User Guide

BL65x AT Interface Application

v4.5

The scope of this guide relates to a general purpose smartBASIC application for Laird Connectivity's BL652, BL653 and BL654 Bluetooth Low Energy modules that allows the module to advertise, scan, and connect. It also offers GATT client and server capabilities all in a single application controlled with the industry standard AT command protocol over a UART interface.

This guide also provides a Virtual Serial Port interface that bridges the UART to a service that provides a streaming connection.

Revision History

Version	Date	Notes	Contributor(s)	Approver
1.0	20 June 2017	Initial Release		Mahendra Tailor
1.1	2 Aug 2017	Fixes to populate commands into the TOC		Mahendra Tailor
2.0	25 June 2018	Bluetooth v5 enhancements + DLE + NFC	Mahendra Tailor	Mahendra Tailor
3.0	10 Mar 2020	Extended Adverts commands	Mahendra Tailor	Dave Drogowski
3.1	11 Aug 2020	Indicated that the advProp parameter is mandatory on the AT+EADV command; Added the following S-Registers: 117, 118, 126, 138, 139, 219, 307; Added the following commands: AT+LMTU, AT+OOBR, AT+OOBL, AT+PKEY	Jamie Mccrae Greg Leach Jennifer Gibbs	Jonathan Kaye
4.0	5 Jan 2021	Localized to BL654. Added the following S-Registers: 127 Updated the following S-Registers with different usage: 307 Added the following unsolicited responses: LL, OB, PI, PL, SR Added the following commands: AT+ADAD, AT+BOVR, AT+CSEC, AT+PCFG, AT+PCNF, AT+TXPO Added the following error codes: 67, 68 Split unsolicited response showcode into: showcode and comparecode Added connection handles to unsolicited responses: comparecode, lescoob, oobkey , passkey, showcode, xxkey Removed state flow diagrams	Jamie Mccrae Greg Leach Jennifer Gibbs	Jonathan Kaye
4.1	27 Jan 2021	Added BL653/BL653µ, added note about BL654/PA power cap	Kieran Mackey	Dave Drogowski
4.2	11 Mar 2021	Added BL652	Kieran Mackey	Dave Drogowski
4.3	28 Feb 2022	Formatting updated with spaces after commas	Jamie Mccrae	Dave Drogowski
4.4	28 Jun 2022	Added description and error codes to NOCARRIER, fixed wrong NOCARRIER entry for OL (out of band key)	Jamie Mccrae	Dave Drogowski
4.5	19 Sep 2022	Added AT+BNDL command Added description for S-Registers 137, 138, 139 in 3.3.54 ATS	Erik Lins	Dave Drogowski

CONTENTS

1 OVERVIEW

This document is a user guide for the AT Interface *smart*BASIC application written for the BL65X Bluetooth Low Energy modules that exposes an industry standard AT command/response protocol. This protocol instructs the module to advertise, scan, connect, and pair. In addition, the *smart*BASIC application exposes AT commands that enable the creation of a GATT server table on-the-fly and, conversely, enables it to be a GATT client to interact with remote GATT servers. It also provides commands to read and write to GPIO pins.

The AT Interface *smart*BASIC application is a collection of many stand-alone applications that are made available in GitHub repositories. This collection is referred to in the singular term *AT Interface app* in the rest of this document. All the variants are available by following one these links depending on the module you have selected.

- https://github.com/LairdCP/BL652-Applications
- https://github.com/LairdCP/BL653-Applications
- https://github.com/LairdCP/BL654-Applications

There are Low Power UART (lpuart) and NFC (nfc) variants for the application available based on your application requirements. The base application is called *$autorun$.AT.interface.PPPP.sb* where PPPP is the target module/device (e.g. BL654). The general form for the variants of the *smart*BASIC applications is *$autorun$.AT.interface.PPPP.FFFF.FFFF…..FFFF.sb* where FFFF is an added feature which can be zero or more of the following:

- **Nfc** – Near field communications extensions (type 2 tags)
- **Lpuart** – Low power UART operation (saving ~ 300 uA idle current)

The module may also operate as a single connection virtual serial port device for the transparent relay of data between a remote device and the module's UART. This occurs in a similar manner as the old AT modems used for data communications on telephone lines. Even the tried and tested same ATD command is used to establish a connection to a peer using the Bluetooth address as the identifier.

