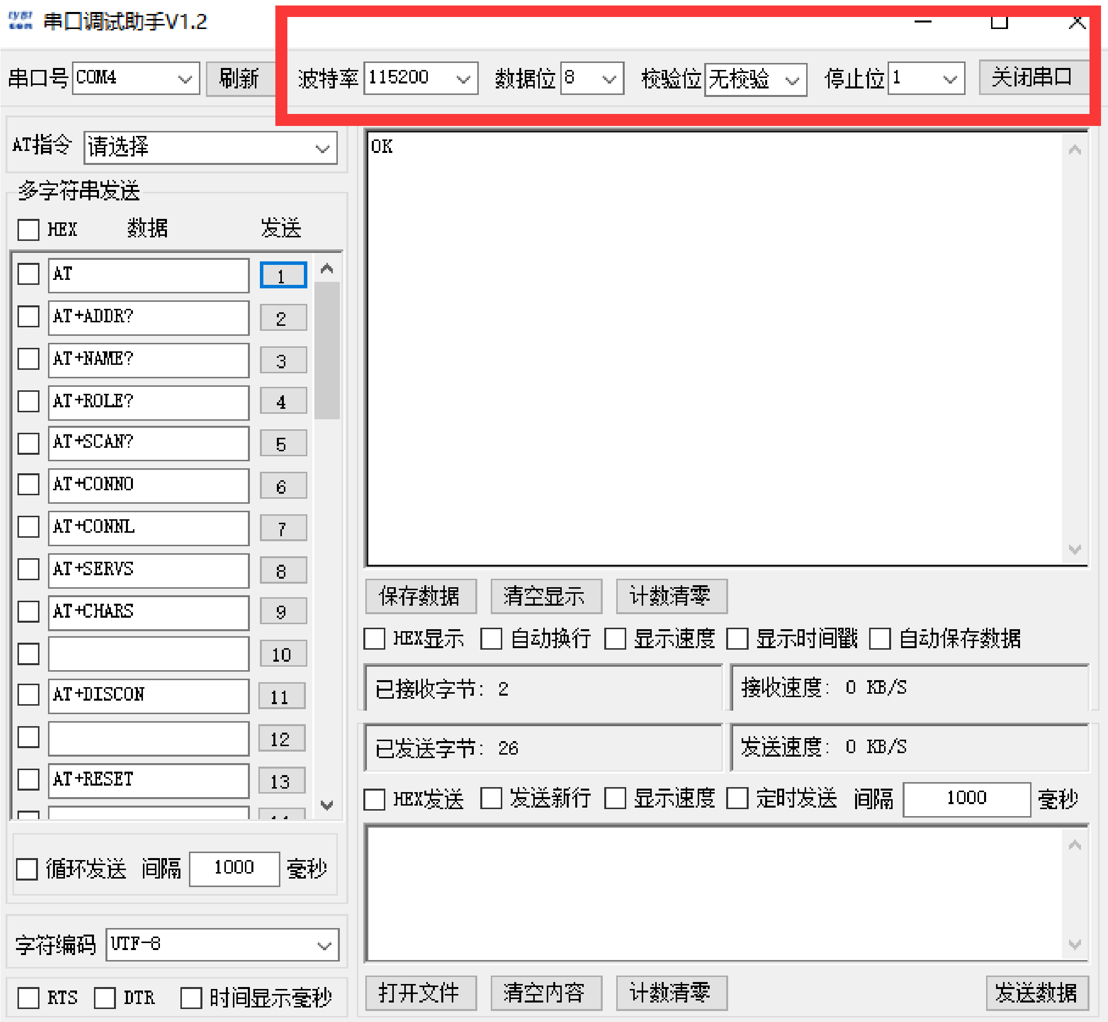
编号	AT 指令	功能描述	默认值
1	AT	测试指令，测试串口及适配器工作状态	
2	AT+ADDR?	查询设备 MAC 地址	
3	AT+VERS?	查询设备固件版本	
4	AT+SCAN?	搜索指令	
5	AT+CONN	连接搜索到指定设备	
6	AT+CONM	通过名称地址连接设备	
7	AT+CON	通过 MAC 地址连接设备	
8	AT+CONNL	连接最后一次连接过的设备	
9	AT+DISCON	断开连接	
10	AT+SERV	设置 service UUID	
11	AT+SERVS	搜索 service UUID	
12	AT+CHARS	刷新 characteristic UUID	
13	AT+CHAR?	查询已发现的特征值 UUID	
14	AT+CHRX	设置接收特征值 UUID	
15	AT+CHTX	设置发送特征值 UUID	
16	AT+SHCH	是否显示服务特征值指令	
17	AT+SDUR	设置搜索时间	
18	AT+RENEW	恢复出厂设置	
19	AT+RESET	重启	
20	AT+RSSI?	查询信号 RSSI 值	
21	AT+RADD?	查询最后一次连接成功的设备 MAC 地址	
22	AT+BAUD	设置串口（UART）波特率	115200BPS
23	AT+PARI	设置串口（UART）校验位	
24	AT+PAIR	设置配对模式	
25	AT+PASS	设置配对密码	
26	AT+RDCH	读取指定特征值的值	
27	AT+TXPW	设置发射功率	
28	AT+GAIN	设置接收增益	
29	AT+AUCO	设置自动连接	
30	AT+STATE?	查询连接状态	
31	AT+FRSSI	设置搜索过滤的信号强度	
32	AT+NICH	打开(使能)指定特征值服务属性	
33	AT+NOCH	关闭指定特征值服务属性	
34	AT+SCTRL	设置、解析及过滤广播包字段指令	
35	AT+RIPRD	设置 RSSI 读取间隔	

