ASCII Interface 2-0-0

User Manual

Version 1.1

Monday, July 19, 2004

blue giga
TECHNOLOGIES

Contents

List of Tables

1. VERSION HISTORY

Version:	Author:	Comments:
1.0	MS	Initial Version
1.1	MS	CALL–functions parameters corrected

2. TERMS & ABBREVIATIONS

Term or Abbreviation:	Explanation:
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
bps	bits per second
HCI	Host Controller Interface
hold mode	Bluetooth low power mode
park mode	Bluetooth low power mode
L2CAP	The Logical Link Control and Adaptation Layer Protocol
RFCOMM	Serial cable emulation protocol; element of Bluetooth
sniff mode	Bluetooth low power mode
UART	Universal Asynchronous Receiver Transmitter
UUID	Universally Unique Identifier
VM	Virtual Machine
WRAP	Wireless Remote Access Platform; Bluegiga Technologies' wireless product family

3. INTRODUCTION

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 the transition to wireless world easy, as no specific Bluetooth know-how has to be obtained.

4. FIRMWARE DISCLAIMER

It is **STRONGLY** suggested to take a backup of the modules initial firmware and parameters so it can be restored in case of a faulty firmware configuration. The backup of the firmware can be taken with the 'dump' option of the *Blueflash* program and the parameters with the 'dump' option if the *PSTool* program. The faulty configuration of the firmware may result in reduced performance or faulty operation. If you do not take the backup of the initial firmware and parameters Bluegiga Technologies can not provide you exact support in cases where modules are not equipped with original firmware.

5. GETTING STARTED

To start using the ASCII Interface, you can use, for example, terminal software such as *Tera Term*. When using *Tera Term*, make sure the module or WRAP THOR Evaluation Kit is connected to your PC serial port. Start the terminal software with the following default UART settings:

- 115200 baud
- 8 data bit
- 1 stop bit
- No parity
- Hardware Flow Control enabled

When you power on the module or evaluation kit you should see the command prompt appear on the terminal software. See picture below.