This application requires that:

- The BL652 is updated to version 28.11.8.0 or newer firmware
- The BL653 is updated to version 30.2.3.0 or newer firmware
- The BL654 is updated to version 29.5.7.2 or newer firmware
- The AT Interface smartBASIC application (or appropriate variant) is loaded to the module

Note: We encourage you to modify the application to alter the behaviour given the *smart*BASIC source code is supplied. If you think your changes are a useful addition/fix, feel free to submit a pull request via GitHub.

2 *SMART*BASIC APPLICATION LOADING INSTRUCTIONS

We assume that the modules can be connected to a PC's serial port (USB or otherwise) and at least RX, TX, CTS, RTS, and DTR pins are connected (where the DTR pin is connected to the nAutorun of the module). If an appropriate Laird Connectivity development kit is used (such as the DVK-BL654), then this is already implemented.

Download all the variants of the AT Interface sample applications from the GitHub repository.

UwTerminalX is used to download the *smart*BASIC application. We assume that the cross-compiler .exe is saved in the same folder as the UwTerminalX folder (although this isn't necessary, because it performs an online compilation if the cross-compiler cannot be located on the PC).

To load the *smart*BASIC application, follow these steps:

1. In UwTerminalX, de-assert DTR by unchecking the DTR box as shown in Figure 1.

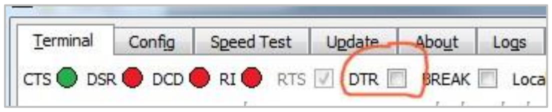


Figure 1: Uncheck DTR box

2. Tick and untick the BREAK checkbox. This results in a warm reset of the module. Given that DTR is no longer asserted, it does not launch into the *$autorun$* application if one exists in the file system.
3. Press **Enter** a few times until you get a `00` response.
4. To replace the *smart*BASIC application without disturbing any configuration settings or the trusted device bond database, enter the following command:

   `AT&F 1`

5. Wait for the following response:

   ```
   FFS Erased, Rebooting…
   OK
   ```

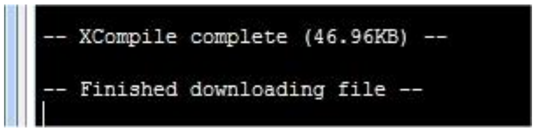
6. To erase all configurations, trusted device bond database, and the application, enter the following command:

   `AT&F *`

7. Wait for the following response:

   ```
   FFS Erased, Rebooting…
   OK
   ```

8. To reload the new application, right-click anywhere within the black area of the window and select **XCompile + Load** > **$autorun$.AT.interface.PPPP.FFFF.FFFF…..FFFF.sb**.
   Where PPPP and FFFF are as explained in the overview section and on completion the following displays (Figure 2):



```
-- XCompile complete (46.96KB) --
-- Finished downloading file --
```

Note: The size (in this case 46.96 KB) may be different as the application evolves.

Figure 2: Completed downloading file

9. Assert DTR by checking the DTR checkbox as shown in Figure 3.

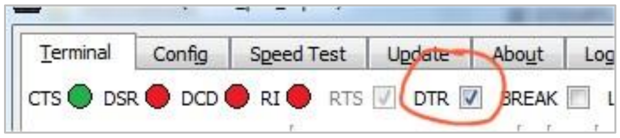


Figure 3: Reassert DTR

10. Tick and untick BREAK to warm reset the module. This causes the application to auto-start.

3 Operation

3.1 Modes

This application provides two mutually exclusive modes of operation on start-up depending on the value of S register 100 (referred to as VSP and non-VSP modes).

In VSP mode (the default), where bit 0 of S register 100 is set, the application initializes the GATT server table by populating it with the VSP service (and the mandatory GAP and GATT services). It then starts advertising to welcome incoming connections. Once connected, data to/from the VSP service is bridged to the UART. Because of this, once a connection is established, AT commands cannot be parsed. While there is no connection, AT commands are parsed and actioned (such as the ATD command to initiate an outgoing connection). The UART is bridged to the VSP service only on connection, either incoming or outgoing.