.

驱动安装

适配器串口芯片采用 Silicon LABS 公司的 CP2102 桥接芯片，
驱动下载地址为：https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers
根据您的电脑的操作系统自行选择对应的驱动。
Windows 8，Windows 10 一般情况下都可以自动加载驱动，用户使用时可以直接把适配器插到电脑上，到设备管理器界面查看是否可以正常加载驱动，如果不能正常加载你驱动，请手动下载并安装。驱动安装好后，设备管理器可以查看到对应的 COM 口，如下图所示。





波特率：115200

数据位：8 位

校验位：无

停止位：1

通过串口助手发送

AT Command

1. AT 测试指令

如果适配器工作正常，发送 AT 会返回 “OK”

AT	
执行指令 **AT**	响应返回 **OK**

示例：

```
AT

OK
```

2. AT+ADDR? 查询适配器 MAC 地址

AT+ADDR?	
查询指令 **AT+ADDR?**	响应 **OK+Get:<MAC>**

参数

<MAC>	适配器的 MAC 地址

示例：

```
AT+ADDR?

OK+Get:0x0C61CFC6F664
```

适配器的 MAC 地址为： **0xC61CFC6F664**

3. AT+VERS? 查询软件版本

AT+VERS?	
查询指令	响应
AT+VERS?	**OK+Get:<VERSION>**

参数

<VERSION>	适配器的软件版本

示例:

AT+VERS?

OK+Get:LY50A02-V08

适配器的软件版本为: LY50A02-V08

4. AT+SCAN?　搜索 BLE 从机设备

AT +SCAN?	
执行指令	响应
AT+SCAN?	**Scanning...** **<INDEX>:<MAC><RSSI><NAME>** **...** **Devices Found:<QUANTITY>**

参数

<INDEX>	已经搜索到的 BLE 设备索引编号
<MAC>	经搜索到的 BLE 设备 MAC 地址
<RSSI>	经搜索到的 BLE 设备的 RSSI 值
<NAME>	经搜索到的 BLE 设备的名称
<QUANTITY>	经搜索到的 BLE 设备的总数量

示例:

```
AT+SCAN?
Scanning...
0: 0x1CCAE32FC8AF, -60, KM-BLE
1: 0x1CCAE32FC512, -72, KM-BLE
2: 0x1CCAE328BE93, -68, KM-BLE
3: 0x1CCAE325E0CB, -51, KM-BLE
4: 0x1CCAE326226D, -69, KM-BLE
5: 0x1CCAE325E051, -60, KM-BLE
6: 0x2CAB332D37A5, -85, KM-BLE
7: 0x2CAB332D52F5, -83, KM-BLE
8: 0x2CAB33355259, -76, KM-BLE
9: 0x2CAB332D4F99, -83
Devices Found: 10
```

5. AT+CONN　通过索引连接 BLE 设备

AT +CONN	
执行指令	响应
AT+CONN<INDEX>	**Connecting** **OK+CONN:<MAC>** **<NUM>: <UUID>, <PROP1>, <PROP2>,...<PROPn>** **...** **Chars Found: <QUANTITY>**

参数

<INDEX>　已搜索到的 BLE 蓝牙设备的索引

<MAC>　已连接 BLE 设备的 MAC 地址

<NUM>　已连接的 BLE 蓝牙设备的 UUID 索引

<UUID>　已连接的 BLE 蓝牙设备的特征值 UUID

<PROP(n)>　已连接的 BLE 蓝牙设备的特征值 UUID 的属性

<QUANTITY>　已连接的 BLE 蓝牙设备的特征值 UUID 的数量

适配器连接 BLE 设备后，默认会列出所有特征值 UUID.