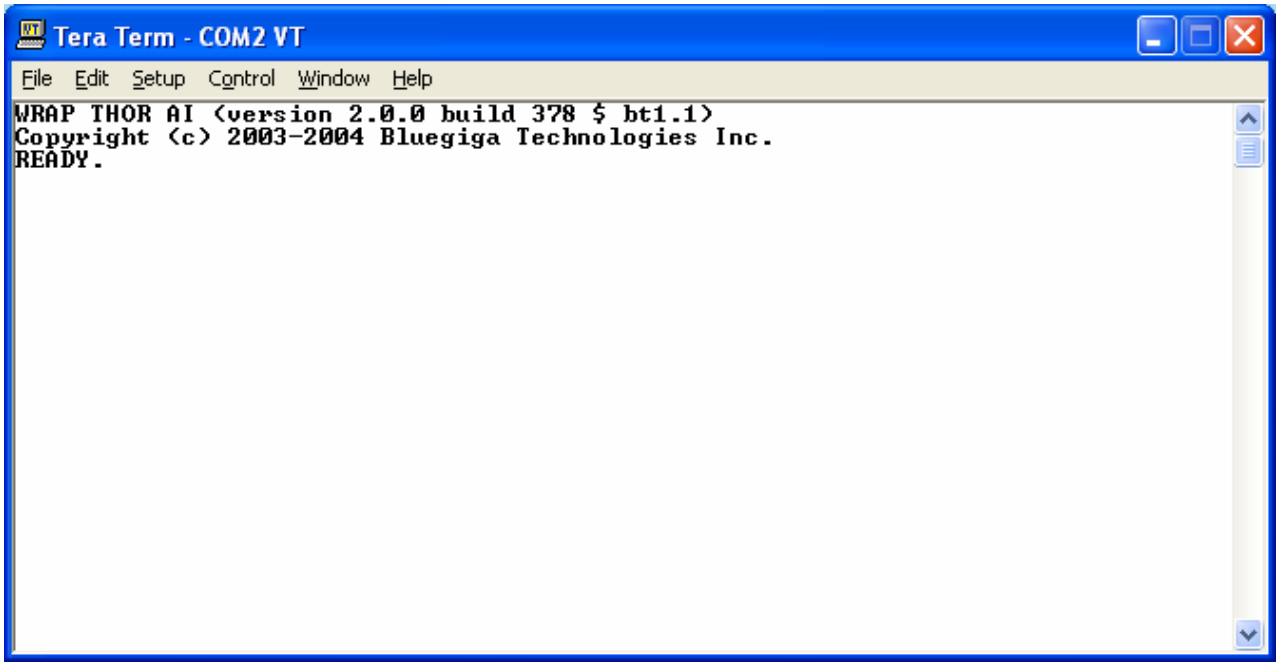


Figure 1: ASCII interface boot prompt

6. ASCII INTERFACE 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 obviously there is not any data available.

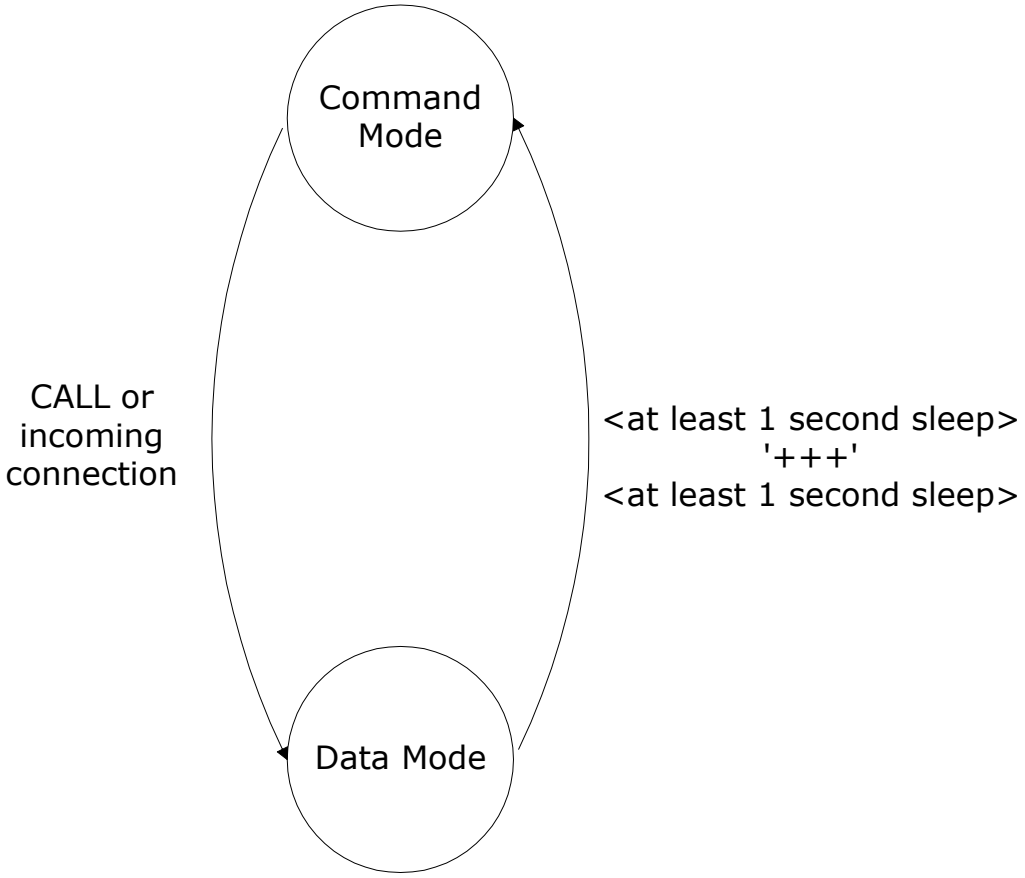


Figure 2: State Transitions

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**).

6.1 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.

6.2 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.

Initial mode	Target mode	Requirements for transition from initial mode to target mode
Command Mode (no link active) In this mode the device can be commanded to perform various actions such as inquiries, calls etc...	Data Mode	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 In this mode all data is sent totally transparently from RS-232 over the Bluetooth RFCOMM link to other device	Command Mode	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).

Table 1: ASCII interface modes and transitions

7. USAGE

ASCII Command Interface can be used from the HOST system by sending ASCII commands through UART. These commands should end with linefeed '\n' character.