In the Non-VSP mode, where bit 0 is not set, the GATT table only contains the mandatory GAP and GATT services and many GATT server-related AT commands that are provided are used to add services and their characteristics. In addition, depending on the states of bits 1 & 2 of S register 100, the module will automatically start advertising and/or scanning. In this mode it is possible to start/stop advertising and scanning as required and to accept or make connections. Once connected, the AT parser is still active, so it is possible to restart adverts or make further connections. In connected states, it is possible to send GATT client-related commands to interact with a slave device. Note that in this mode virtually all commands are modal, meaning an OK or ERROR response is sent immediately after the command is processed, followed by one or more asynchronous messages. All the asynchronous responses start with a unique 2 letter sequence and so can be demultiplexed and actioned appropriately by the host.

Note: If scanning is enabled, the UART host should expect asynchronous (that is, arrive anytime) responses which contain advert reports. These could be intermixed with normal responses. For this reason, every response type in the Response section has a unique two letter (case-sensitive) start which allows the host to demultiplex them appropriately.

3.2 Data Flow Control

The host that is driving the UART interface must strictly adhere to RTS/CTS handshaking to ensure that data buffering and management are not compromised. If the module de-asserts its CTS line, the host stops sending data as soon as possible and conversely, if the host de-asserts its RTS line, the module stops sending data to it.

3.3 AT Commands

These are text commands starting with the character sequence AT and terminated by a \r character (ASCII code 0x0D). Commands are not case sensitive and have zero or more parameters. Multiple parameters are separated by the comma (,) character and some commands tolerate empty fields (two consecutive commas) and provide a default value. If more parameters are supplied than those specified, then the extra parameters are either silently ignored or result in a syntax error response.

The UART receive ring buffer has a default non-zero size which is at least 256 bytes long. It is imperative that any command, which is terminated by a \r is not larger than this. Otherwise, the system locks since the CTS is de-asserted and the host is unable to send the \r to empty the buffer. The size of the UART RX buffer can be modified using the S register 203.

The concept behind the application is that the host sends AT commands to perform various actions like advertise, scan, connect, pair, get local information, set configuration values, GATT read, and GATT write.

These AT commands are described in this section in alphabetical order. Also listed are responses to these commands which are described in the next chapter.

Many commands take parameters which are either integer values, strings or hex strings:

- Integer values – Can be entered as binary or in hexadecimal using the syntax 0xhh..hh.
- Hex strings – Only contain the letters 0-9, A-F and a-f and shall be exactly an even number long. Otherwise they are treated as syntax errors.

Note: In the following command, there is a space between the AT command and the parameters it accepts. That space is not mandatory; it's only used for visual clarity.

3.3.1 [Empty Line]

Command	`//Empty line with 0 or more whitespace//`
Possible Responses	`OK`

3.3.2 ^^^^

Command	**`^^^^`**
Description	When in a VSP connection and S register 109 is set to -1, a connection can be dropped by sending these four characters with intervening delays of at least the time specified in S register 210. Set to 250 milliseconds by default. The number of ^ characters required to trigger a disconnection is set via S register 111. You may want to increase it to ensure that the probability of unintended connection drop is lower than what it is when set to the default value of 4. The purpose of the intervening delay is to ensure that normal data transfer containing consecutive ^ characters does not induce a disconnection.
Possible Responses	`NOCARRIER`

3.3.3 AT

Command	**`AT`**
Description	No action performed other than to send the OK response
Possible Responses	`OK`

3.3.4 AT%S

Command

```
AT%S n="string"
AT%S n?
AT%S n=?
```

Description

These commands are to set, get, and get the range of valid lengths of the strings of any string valued S Register respectively where the S register is identified by the integer value n.

"string" – When setting, "string" is the new value; the double quotes are mandatory. To embed a non-printable character in the string, escape it using the three-character \hh sequence where hh is the ASCII value of the character in hexadecimal. For example, to filter the six-byte null terminated string "Hello", the string is entered as "Hello\00"

n? and **n=?** – For these variants, the returned value is enclosed in \n and \r and are sent before the OK. The two integer values returned by **n=?** are separated by a comma.

Note: When setting the value, it is not retained over a power cycle or a warm reset triggered using the ATZ command. See the AT&W command to make all changed values permanent.

The following S Registers are defined:

Register	Description
0	*Device Name* The length is between 1 and 12
1	*VSP Service Base 128-bit UUID* The length is exactly 32 characters, all hex digits.
2	*Scan Pattern* The length is between 0 and 20 This string specifies a pattern for filtering incoming advert report via scans. If the advert report contains at least one match, then it is reported to the host via the UART. For example, it can be set to the device name of a device and in that case, only that device's adverts are sent to the host. Use the three-character sequence \hh to enter a non-printable character in the string.