示例

```
AT+SCAN?
Scanning...
0: 0x1CCAE32FC8AF, -60, KM-BLE
1: 0x1CCAE32FC512, -72, KM-BLE
2: 0x1CCAE328BE93, -68, KM-BLE
3: 0x1CCAE325E0CB, -51, KM-BLE
4: 0x1CCAE326226D, -69, KM-BLE
5: 0x1CCAE325E051, -60, KM-BLE
6: 0x2CAB332D37A5, -85, KM-BLE
7: 0x2CAB332D52F5, -83, KM-BLE
8: 0x2CAB33355259, -76, KM-BLE
9: 0x2CAB332D4F99, -83
```

```
Devices Found: 10
AT+CONN0
Connecting
OK+CONN:0x1CCAE32FC8AF
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify
Chars Found: 5
```

6. AT+CONM 通过名称连接设备

AT+CONM	
执行指令 **AT+CONM<NAME>**	响应 **Scanning...** **Connecting** **OK+CONN:<MAC>** **<NUM>: <UUID>,<PROP1>, <PROP2>,...<PROPn>** **...** **Chars Found: <QUANTITY>**

参数

<NAME> BLE 设备名称

<MAC> 已连接 BLE 设备的 MAC 地址

<NUM> 已连接的 BLE 蓝牙设备的 UUID 索引

<UUID> 已连接的 BLE 蓝牙设备的特征值 UUID

<PROP(n)> 已连接的 BLE 蓝牙设备的特征值 UUID 的属性

<QUANTITY>已连接的 BLE 蓝牙设备的特征值 UUID 的数量

适配器连接 BLE 设备后，默认会列出所有特征值 UUID.

示例

```
AT+CONMKM-BLE
Scanning...
Connecting
OK+CONN:0x1CCAE32FC8AF
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify
Chars Found: 5
```

7. AT+CON 通过 MAC 地址连接设备

AT+CON	
执行指令	响应
AT+CON<MAC>	**Scanning...** **Connecting** **OK+CONN:<MAC>** **<NUM>: <UUID>,<PROP1>, <PROP2>,...<PROPn>** **...** **Chars Found: <QUANTITY>**

参数

<MAC> 已连接 BLE 设备的 MAC 地址

<NUM> 已连接的 BLE 蓝牙设备的 UUID 索引

<UUID> 已连接的 BLE 蓝牙设备的特征值 UUID

<PROP(n)> 已连接的 BLE 蓝牙设备的特征值 UUID 的属性

< QUANTITY>已连接的 BLE 蓝牙设备的特征值 UUID 的数量

适配器连接 BLE 设备后，默认会列出所有特征值 UUID.

示例

```
AT+CON1CCAE32FC8AF
Scanning...
Connecting
OK+CONN:0x1CCAE32FC8AF
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify
Chars Found: 5
```

8. AT+CONNL 连接上一次成功连接过的设备

The dongle can use this command to connect to the last successfully connected device.

AT +CONNL	
执行指令	响应
AT+CONNL	**Connecting** **OK+CONN:<MAC>** **<NUM>: <UUID>,<PROP1>, <PROP2>,...,<PROPn>** **...** **<NUM+n>: <UUID>,<PROP1>, <PROP2>,...,<PROPn>** **Chars Found: <QUANTITY>**

参数

<MAC>	已连接 BLE 设备的 MAC 地址
<NUM>	已连接的 BLE 蓝牙设备的 UUID 索引
<UUID>	已连接的 BLE 蓝牙设备的特征值 UUID

<PROP(n)> 已连接的 BLE 蓝牙设备的特征值 UUID 的属性

<QUANTITY> 已连接的 BLE 蓝牙设备的特征值 UUID 的数量

适配器连接 BLE 设备后，默认会列出所有特征值 UUID.

示例

```
AT+CONNL
Connecting
OK+CONN:0x1CCAE32FC8AF
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify
Chars Found: 5
```

9. AT+DISCON 断开连接

AT+DISCON	
执行指令 **AT+DISCON**	响应 **OK+LOST**

适配器收到 AT+DISCON 指令后，会与连接的的设备断开，并返回 **OK+LOST**

示例

```
AT+DISCON

OK+LOST
```

10.AT+SERV 设置 Service UUID

AT+SERV	
执行指令 **AT+SERV<UUID>**	响应 **OK+Set:<UUID>**
执行指令 **AT+SERV<INDEX>** (连接后可用)	**OK+Set:<UUID>**
查询指令 **AT+SERV?**	响应 **OK+Get:<UUID>**

参数

<UUID> 已发现服务的 UUID

<INDEX> 已发现服务 UUID 的索引

设置指定服务 UUID ， 适配器就回发现该服务下的特征值 UUID。如果没有设定服务 UUID,那么适配器连接设备后会发现所有服务下的特征值 UUID.