When installed and configured the module can be commanded from the host with the following commands:

- CALL
- CLOSE
- HELP
- INQUIRY
- LIST
- NAME
- RESET
- SELECT
- SET
- TESTMODE

7.1 Typocraphical conventions

The commands and their usage are described in the later parts of this chapter. Commands and output synopsis are presented as follows:

Synopsis:
COMMAND {*required parameter*} [*optional parameter*] STATIC TEXT [*2ND OPTIONAL PARAMETER*] *or* **Alternative syntax**

Command parameters on the other hand are described like this:

Description:	
parameter	Description

Responses to the command are described as in the table below:

Response:	
RESPONSE { *parameters* }	
parameter	Description

Events generated by commands or actions are described as follow:

Events:
EVENT Description

And finally examples shown are described like this:

```
EXAMPLE COMMAND

RESPONSE TO COMMAND
```

NOTE!

- The parser is not case sensitive!
- ASCII Interface 0.0.2 does not accept backspaces, but 2.0.0 does.

7.2 INQUIRY

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

Synopsis:
INQUIRY {*timeout*} [*NAME*] *or* **I {timeout} [N]**

Description:	
timeout	The maximum amount of time (in units of 1.28 seconds) before the inquiry process is halted.
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!

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.

Events:

INQUIRY_PARTIAL events are delivered as devices are found.

NAME events are delivered after **INQUIRY** if ***NAME*** flag is present.

7.2.1 Examples

```
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
```

```
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"
```

7.3 LIST

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

Synopsis:
LIST *or* **L**

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:
No response

7.3.1 Examples

```
LIST
LIST 2
LIST 0 CONNECTED RFCOMM 669 0 0 40 8d 8d 00:07:80:80:31:e6 1
INCOMING SNIFF SLAVE ENCRYPTED
LIST 1 CONNECTED RFCOMM 669 0 0 18 8d 8d 00:07:80:80:32:0e 1
OUTGOING ACTIVE MASTER ENCRYPTED
```

7.4 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 *or* **C {address} {target} RFC**

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.

7.4.1 Examples

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
```

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
```

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
```

NOTE!

If CALL is used with CHANNEL instead of UUID it will be on average 300ms faster, since there is no need to do service discovery. However the channel of serial port profile (SPP) must be in that case known and it may vary between different *Bluetooth* devices. In ASCII interface the channel for SPP is always 1.

7.5 CLOSE

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

Synopsis:
CLOSE {*link_id*} *or* **CL {*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.

7.5.1 Examples

Closing active connections.

```
LIST

LIST 2
LIST 0 CONNECTED RFCOMM 668 0 0 32 8d 8d 00:07:80:80:38:77 1
OUTGOING ACTIVE MASTER ENCRYPTED
LIST 1 CONNECTED RFCOMM 668 0 0 7 8d 8d 00:07:80:80:36:85 1
OUTGOING ACTIVE MASTER ENCRYPTED
CLOSE 0
```

```
NO CARRIER 0 ERROR 0
LIST

LIST 1
LIST 1 CONNECTED RFCOMM 668 0 0 18 8d 8d 00:07:80:80:36:85 1
OUTGOING ACTIVE MASTER ENCRYPTED
```

7.6 NAME

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

Synopsis:
NAME {*addr*} *or* **N {addr}**

Description:	
addr	Address of the Bluetooth device.

Response:
No response

Events:
NAME event is delivered when friendly name is known. **NAME ERROR** event is delivered if friendly name lookup fails.

7.6.1 Examples

Successful name query.

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

Unsuccessfull name query.

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

7.7 RESET

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

Synopsis:
RESET

Response:
No response

Events:
None

7.8 SELECT

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

Synopsis:
SELECT {*link_id*} *or* **SEL {*link_id*}**

Description:	
link_id	Numeric connection identifier

Response:
No response. ASCII Interface goes to data mode with the link ***link_id***.

Events:
None

7.8.1 Examples

Changing between links

