ASCII Interface 2.0.0 Manual

$Revision: 1.4 $

Bluegiga Technologies

ASCII Interface 2.0.0 Manual

by Bluegiga Technologies

Table of Contents

Preface

WRAP THOR ASCII Interface is firmware which allows easy access to Bluetooth functionality. It makes the radio interface totally transparent and host system can control connections with simple ASCII commands strings. This makes transition to wireless world easy as no specific Bluetooth know-how has to be obtained.

Typographical Conventions

Different typographical conventions used in this manual are described in this chapter.

- Screen output seen on terminal is presented as follows:

```
OUTPUT FROM ASCII Interface
INPUT FROM USER
MORE OUTPUT
```

- Command and output synopsis are presented as follows:

  COMMAND {required parameter} [optional parameter] STATIC TEXT [2nd optional parameter]

- Command and event references are presented as follows:

  COMMAND and `EVENT`.

Chapter 1. Usage

ASCII Interface is terminal controlled firmware which means it can be used with any terminal emulation software, such as Hyperterminal in Windows or Minicom in Linux. Initial port settings for ASCII interface are 115200,8n1 (baud rate 115200 bps, 8 data bits, no parity, one stop bit) and hardware flow control enabled. When you power-on the module or evaluation kit you should see the command prompt appear on the terminal emulation software.

After power-on you can check ASCII Interface configuration, such as Bluetooth device address, by command **SET**.

Example 1-1. ASCII Interface at initial state

```
WRAP THOR AI (version 2.0.0-rc1 build 344 $ bt1.1)
Copyright (c) 2003-2004 Bluegiga Technologies Inc.
READY.
SET
SET BT BDADDR 00:07:80:a5:c1:11
SET BT NAME WRAP AI
SET BT CLASS 001f00
SET Control BAUD 115200,8n1
SET Control ECHO 7
SET
```

Chapter 2. Operational Modes

ASCII Interface has two operational modes, command mode and data mode. Command mode is default mode when there is no connections. It is possible to switch between modes at any time when there are any connections. Data mode is not available if there is no connections (because there is not any data available).

Switching from data mode to command mode is issued with the following escape sequence:

<at least 1 second sleep> **+++** <at least 1 second sleep>

Same sequence or command **SELECT** may be used to return to data mode.

When ASCII Interface enters to command mode `READY` event is delivered (unless masked away with **SET CONTROL ECHO**).

Command Mode

Command mode is default mode when ASCII Interface is powered. In command mode commands can be entered to ASCII Interface to perform various activities.

Incoming data from remote devices is buffered when ASCII Interface is in command mode.

Note: Because of embedded nature of ASCII Interface buffering capabilities are low and only small amounts of data can be received to buffers.

Mode is changed from command mode to data mode when

- User switches mode either using escape sequence <1s>**+++**<1s> or using command **SELECT**.
- Connection is successfully created using command **CALL** (`CONNECT` event is used to notify for successful link creation).
- Remote device has connected us (`RING` event is used to notify for incoming connections).

Data mode

Data mode is default mode when there are any connections. In data mode all data is sent totally transparently from UART over the Bluetooth RFCOMM link to other device and vice versa.

Mode is changed from data mode to command mode when

- User switches mode using escape sequence <1s>**+++**<1s>.
- Link is terminated (closed by remote device or link loss) (`NO CARRIER` event is used to notify for link termination).

Chapter 3. Commands

This chapter describes different commands used to control the behaviour of ASCII Interface.

Every command is typed into one line and is executed by line feed (CR+LF, ASCII13+ASCII10). ASCII Interface is case insensitive ie. command may be entered in upper-, lower- or even mixed case letters.

CALL

Command **CALL** is used to initiate connections to the remote device. Connections are closed using command **CLOSE**. Currently open connections can be viewed using command **LIST**.

Synopsis

CALL {address} {target} RFCOMM

Description

address

Bluetooth device address of the remote device

target

RFCOMM target for the connection. Target may be one of the following:

channel

RFCOMM channel number

Format: xx (hex)

uuid16

16 bit UUID for searching channel

Format: xxxx (hex)

uuid32

32 bit UUID for searching channel

Format: xxxxxxxx (hex)

uuid128

128 bit UUID for searching channel

Format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (hex)

Response

```
CALL {link_id}
```

link_id

Numeric connection identifier

Events

- CONNECT event is delivered after successful **CALL** command.
- NO CARRIER event is delivered if **CALL** fails.