示例

AT+SERVFFF0

OK+Set:FFF0

11.AT+SERVS 搜索 service UUID

AT+SERVS	
执行指令	响应
AT+SERVS	**<INDEX>:<UUID>** **...** **<INDEX+n>:<UUID>** **Services Found: <QUANTITY>**

参数

<INDEX> 已连接设备服务 UUID 的索引

<UUID> 已连接设备服务 UUID

<QUANTITY> 发现所有服务 UUID 的数量

用来查询已连接设备的服务 UUID

示例

AT+SERVS
0: 1800
1: 1801
2: 81469E83-EF6F-42AF-B1C6-F339DBDCE2EA
Services Found: 3
AT+SERV2
OK+Set: 81469E83-EF6F-42AF-B1C6-F339DBDCE2EA

12.AT+CHARS　刷新特征值 UUID

AT+CHARS	
执行指令	响应
AT+CHARS	**< INDEX >:<UUID>,<PRO1>,...,<PROPn>** **...** **< INDEX +n>:<UUID>,<PRO1>,...,<PROPn>** **Chars Found: <QUANTITY>**

参数

< INDEX >　已连接的 BLE 蓝牙设备的 UUID 索引

<UUID>　已连接的 BLE 蓝牙设备的特征值 UUID

<PROP(n)>　已连接的 BLE 蓝牙设备的特征值 UUID 的属性

<QUANTITY>已连接的 BLE 蓝牙设备的特征值 UUID 的数量

用来刷新已连接设备的特征值 UUID。如果设置了服务 UUID,那么只列出该服务下的特征值 UUID.

示例

```
AT+CHARS
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify
Chars Found: 5
```

13.AT+CHAR?　查询已经发现的特征值 UUID

AT+CHAR?	
执行指令	响应
AT+CHAR?	**<INDEX>:<UUID>,<PRO1>,...,<PROPn>** **...** **< INDEX +n>:<UUID>,<PRO1>,...,<PROPn>** **Chars Found: <QUANTITY>**

参数

< INDEX >　已连接的 BLE 蓝牙设备的 UUID 索引

<UUID>　已连接的 BLE 蓝牙设备的特征值 UUID

<PROP(n)> 已连接的 BLE 蓝牙设备的特征值 UUID 的属性

<QUANTITY> 已连接的 BLE 蓝牙设备的特征值 UUID 的数量

这个指令用来刷新当前特征值 UUID 列表

示例

```
AT+CHAR?
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify
Chars Found: 5
```

14. AT+CHRX 设置接收特征值 UUID

适配器支持 16 位及 128 位 UUID 特征值。这个指令用来设置适配器接收特征值 UUID.用来接收 BLE 从机设备 Notify 数据或者 Indicate 数据。

AT+CHRX	
执行指令 **AT+CHRX<UUID>**	响应 **OK+Set:<UUID>**
执行指令 **AT+CHRX<INDEX>** (连接之后可用)	**OK+Set:<UUID>**

参数

<UUID> 接收数据特征值 UUID

<INDEX> 已连接的 BLE 蓝牙设备的特征值 UUID 的索引

示例 1

AT+CHRXFFF0

OK+Set:FFF0

示例 2

AT+CHRX8146C201EF6F42AFB1C6F339DBDCE2EA

OK+Set:8146C201-EF6F-42AF-B1C6-F339DBDCE2EA

示例 3

```
AT+CON1CCAE326226D          //通过 MAC 连接从机设备
Connecting
OK+CONN:0x1CCAE326226D
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify
Chars Found: 5
AT+CHRX4                    // 通过索引设置接收数据特征值 UUID
OK+Set:8146C201-EF6F-42AF-B1C6-F339DBDCE2EA
AT+CHARS                    // 刷新特征值 UUID
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify; RX
Chars Found: 5
```

15.AT+CHTX 设置适配器发送数据特征值 UUID

适配器支持 16 位及 128 位 UUID 特征值。这个指令用来设置适配器发送特征值 UUID.用来向 BLE 从机发送数据。

AT+CHTX	
执行指令 **AT+CHTX<UUID>**	响应 **OK+Set:<UUID>**
执行指令 **AT+CHTX<INDEX>** (连接后可用)	**OK+Set:<UUID>**