```
LIST

LIST 2
LIST 0 CONNECTED RFCOMM 668 0 0 243 8d 8d 00:07:80:80:38:77 1
OUTGOING ACTIVE MASTER ENCRYPTED
LIST 1 CONNECTED RFCOMM 668 0 0 419 8d 8d 00:07:80:80:36:85 1
OUTGOING ACTIVE MASTER ENCRYPTED
SELECT 1     (Transition to DATA mode – Device: 00:07:80:80:36:85)
```

7.9 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

7.9.1 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 BDADDR {*friendly_name*}

friendly_name

Friendly name of local device

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 *'

7.9.2 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)

7.10 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

8. ASCII INTERFACE 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.

8.1 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**

8.2 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**

8.3 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**

8.4 READY

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

Synopsis:
READY.

8.5 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**

8.6 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**

8.7 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 device
channel	Local RFCOMM channel

See also: **INQUIRY, NAME**

8.8 SYNTAX ERROR

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

Synopsis:
SYNTAX ERROR

9. ASCII INTERFACE ERROR MESSAGES

This chapter briefly presents the ASCII interface's error messages.

9.1 HCI errors

HCI errors start with code: ***0x100***

ERROR MESSAGE	CODE
HCI_SUCCESS	0x00
HCI_ERROR_ILLEGAL_COMMAND	0x01
HCI_ERROR_NO_CONNECTION	0x02
HCI_ERROR_HARDWARE_FAIL	0x03
HCI_ERROR_PAGE_TIMEOUT	0x04
HCI_ERROR_AUTH_FAIL	0x05
HCI_ERROR_KEY_MISSING	0x06
HCI_ERROR_MEMORY_FULL	0x07
HCI_ERROR_CONN_TIMEOUT	0x08
HCI_ERROR_MAX_NR_OF_CONNS	0x09
HCI_ERROR_MAX_NR_OF_SCO	0x0a
HCI_ERROR_MAX_NR_OF_ACL	0x0b
HCI_ERROR_COMMAND_DISALLOWED	0x0c
HCI_ERROR_REJ_BY_REMOTE_NO_RES	0x0d
HCI_ERROR_REJ_BY_REMOTE_SEC	0x0e
HCI_ERROR_REJ_BY_REMOTE_PERS	0x0f

HCI_ERROR_HOST_TIMEOUT	0x10
HCI_ERROR_UNSUPPORTED_FEATURE	0x11
HCI_ERROR_ILLEGAL_FORMAT	0x12
HCI_ERROR_OETC_USER	0x13
HCI_ERROR_OETC_LOW_RESOURCE	0x14
HCI_ERROR_OETC_POWERING_OFF	0x15
HCI_ERROR_CONN_TERM_LOCAL_HOST	0x16
HCI_ERROR_AUTH_REPEATED	0x17