Examples

Example 3-1. Creating successful connection to 00:07:80:bf:bf:01 channel 1

```
CALL 00:07:80:bf:bf:01 1 RFCOMM
CALL 0
CONNECT 0 RFCOMM 1
```

Example 3-2. Creating successful connection to 00:07:80:bf:bf:01 Serial Port Profile (UUID16 SPP = 1101)

```
CALL 00:07:80:bf:bf:01 1101 RFCOMM
CALL 0
CONNECT 0 RFCOMM 2
```

Example 3-3. Unsuccessful connection attempt to 00:07:80:bf:bf:01

```
CALL 00:07:80:bf:bf:01 1 RFCOMM
CALL 0
NO CARRIER 0 ERROR 406 RFC_CONNECTION_FAILED
```

CLOSE

Command **CLOSE** is used to terminate previously opened connection. See command **CALL** for more information about opening connections.

Synopsis

CLOSE {link_id}

Description

link_id

Numeric connection identifier from previously used command **CALL** or from event **RING**.

Response

No response.

Events

- **NO CARRIER** event is delivered after link is closed.

INQUIRY

Command **INQUIRY** is used to find other Bluetooth devices in the area.

Synopsis

INQUIRY {timeout} [NAME]

Description

timeout

The maximum amount of time (in units of 1.28 seconds) before the inquiry process is halted

> **Note:** It may take up to 10.24 seconds for Bluetooth device to answer inquiry scan and thus timeout value should be at least 8 if it is necessary to find every device in the area.

NAME

Optional flag to automatically request friendly name for found devices, see command **NAME** for more information about remote name request

Response

```
INQUIRY {num_of_devices}
```

```
INQUIRY {addr} {class_of_device}*
```

num_of_devices

Amount of found devices

addr

Bluetooth device address of found device

class_of_device

Bluetooth Class of Device of found device

> **Note:** Response from **INQUIRY** comes after specified timeout.

Events

- `INQUIRY_PARTIAL` events are delivered as devices are found.
- `NAME` events are delivered after **INQUIRY** if NAME flag is present.

Examples

Example 3-4. Inquiry without friendly name request

```
INQUIRY 10
INQUIRY_PARTIAL 00:07:80:bf:bf:01 001f00
INQUIRY_PARTIAL 00:07:80:80:05:65 920300
INQUIRY_PARTIAL 00:07:80:80:32:e0 920300
INQUIRY 3
INQUIRY 00:07:80:bf:bf:01 001f00
INQUIRY 00:07:80:80:05:65 920300
INQUIRY 00:07:80:80:32:e0 920300
```

Example 3-5. Inquiry with friendly name request

```
INQUIRY 10 NAME
INQUIRY_PARTIAL 00:07:80:bf:bf:01 001f00
INQUIRY_PARTIAL 00:07:80:80:05:65 920300
INQUIRY_PARTIAL 00:07:80:80:32:e0 920300
INQUIRY 3
INQUIRY 00:07:80:bf:bf:01 001f00
INQUIRY 00:07:80:80:05:65 920300
INQUIRY 00:07:80:80:32:e0 920300
NAME 00:07:80:bf:bf:01 "AI bf:01"
NAME 00:07:80:80:05:65 "WRAP AS"
NAME 00:07:80:80:32:e0 "WRAP THOR"
```

LIST

Command **LIST** shows information about connections currently open.

Synopsis

LIST

Response

```
LIST {num_of_links}
```

```
LIST {link_id} CONNECTED RFCOMM {blocksize} 0 0 {elapsed_time}
{local_msc} {remote_msc} {addr} {channel} {direction} {powermode}
{role} {crypt}*
```

num_of_links

Number of currently open links

link_id

Numeric connection identifier

blocksize

Data packet size, ie. how many bytes data can be sent in one packet

elapsed_time

Link life time in seconds

local_msc & remote_msc

Serial port status bits, "8d" is normal value

addr

Bluetooth device address of the remote device

channel

RFCOMM channel number at remote device

direction

Direction of the link

"OUTGOING"

Link is initiated by local device (using command **CALL**)

"INCOMING"

Link is initiated by the remote device

powermode

Power mode for the link

"ACTIVE"

Link is in active mode

"SNIFF"

Link is in sniff mode

"HOLD"

Link is in hold mode

"PARK"

Link is in park mode

role

Role of the link

"MASTER"

ASCII Interface is the master device of this link

"SLAVE"

ASCII Interface is the slave device of this link

crypt

Encryption state of the link

"PLAIN"

Link is not encrypted

"ENCRYPTED"

Link is encrypted

Events

None.