参数

<UUID> 发送数据特征值 UUID

<INDEX> 已连接的 BLE 蓝牙设备的特征值 UUID 的索引

示例 1

```
AT+CHTXFFF0

OK+Set:FFF0
```

示例 2

```
AT+CHTX8146C201EF6F42AFB1C6F339DBDCE2EA

OK+Set:8146C201-EF6F-42AF-B1C6-F339DBDCE2EA
```

示例 3

```
AT+CON1CCAE326226D            //通过 MAC 连接从机设备
Connecting
OK+CONN:0x1CCAE326226D
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify
Chars Found: 5
AT+CHTX3                      //通过索引设置发送数据特征值 UUID
OK+Set:8146C201-EF6F-42AF-B1C6-F339DBDCE2EA
AT+CHARS                      //刷新特征值 UUID
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write; TX
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify; RX
Chars Found: 5
```

16. AT+SHCH 设置特征值列表显示开关

AT+SHCH	
执行指令 **AT+SHCH<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+SHCH?**	响应 **OK+Get:<VALUE>**

参数

<VALUE>	1	打开特征值列表显示开关 (默认值 显示特征值列表)
	0	关闭特征值列表显示开关

这个指令的作用是用于连接从机蓝牙后 是否显示特征值列表

示例

```
AT+SHCH1
OK+Set:1
AT+CON1CCAE326226D
Connecting
OK+CONN:0x1CCAE326226D
0: 2A00, Read, Write
1: 2A01, Read
2: 2A04, Read
3: 8146C203-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Write; TX
4: 8146C201-EF6F-42AF-B1C6-F339DBDCE2EA, Read, Notify; RX
Chars Found: 5
AT+DISCON
OK+LOST
AT+SHCH0
OK+Set:0
AT+CON1CCAE326226D
Connecting
OK+CONN:0x1CCAE326226D
```

17.AT+SDUR 设置搜索时间

AT+SDUR	
执行指令 **AT+SDUR<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+SDUR?**	响应 **OK+Get:<VALUE>**

参数

<VALUE> 1 设置搜索时间为 1s

2 设置搜索时间为 2s

3 设置搜索时间为 3s

<u>4</u> 设置搜索时间为 4s(默认值)

…

99 最大值为 99s

设置的时间单位默认为秒（s）

示例

AT+SDUR5 // 设置搜索时间为 5s
OK+Set:5

18. AT+RENEW 恢复出厂设置

AT+RENEW	
执行指令	响应
AT+RENEW	**OK+RENEW**

参数

NONE

所有参数恢复到出厂默认状态

示例

```
AT+RENEW
OK+RENEW
```

19. AT+RESET 重新启动

适配器会重新启动。建议批量测产品时，每测完一个产品，发次命令重启一次适配器，会提高产线测试效率。

AT+RESET	
执行指令	响应
AT+RESET	**OK+RESET**

参数

NONE

示例

```
AT+RESET
OK+RESET
```

20.AT+RSSI? 查询已连接设备的 RSSI 值

AT+RSSI?	
执行指令	响应
AT+RSSI?	**RSSI Start...** **RSSI (dB): <VALUE1>** **RSSI (dB): <VALUE2>** **...** **RSSI (dB): <VALUEn>**
AT+RSSI?	**RSSI Cancelled**

参数

<VALUEn>	**RSSI** 值

第一次发送 AT+RSSI?，执行开始查询

第二次发送 AT+RSSI?，停止查询

示例

```
AT+RSSI?
RSSI Start...
RSSI (dB): -51
RSSI (dB): -49
RSSI (dB): -51
AT+RSSI?
RSSI Cancelled
```

21.AT+RADD? 查询最后一次连接过的设备 MAC 地址（成功连接）

AT+RADD?	
查询指令 **AT+RADD?**	响应 **OK+Get:<MAC>**

参数

<MAC>	最后一次连接过的设备 MAC 地址（成功连接）

示例

AT+RADD?

OK+Get:0x3CFA431B6C99

最后一次连接设备的 MAC 地址为： **0x3CFA431B6C99**

22.AT+BAUD 查询/设置波特率

AT+BAUD	
执行指令 **AT+BAUD<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+BAUD?**	**OK+Get:<VALUE>**

参数

<VALUE>	0 设置波特率为 9600 1 设置波特率为 19200 2 设置波特率为 38400

3　设置波特率为 57600

4　设置波特率为 115200

示例

```
AT+BAUD4          //设置波特率为 115200 bps
OK+Set:4
```

示例

```
AT+BAUD?
OK+Get:4          //查询到的波特率为 115200 bps
```

23. AT+PARI　查询/设置串口（UART）校验方式

AT+PARI	
执行指令 **AT+PARI<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+PARI?**	**OK+Get:<VALUE>**

参数

<VALUE>　0　设置串口（UART）校验位为　0　无校验　(默认值)