HCI_ERROR_PAIRING_NOT_ALLOWED	0x18
HCI_ERROR_UNKNOWN_LMP_PDU	0x19
HCI_ERROR_UNSUPPORTED_REM_FEATURE	0x1a
HCI_ERROR_SCO_OFFSET_REJECTED	0x1b
HCI_ERROR_SCO_INTERVAL_REJECTED	0x1c
HCI_ERROR_SCO_AIR_MODE_REJECTED	0x1d
HCI_ERROR_INVALID_LMP_PARAMETERS	0x1e
HCI_ERROR_UNSPECIFIED	0x1f
HCI_ERROR_UNSUPP_LMP_PARAM	0x20
HCI_ERROR_ROLE_CHANGE_NOT_ALLOWED	0x21
HCI_ERROR_LMP_RESPONSE_TIMEOUT	0x22
HCI_ERROR_LMP_TRANSACTION_COLLISION	0x23
HCI_ERROR_LMP_PDU_NOT_ALLOWED	0x24

HCI_ERROR_ENC_MODE_NOT_ACCEPTABLE	0x25
HCI_ERROR_UNIT_KEY_USED	0x26
HCI_ERROR_QOS_NOT_SUPPORTED	0x27
HCI_ERROR_INSTANT_PASSED	0x28
HCI_ERROR_PAIR_UNIT_KEY_NO_SUPPORT	0x29
HCI_ERROR_CHANNEL_CLASS_NO_SUPPORT	0x2e

Table 2: HCI errors

9.2 L2CAP errors

L2CAP errors start with code: ***0x200***

9.3 SDP errors

SDP errors start with code: ***0x300***

ERROR MESSAGE	CODE
SDC_OK	0x00
SDC_OPEN_SEARCH_BUSY	0x01
SDC_OPEN_SEARCH_FAILED	0x02
SDC_OPEN_SEARCH_OPEN	0x03
SDC_OPEN_DISCONNECTED	0x04
SDC_NO_RESPONSE_DATA	0x11
SDC_ERROR_RESPONSE_PDU	0x10
SDC_CON_DISCONNECTED	0x12
SDC_CONNECTION_ERROR	0x13
SDC_CONFIGURE_ERROR	0x14
SDC_SEARCH_DATA_ERROR	0x15
SDC_DATA_CFM_ERROR	0x16
SDC_SEARCH_BUSY	0x17
SDC_RESPONSE_PDU_HEADER_ERROR	0x18
SDC_RESPONSE_PDU_SIZE_ERROR	0x19
SDC_RESPONSE_TIMEOUT_ERROR	0x1a

SDC_SEARCH_SIZE_TOO_BIG	0x1b
SDC_RESPONSE_OUT_OF_MEMORY	0x1c
SDC_RESPONSE_TERMINATED	0x1d

Table 3: SDP errors

9.4 RFCOMM errors

RFCOMM errors start with code: ***0x400***

ERROR MESSAGE	CODE
RFC_OK	0x00
RFC_CONNECTION_PENDING	0x01
RFC_CONNECTION_REJ_PSM	0x02
RFC_CONNECTION_REJ_SECURITY	0x03
RFC_CONNECTION_REJ_RESOURCES	0x04
RFC_CONNECTION_REJ_NOT_READY	0x05
RFC_CONNECTION_FAILED	0x06
RFC_CONNECTION_TIMEOUT	0x07
RFC_NORMAL_DISCONNECT	0x08
RFC_ABNORMAL_DISCONNECT	0x09
RFC_CONFIG_UNACCEPTABLE	0x0a
RFC_CONFIG_REJECTED	0x0b
RFC_CONFIG_INVALID_CID	0x0c
RFC_CONFIG_UNKNOWN	0x0d
RFC_CONFIG_REJECTED_LOCALLY	0x0e
RFC_CONFIG_TIMEOUT	0x0f
RFC_REMOTE_REFUSAL	0x11
RFC_RACE_CONDITION_DETECTED	0x12

RFC_INSUFFICIENT_RESOURCES	0x13
RFC_CANNOT_CHANGE_FLOW_CONTROL_MECHANISM	0x14
RFC_DLC_ALREADY_EXISTS	0x15
RFC_DLC_REJ_SECURITY	0x16
RFC_GENERIC_REFUSAL	0x1f
RFC_UNEXPECTED_PRIMITIVE	0x20
RFC_INVALID_SERVER_CHANNEL	0x21
RFC_UNKNOWN_MUX_ID	0x22
RFC_LOCAL_ENTITY_TERMINATED_CONNECTION	0x23