Examples

Example 3-6. List with 1 active connection and 1 connection in sniff mode

```
LIST
LIST 2
LIST 0 CONNECTED RFCOMM 669 0 0 40 8d 8d 00:07:80:80:31:e6 1 INCOMING SNIFF SLAVE EN-
CRYPTED
```

```
LIST 1 CONNECTED RFCOMM 669 0 0 18 8d 8d 00:07:80:80:32:0e 1 OUTGOING AC-
TIVE MASTER ENCRYPTED
```

NAME

Command **NAME** is used retrieve friendly name of the device.

Synopsis

NAME {address}

Description

addr

Bluetooth device address of the device.

Response

None.

Events

- `NAME` event is delivered when friendly name is known.
- `NAME ERROR` event is delivered if friendly name lookup fails.

Examples

Example 3-7. Successful name query

```
NAME 00:07:80:bf:bf:01
NAME 00:07:80:bf:bf:01 "AI bf:01"
```

Example 3-8. Unsuccessful name query

```
NAME 00:07:80:bf:bf:bf
NAME ERROR 104 00:07:80:bf:bf:bf HCI_ERROR_PAGE_TIMEOUT
```

RESET

Command **RESET** is used to reset ASCII Interface.

Synopsis

RESET

Response

None.

Events

None.

SELECT

Command **SELECT** is used to switch to data mode.

Synopsis

SELECT {link_id}

Description

link_id

Numeric connection identifier

Response

None. ASCII Interface goes to data mode with the link link_id.

Events

None.

SET

SET displays or sets configuration values of ASCII Interface.

Synopsis

SET [{category} {option} {value}]

Description

Without any parameters **SET** displays current configuration.

category

Category of setting

"BT"

Changes different Bluetooth related settings. See **SET BT** for more information about options.

"CONTROL"

Changes different ASCII Interface settings. See **SET CONTROL** for more information about options.

option

Option name, depends on category. See following sections for more information.

value

Value for option. See following sections for more information.

Response

- If issued without parameters:

  ```
  SET {category} {option} [value]*
  SET
  ```

- If issued with parameters:

None.

Events

None.

SET BT

Bluetooth related settings.

SET BT BDADDR

List format

```
SET BT BDADDR {addr}
```

addr

Bluetooth device address of local device

Note: This value is read-only.

SET BT NAME

List format

```
SET BT NAME {friendly_name}
```

Set format

SET BT NAME [friendly_name]

friendly_name

Friendly name of local device

Warning

If friendly_name is left empty some device may have problems showing device.

SET BT CLASS

List format

```
SET BT CLASS {class_of_device}
```

Set format

SET BT CLASS {class_of_device}

class_of_device

Bluetooth Class of Device of local device

SET BT AUTH

List format

```
SET BT AUTH * {pin_code}
```

Note: SET BT AUTH is not visible if pin_code is disabled.

Set format

SET BT AUTH * [pin_code]

pin_code

Pin code for authorized connections. Authorization is required if this option is present.

SET BT PAIR

List format

```
SET BT PAIR {addr} {link_key}
```

Note: SET BT PAIR is not visible if there are not paired devices.

Set format

SET BT PAIR {addr} [link_key]

addr

Bluetooth device address of the paired device link_key

Link key for authenticated connection

To remove device from list of known devices left link_key parameter empty.

Tip: To remove every known device use * as addr (**SET BT PAIR ***).

SET CONTROL

Common ASCII Interface settings.

SET CONTROL BAUD

List format

```
SET CONTROL BAUD {baud_rate},8{parity}{stop_bits}
```

Set format

SET CONTROL BAUD {baud_rate} ,8 {parity} {stop_bits}

Important: Parameters in **SET CONTROL BAUD** must be typed together!

baud_rate

UART baud rate in bps

",8"

Static string indicating UART uses 8 data bits

parity

UART parity setting

"n"

None parity

"e"

Even parity

"o"

Odd parity

stop_bits

Number of stop bits in UART communications

"1"

One stop bit

"2"

Two stop bits

SET CONTROL ECHO

List format

```
SET CONTROL ECHO {echo_mask}
```

Set format

SET CONTROL ECHO {echo_mask}

echo_mask

Bit mask for controlling echo and events displaying

Bit 0

If set start-up banner is visible

Bit 1

If set characters are echoed back to client in command mode

Bit 2

If set events are displayed when in command mode

Default value for **SET CONTROL ECHO** is 7 (bits 0..2 set).