1　设置串口（UART）校验位为 1　奇校验

2　设置串口（UART）校验位为 2　偶校验

示例

```
AT+ PARI0          // Set uart parity NO PARITY
OK+Set:0
```

示例

AT+ PARI?
OK+Get:0　　　// Query uart parity is NO PARITY

24.AT+PAIR　　查询/设置配对模式

AT+PAIR	
执行指令 **AT+PAIR<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+PAIR?**	**OK+Get:<VALUE>**

参数

<VALUE>　0　无需配对，不需要密码直接连接（默认值）

1 简单配对，静态密码验证

2 动态配对模式，动态密码验证

4 安全验证模式，数字确认验证

注：静态密码是从设备的固定密码，并且每次密码都相同。

动态密码是蓝牙从设备的随机密码。每次建立连接时，密码都会更改。同时，AT＋PASS 命令用于输入动态密码。

安全配对模式通过数字确认验证。用户可以使用 AT＋PASS1 或 AT＋PASS0 接受或拒绝连接。

示例

```
AT+ PARI0          // 设置无需配对模式
OK+Set:0
```

示例

```
AT+ PARI?
OK+Get:0          // 查询配对模式
```

25.AT+PASS　　查询/设置配对密码

AT+PASS	
执行指令 **AT+PASS <VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+PASS?**	**OK+Get:<VALUE>**

参数

<VALUE>	000000-999999	密码

注：如果 BLE 从机设备的密码是一个静态的（固定的），那么适配器需要在连接前需要设置好密码。

如果 BLE 从机设备的密码是动态的。那么配对模式需要设置成 **AT+ PARI2.**连接后，通过这个指令输入动态密码。

如果 BLE 从机设备是安全认证模式，那么配对模式需要设置成 **AT+ PARI4.** 适配器连接 BLE 设备后，会通过串口发出，是否接受配对，回复‘0’

(AT+PASS0)，表示拒绝配对；回复非‘0’的数字（AT+PASS1），表示接受配对。

示例

```
AT+ PASS123456          //设置密码为 123456
OK+Set:123456
```

示例

```
AT+ PASS?
OK+Get:123456          // 查询到的密码为 123456
```

26. AT+RDCH 读取特征值 UUID 的值

AT+RDCH	
执行指令 **AT+RDCH<UUID>**	响应 **OK+Get:<VALUE>**

参数

<UUID> 需要读取值的 UUID

<VALUE> 读取到的值

示例

```
AT+RDCHFFE0          //
OK+Get:123456   // 读取到值为 123456
```

27. AT+TXPW 设置适配器发射功率

AT+TXPW	
执行指令 **AT+TXPW<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+TXPW?**	**OK+Get:<VALUE>**

参数

<VALUE>	0	设置发射功率为 -21dbm
	1	设置发射功率为 -18dbm
	2	设置发射功率为 -15dbm
	3	设置发射功率为 -12dbm
	4	设置发射功率为 -9dbm
	5	设置发射功率为 -6dbm
	6	设置发射功率为 -3dbm
	<u>7</u>	设置发射功率为 0dbm(默认值)
	8	设置发射功率为 +1dbm
	9	设置发射功率为+2dbm
	A	设置发射功率为 +3dbm
	B	设置发射功率为 +4dbm
	<u>C</u>	设置发射功率为 +5dbm

示例 1

AT+TXPWC // 设置发射功率为 +5dbm
OK+Set:C

示例 2

AT+TXPW?
OK+Get:0 // 查询到发射功率为-21dbm

28. AT+GAIN 查询/设置接收增益

AT+GAIN	
执行指令 **AT+GAIN<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+GAIN?**	**OK+Get:<VALUE>**

参数

<VALUE>	0 标准接收增益 **(默认值)** 1 高接收增益

示例 1

AT+GAIN1 // 设置适配器为高接收增益
OK+Set:1

示例 2

AT+GAIN?
OK+Get:0 // 查询到适配器设置的为标准接收增益

29.AT+AUCO设置适配器自动连接模式

AT+AUCO	
执行指令 **AT+AUCO<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+AUCO?**	**OK+Get:<VALUE>**

参数

<VALUE>	0	不自动连接 **(默认值)**
	1	自动连接

示例 1

AT+AUCO1 // 设置适配器为自动连接模式
OK+Set:1

示例 2

AT+AUCO?
OK+Get:0 // 查询适配器当前模式为：不自动连接 BLE 从机模式

30.AT+STATE? 查询适配器的连接状态

AT+STATE?	
查询指令 **AT+STATE?**	响应 **OK+Get:<VALUE>**

参数

<VALUE>	0	未连接
	1	已连接

示例 1

```
AT+STATE?
OK+Set:1              // 连接状态
```

示例 2

```
AT+ STATE?
OK+Get:0            // 适配器是未连接状态
```

31.AT+FRSSI 设置搜索过滤的信号强度

AT+FRSSI	
执行指令 **AT+FRSSI<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+ FRSSI?**	**OK+Get:<VALUE>**