RFC_UNKNOWN_PRIMITIVE	0x24
RFC_MAX_PAYLOAD_EXCEEDED	0x25
RFC_INCONSISTENT_PARAMETERS	0x26
RFC_INSUFFICIENT_CREDITS	0x27
RFC_CREDIT_FLOW_CONTROL_PROTOCOL_VIOLATION	0x28
RFC_RES_ACK_TIMEOUT	0x30

Table 4: RFCOMM errors

10. CHANGING PARAMETERS WITH PSTOOL

Note:

Remember to follow these instructions carefully! Otherwise you might render your module to a state where you need to use SPI interface to change it back to virtual machine mode. Make sure you have installed ***BlueSuite***, so you have ***PsTool*** software available.

If you want to change settings, for example, of the UART baud rate, do as follows (this is shown in pictures in Appendix IV):

1) Type "**BCSP_ENABLE**" in the ASCII Interface
2) Close terminal software
3) Start ***PsTool*** software
4) Select BCSP as the protocol and select baud appropriately and press "**OK**"
5) With ***PsTool*** you can find the UART settings near the bottom of the list (they are alphabetically listed)
6) Change your settings (Press **Set** button after each change!)
7) Select ***View*** -> ***Programmer ID's***
8) Change the following settings to access ASCII interface again
    a. **PSKEY_UART_CONFIG** to **168** (Press **Set** button)
    b. **PSKEY_HOST_INTERFACE to vm_access_to_uart** (or 4) (Press **Set** button)
    c. **PSKEY_VM_DISABLE** to **False** (Press **Set** button)
9) Close ***PsTool***, start Terminal software and Reset the Evaluation Kit

WARNING!

It's absolutely important to change the parameters described in step 8. If you do not change these parameters, you will not be able to use ASCII interface again. On the other hand, if you change only one or two of these parameters, you will not be able to use either ASCII interface or PSTool again.

11. TROUBLESHOOTING

11.1 I get no response from the ASCII Command Interface?

Make sure your terminal settings are correct. Use ***PsTool*** to check the UART settings from the WRAP THOR Bluetooth module and make similar settings into your terminal software.

Check also your ECHO MODE settings. If you have set ECHO MODE to 0, you should not be able to see any responses.

11.2 I can connect only to two devices at a time?

Only two connections at a time are supported.

11.3 I changed 'UART Baud rate' key, but it didn't seem to work?