Warning

If every bit is set off (value 0) it is quite impossible to know the status of ASCII Interface.

If Bit 2 is set off it is very hard to detect whether ASCII Interface is in command mode or in data mode.

SET CONTROL INIT

List format

```
SET CONTROL INIT {command}
```

Set format

SET CONTROL INIT [command]

command

Any ASCII Interface command string.

This command is automatically executed every time ASCII Interface starts (after power-on, **RESET** or watchdog event)

TESTMODE

Command **TESTMODE** enables Bluetooth Test Mode in which Bluetooth Testers may be used to test radio environment.

Synopsis

TESTMODE

Response

```
TEST 0
```

Events

None.

Chapter 4. Events

Events are mechanism that ASCII Interface uses to notify the User for completed commands, incoming connections, etc. If ASCII Interface is in data mode only possible event is `NO CARRIER` event for corresponding link.

Events may be masked away by removing Bit 2 on command **SET CONTROL ECHO**.

> **Note:** ASCII Interface is designed so that unwanted events can be safely ignored. Events `CONNECT`, NO `CARRIER` and `RING` change the mode of operation and therefore they cannot be ignored.

CONNECT

`CONNECT` event is used to notify for successful link establishment.

> **Note:** ASCII Interface automatically goes into data mode after `CONNECT` event.

Synopsis

CONNECT {link_id} RFCOMM {channel}

Description

link_id

Numeric connection identifier.

channel

Connected RFCOMM channel number.

See also

CALL, **LIST**

INQUIRY_PARTIAL

`INQUIRY_PARTIAL` event is used to notify found Bluetooth device. This event precedes response for **INQUIRY** command.

Synopsis

INQUIRY_PARTIAL {addr} {class_of_device}

Description

addr

Bluetooth device address of found device.

class_of_device

Bluetooth Class of Device of found device.

See also

INQUIRY

NO CARRIER

`NO CARRIER` event is used to notify for link loss or alternatively failure in link establishment.

Synopsis

NO CARRIER {link_id} RFCOMM {error_code} [message]

Description

link_id

Numeric connection identifier

error_code

Code describing error

message

Optional verbose error message

See also

CALL, **CLOSE**, **LIST**, **RING**

READY

`READY` event is used to notify for switching to command mode.

Synopsis

READY.

See also

Operational modes

NAME

`NAME` event is used to notify for successful lookup for Bluetooth friendly name of the remote device.

Synopsis

NAME {addr} {"friendly_name"}

Description

addr

Bluetooth device address of the device.

friendly_name

Friendly name of the device.

See also

INQUIRY, **NAME**

NAME ERROR

`NAME ERROR` event is used to notify for Bluetooth friendly name lookup failure.

Synopsis

NAME ERROR {error_code} {addr} [message]

Description

error_code

Code describing error.

addr

Bluetooth device address of the device.

message

Optional verbose error message.

See also

INQUIRY, **NAME**

RING

`RING` event is used to notify for incoming connection. Incoming connections are accepted only if there is no existing links.

Synopsis

RING {link_id} {addr} {channel} RFCOMM

Description

link_id

Numeric connection identifier

addr

Bluetooth device address of the remote device

channel

Local RFCOMM channel

See also

CLOSE, **LIST**

SYNTAX ERROR

`SYNTAX ERROR` is not an actual event but error message describing faulty typed command or error in command parameters.

Synopsis

SYNTAX ERROR

Chapter 5. Troubleshooting

This chapter introduces some usual error situations with possible solutions. Before contacting Bluegiga Technologies Technical Support at `<support@bluegiga.com>` please carefully check through this chapter.

Problem	Possible solutions
ASCII Interface does not start or output is just some garbage	Check your cable and terminal emulation settings. Default terminal settings are 115200,8n1 (baud rate 115200 bps, 8 data bits, no parity, one stop bit).

Appendix A. Acronyms and Definitions

BluetoothTM	Set of technologies providing audio and data transfer over short-range radio connections
bps	bits per second
hold mode	Bluetooth low power mode
park mode	Bluetooth low power mode
RFCOMM	Serial cable emulation protocol; element of Bluetooth
sniff mode	Bluetooth low power mode
UART	Universal Asynchronous Receiver Transmitter
UUID	Universally Unique Identifier
WRAP	Wireless Remote Access Platform; Bluegiga Technologies' wireless product family