（V15 版本增加）

参数

<VALUE>	设置过滤的信号强度，即过滤小于设置的信号强度（负值）
	当 VALUE=0 时， 不过滤（即取消过滤）

示例 1

```
AT+FRSSI-60          // 设置信号强度过滤掉小于-60db 设备
OK+Set:-60
```

示例 2

```
AT+FRSSI?
OK+Get:0xC4          // 查询的结果是带符号的十六进制数表示， C4 对应-60db
```

32.AT+NICH　　打开(使能)指定特征值服务属性

AT+NICH	
执行指令	响应
AT+NICH<UUID>	无

（V12 版本增加）

参数

<UUID>	为需要使能的 UUID 或者 UUID 对应的索引

示例

```
AT+NICHFFF1     // 使能从机设备 UUID 为 FFF1 的 Notify 属性或者 Indicate 属性
```

33.AT+NOCH　　关闭指定特征值服务属性

AT+NOCH	
执行指令	响应
AT+NOCH<UUID>	无

（V12 版本增加）

参数

<UUID>	为需要使能的 UUID 或者 UUID 对应的索引

示例

AT+NOCHFFF1 // 关闭从机设备 UUID 为 FFF1 的 Notify 属性或者 Indicate 属性

34.AT+SCTRL 设置、解析及过滤广播包字段指令

AT+SCTRL	
执行指令 **AT+SCTRL<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+ SCTRL?**	**OK+Get:<VALUE>**

（V13 版本增加）

这个指令的应用是根据广播包过滤需要的设备。比如过滤指定 UUID 的设备，或者根据广播内容自定义过滤。广播包的定义是从机定义的，适配器只能读取。

参数

<VALUE> 0x00 //不显示及不过滤广播包字段

```
#define GAP_ADTYPE_FLAGS                         0x01 //!< @ref GAP_Discovery
#define GAP_ADTYPE_16BIT_MORE                    0x02 //!< Service: More 16-bit UUIDs available
#define GAP_ADTYPE_16BIT_COMPLETE                0x03 //!< Service: Complete list of 16-bit UUIDs
#define GAP_ADTYPE_32BIT_MORE                    0x04 //!< Service: More 32-bit UUIDs available
#define GAP_ADTYPE_32BIT_COMPLETE                0x05 //!< Service: Complete list of 32-bit UUIDs
#define GAP_ADTYPE_128BIT_MORE                   0x06 //!< Service: More 128-bit UUIDs available
#define GAP_ADTYPE_128BIT_COMPLETE               0x07 //!< Service: Complete list of 128-bit UUIDs
#define GAP_ADTYPE_LOCAL_NAME_SHORT              0x08 //!< Shortened local name
#define GAP_ADTYPE_LOCAL_NAME_COMPLETE           0x09 //!< Complete local name
#define GAP_ADTYPE_POWER_LEVEL                   0x0A //!< TX Power Level: 0xXX: -127 to +127 dBm
#define GAP_ADTYPE_OOB_CLASS_OF_DEVICE           0x0D //!< Simple Pairing OOB Tag: Class of device (3 octets)
#define GAP_ADTYPE_OOB_SIMPLE_PAIRING_HASHC      0x0E //!< Simple Pairing OOB Tag: Simple Pairing Hash C (16 octets)
#define GAP_ADTYPE_OOB_SIMPLE_PAIRING_RANDR      0x0F //!< Simple Pairing OOB Tag: Simple Pairing Randomizer R (16 octets)
#define GAP_ADTYPE_SM_TK                         0x10 //!< Security Manager TK Value
#define GAP_ADTYPE_SM_OOB_FLAG                   0x11 //!< Security Manager OOB Flags
#define GAP_ADTYPE_SLAVE_CONN_INTERVAL_RANGE     0x12 //!< Min and Max values of the connection interval (2 octets Min, 2 octets Ma
#define GAP_ADTYPE_SIGNED_DATA                   0x13 //!< Signed Data field
#define GAP_ADTYPE_SERVICES_LIST_16BIT           0x14 //!< Service Solicitation: list of 16-bit Service UUIDs
#define GAP_ADTYPE_SERVICES_LIST_128BIT          0x15 //!< Service Solicitation: list of 128-bit Service UUIDs
#define GAP_ADTYPE_SERVICE_DATA                  0x16 //!< Service Data - 16-bit UUID
#define GAP_ADTYPE_PUBLIC_TARGET_ADDR            0x17 //!< Public Target Address
#define GAP_ADTYPE_RANDOM_TARGET_ADDR            0x18 //!< Random Target Address
#define GAP_ADTYPE_APPEARANCE                    0x19 //!< Appearance
#define GAP_ADTYPE_ADV_INTERVAL                  0x1A //!< Advertising Interval
#define GAP_ADTYPE_LE_BD_ADDR                    0x1B //!< LE Bluetooth Device Address
#define GAP_ADTYPE_LE_ROLE                       0x1C //!< LE Role
#define GAP_ADTYPE_SIMPLE_PAIRING_HASHC_256      0x1D //!< Simple Pairing Hash C-256
#define GAP_ADTYPE_SIMPLE_PAIRING_RANDR_256      0x1E //!< Simple Pairing Randomizer R-256
#define GAP_ADTYPE_SERVICE_DATA_32BIT            0x20 //!< Service Data - 32-bit UUID
#define GAP_ADTYPE_SERVICE_DATA_128BIT           0x21 //!< Service Data - 128-bit UUID
#define GAP_ADTYPE_3D_INFO_DATA                  0x3D //!< 3D Information Data
#define GAP_ADTYPE_MANUFACTURER_SPECIFIC         0xFF //!< Manufacturer Specific Data: first 2 octets contain the Company Identifie
```