UART baud rate is stored now into user keys instead of '***UART baud rate'*** key. Delete '***User configuration data 26***' in order to return back to default settings ***115200,8n1***.

12. KNOWN ISSUES

1. Only two connections at a time are supported.

2. ASCII interface 2.0 is not visible in INQUIRY if there is a connection

3. You can not form two connections from one ASCII interface into one device. This is a property of CSR's *Bluetooth* stack

4. At this time you can NOT connect form ASCII interface to Nokia mobile phones. The connection works vice versa though.

5. At this time you can not open connection form ASCII interface and than wait an other device to connect ASCII interface – you have to form both the connections form a single ASCII interface module

13. SUPPORT

Contact Bluegiga Technologies: support@bluegiga.com

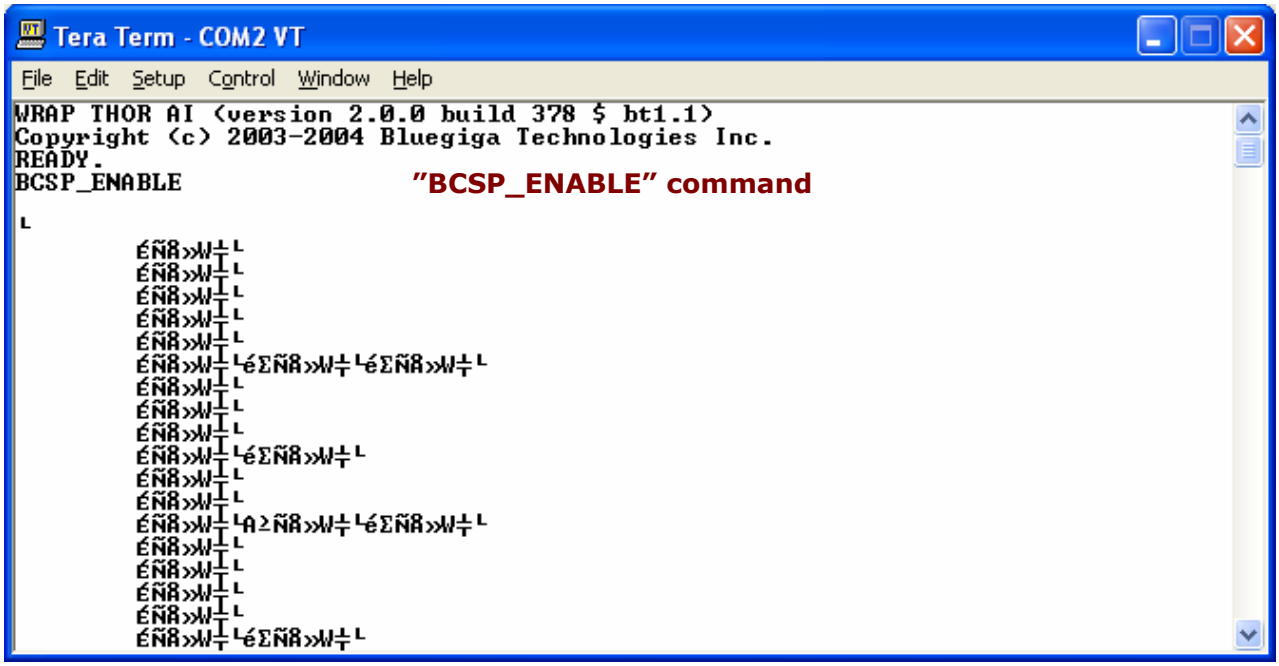
14. APPENDIX I – CHANGING PARAMETERS WITH PSTOOL

Initial mode: Command

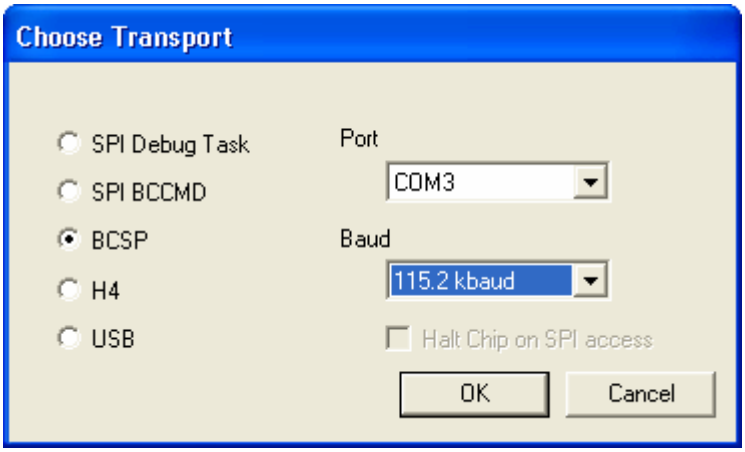
State transition: Command

Description: Changing the settings of the persistence store with ASCII Interface and PsTool software of the BlueSuite.

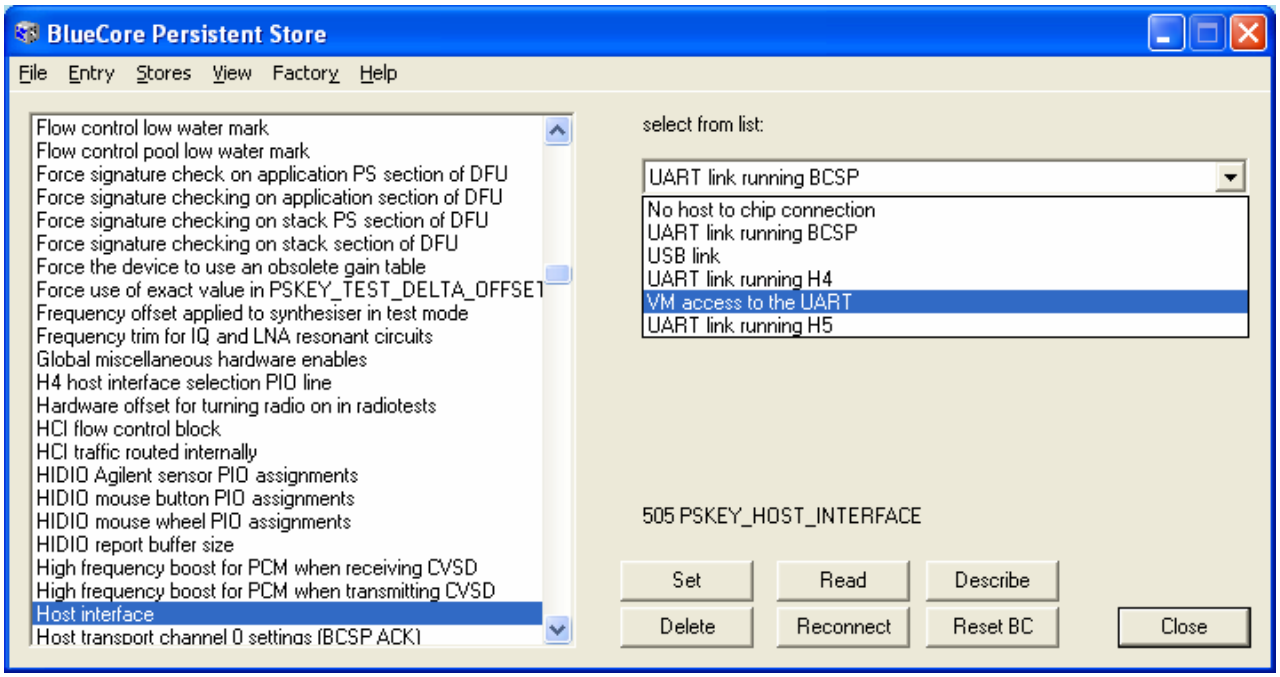
1. First issue the '**BCSP_ENABLE'** command when connected to the ASCII interface



2. Than start the PSTool software, select BCSP, COM port and your current baud rate. Then Press OK.



3. Change the settings you want.



4. To make transition from configuration mode back to ASCII Interface perform the following steps:

    a. Change the **'Host interface'** to **'VM access to the UART'**. Press **SET** button.

    b. Change **'VM disable'** to **'False'**. Press **Set** button.

    c. Change '**UART Configuration bitfield'** to **'168'**. Press **SET** button. (**'00a8'** in PSTool 1.20)

5. Close the PSTool software and start your terminal software and you are ready to use ASCII interface again.

WARNING!

If you do not change the mentioned settings and values back you may not be able to use the ASCII interface again. You also may not be able to communicate with the module and the PSTool software. In this case the module has to be accessed with the SPI cable (Onboard Installation Kit) to change the settings.