Possible Responses

3.3.5 AT&F

Command

```
AT&F
```

Description

Clear all S register settings back to defaults *and* clear the trusted device bond database and then perform a warm reset.

Possible Responses

```
OK (after the warm reset)
ERROR
```

3.3.6 AT&W

Command	**`AT&W`**
Description	Save all S registers to non-volatile memory.
Possible Responses	`OK` `ERROR`

3.3.7 AT+AARA

Command	**`AT+AARA tag, "payload"`**
Description	<tag> can take the range of 0..255 This command is used to add an AD element with tag and payload specified to the advert report cache variables for adverts that are used when operating in non-VSP mode. To add to the scan report, use AT+ASRA. This does not affect the adverts that are already committed to the radio and can be called multiple times to add more AD elements. To commit to the radio, use the AT+ACMT command.
Possible Responses	`OK` `ERROR`

3.3.8 AT+ACMT

Command	**`AT+ACMT`**
Description	The non-VSP advert and scan report caches created using AT+ARST, AT+AARA, and AT+ASRA are committed for transmission by the radio
Possible Responses	`OK` `ERROR`

3.3.9 AT+ADAD

Command	**`AT+ADAD`**
	`AT+ADAD <advset>`

Description	<advset> can take the range 0..n, where n is the maximum number of advert sets. Used to retrieve the advertising address of the specified advert set (or default if not provided). Device must be advertising to use this function, other

Possible Responses	`OK`
	`ERROR`
	`ADAD`

3.3.10 AT+ARST

Command	**`AT+ARST <connectable>`**
Description	<connectable> range of 0..1, if 0 then the advert report will not have the flags AD element. Used to clear the advert and scan report cache variables for adverts that are used when operating in non-VSP mode. This does not affect the adverts that are already committed to the radio.
Possible Responses	`OK` `ERROR`

3.3.11 AT+ASRA

Command	`AT+ASRA tag, "payload"`
Description	<tag> can take the range 0..255 Used to add an AD element with tag and payload specified to the scan report cache variables for adverts that are used when operating in non-VSP mode. To add to the advert report, use AT+AARA This does not affect the adverts that are already committed to the radio and can be called multiple times to add more AD elements. To commit to the radio, use the AT+ACMT command.
Possible Responses	`OK` `ERROR`

3.3.12 AT+BNDD

Command	`AT+BNDD address`
Description	Use this command to delete a device from the trusted device bond database.
Possible Responses	`OK` `ERROR`

3.3.13 AT+BNDP

Command	`AT+BNDP address`
Description	When a pairing is successful, the pairing keys and address are stored in the trusted device bond database and are marked as a rolling type. If the database is full, to guarantee storage of the newest pairing, the oldest rolling record is automatically deleted to make space. User this to change the type of record to persistent so it can only be deleted if explicitly done using the AT+BNDD command.
Possible Responses	`OK` `ERROR`

3.3.14 AT+BNDT

Command	**`AT+BNDT address`**

Description

Checks if a device identified by *address* (a 14-digit hex string) is present in the trusted device bond database (a result of a successful pairing).

The following response is sent before the OK if it is not trusted:

```
\n0\r
```

If trusted, the response is:

```
\n1, t, 14digithexaddr\r
```

…where *t* is 0 if the pairing is persistent and !0 if rolling.

Note: *Rolling* means that, at some point, it could be automatically deleted on a new pairing if the database is full.

14digithexaddr is the actual Bluetooth address of the device if the *address* passed to this command is a resolvable address.

At any time, the command ATI2009 returns the number of devices in the trusted device bond database.

Possible Responses

```
OK
ERROR
```

3.3.15 AT+BNDL

Command	**`AT+BNDL`**

Description

Used to show statistics of the trusted device bond database and list stored devices. First line of output shows max. capacity and number of rolling/persistent devices stored. Followed by the list of devices with an ID, MAC address and an extra information field. The extra information is an eight-digit hex number with the following bit values:

Bit	Description
0 to 15	Reserved for internal use
16	Set if IRK (Identity Resolving Key) exists for the trusted device.
17	Set if CSRK (Connection Signing Resolving Key) exists for the trusted device.
18	Set if LTK as Slave exists for the trusted device.
19	Set if LTK as Master exists for the trusted device.
20	Set if bond is in “rolling” database.