示例

```
AT+SCTRL01         // 设置过滤广播包中有 GAP_ADTYPE_FLAGS 字段的设备
OK+Set:1

AT+SCAN?          // 发搜索指令可以看到过滤后的结果

Scanning...
0: 0x1CCAE328BE93, -66, SBC-ST102100663075, <020106>
1: 0x1CCAE32FC8AF, -64, , <020106>
2: 0x1CCAE325E051, -71, , <020106>
3: 0x1CCAE326226D, -61, , <020106>
4: 0x1CCAE32FC512, -56, SBC-ST102101123490, <020106>
5: 0x1CCAE325E0CB, -58, , <020106>
Devices Found: 6
```

35. AT+RIPRD 设置 RSSI 读取间隔

AT+RIPRD	
执行指令 **AT+RIPRD<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+RIPRD?**	响应 **OK+Get:<VALUE>**

参数

<VALUE>	1	设置读取间隔为 1s (默认值)
	2	设置读取间隔为 2s
	3	设置读取间隔为 3s
	4	设置读取间隔为 4s
	…	
	99	最大值为 99s

设置的时间单位默认为秒（s）

示例

```
AT+ RIPRD2          //设置读取间隔为 2s
OK+Set:2
```

36.AT+DIDLY 设置发现服务延时时间

AT+DIDLY	
执行指令 **AT+DIDLY<VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+DIDLY?**	响应 **OK+Get:<VALUE>**

（V17 版本增加）

参数

<VALUE>

1 发现服务延时时间为 1s (默认值)

2 发现服务延时时间为 2s

3 发现服务延时时间为 3s

4 发现服务延时时间为 4s

…

99 最大值为 99s

设置的时间单位默认为秒（s）

示例

```
AT+DIDLY1          //发现服务延时时间为 1s
OK+Set:1
```

37.AT+UCTRL 设置 扫描时间/发现服务/读取 RSSI 时间单位

AT+DIDLY	
执行指令 **AT+UCTRL <VALUE>**	响应 **OK+Set:<VALUE>**
查询指令 **AT+UCTRL?**	响应 **OK+Get:<VALUE>**

（V17 版本增加）

参数

<VALUE>

VALUE 值的各个 bit 表示的含义

字节	bit7	bit6	bit5	bit4	bit3	bit2	bit1	Bit0
描述	-	-	RSSI period		SVC DISC delay		Scan units	

Scan units: 扫描时间基本时间单位:默认为 1000ms

相关的 AT 指令: AT+SDUR

可通过这两个 bits 位更改扫描时间单位

bit1	**bit0**	**时间的单位**
0	**0**	**1000ms**
0	**1**	**100ms**
1	**0**	**10ms**
1	**1**	**1ms**

SVC DISC delay: 连接后服务发现延迟时间单位:默认为 1000ms

相关的 AT 指令:AT+DIDLY

可通过这两个 bits 位更改连接后服务发现延迟时间单位

bit3	bit2	时间的单位
0	0	1000ms
0	1	100ms
1	0	10ms
1	1	1ms

RSSI period: 信号强度返回时间单位:默认为 1000ms

相关的 AT 指令:AT+RSSI?

可通过这两个 bits 位更改信号强度返回时间单位

bit5	bit4	时间的单位
0	0	1000ms
0	1	100ms
1	0	10ms
1	1	1ms

bit6 及 bit7 未使用，无意义。

示例

AT+ UCTRL01 //将搜索时间的单位改为 100ms, 如果 AT+SDUR 为默认值 4 时, 那么搜索时间由 4000 变为 400ms

OK+Set: 0x1