Possible Responses

```
OK
ERROR
```

3.3.16 AT+BNDX

Command	**`AT+BNDX`**
Description	Use this command to delete all devices from the trusted device bond database, both rolling and persistent types.

Possible Responses	`OK` `ERROR`

3.3.17 AT+BOVR

Command	**`AT+BOVR hIdx, accept`**
Description	Used to respond to a bond overwrite unsolicited response (OB) where hIdx is the connection handle and accept is 1 to accept the bond overwrite or 0 to decline it.
Possible Responses	`OK` `ERROR`

3.3.18 AT+CSEC

Command	**`AT+CSEC hIdx`**
Description	Queries the security of an active connection where hIdx is the connection handle, data is returned via the CSEC response if a valid connection handle was supplied.

Possible Responses	`OK`
	`ERROR`
	`CSEC`

3.3.19 AT+EADV

Command

`AT+EADV<advProp>, <advIntvlMs>, <maxCount>, <priSecPhy>, <peerAddr>, <chanMask>`

Description

Start normal or extended adverts which are non-VSP related. If the optional parameters are missing, then default values are used as follows (**Note that the advProp parameter is mandatory**):

- advIntvlMs: value from S register 208
- maxCount: 0
- priSecPhy: 0
- peerAddr: empty hex string
- chanMask: empty hex string

An example command is as follows:

```
AT+EADV3, 100, , , "123457890", ""
```

In this example, advProp is 3, advIntvlMs is 100, maxCount and priSecPhy are left default, peerAddr is 1234567890, and chanMask is default.

'advProp' is a bitmask where bit 0 is set for connectable adverts, bit 1 is set for scannable adverts, bit 2 for directed adverts and in that case 'peerAddr' must be a valid 14 hex digit string and bit 3 is set for extended adverts.

'advIntvlMs' is the interval in milliseconds for the repeated adverts and must be a minimum of 20ms and a maximum of 32 seconds

'maxCount' is a value in the range 0 to 255. If non-zero is specified, then the advertising will automatically stop after that many adverts have been sent and the "AE:" async response will be sent to the host.

'priSecPhy' specifies the PHY on which the primary and secondary packets are sent. Bit 0 is clear for primary adverts on 1MPHY and is set for primary adverts on LECODED. Bits 1, 2, and 3 specify the PHY on which the secondary packets are sent, where 000 means same PHY as primary, 001 means 1MPHY, 010 means 2MPHY and 011 means LECODED.

'peerAddr' is either empty (when bit 2 of 'advProp' is clear) or a 14-hex digit string which specifies the central device the advert is targeted at.

'chanMask' is either empty (which means use all channels) or a 10-hex digit string which specifies the value for a 5-byte binary string that denotes the 40 channels.

Note: These default values are cached on powerup/reset. If the S registers are changed, there must be an AT&W and then a reset.

If this command is received when in VSP mode, it exits to non-VSP mode and remains in that new mode.

When these adverts are received by a scanner running this application then they will appear as 'ADE' and 'ADS' response for advert and scan responses respectively.

Also see the AT+LADV command which is used to send normal 4.x adverts.

Possible Responses

```
OK
ERROR
```

3.3.20 AT+GCTM

Command **AT+GCTM hIdx**

Description

This is a GATT client-related command.

Use it to obtain the GATT table schema (such as the structure) of the peer connected on the handle identified by hIdx.

This results in many responses starting with either *TM:S* or *TM: C* and *TM: D*.

For example, the following from a device contains three services:

- First service – Contains four characteristics
- Second service – Contains one characteristic
- Third service – Contains four characteristics

In addition, the characteristic in the second service has a descriptor. In total, there are three descriptors in the entire GATT table.

```
AT+GCTM1
TM:S:1, (9), FE011800
TM: C:3, 00000002, FE012A00, 0
TM: C:5, 00000002, FE012A01, 0
TM: C:7, 00000002, FE012A04, 0
TM: C:9, 00000002, FE012AA6, 0
TM:S:10, (13), FE011801
TM: C:12, 00000020, FE012A05, 0
TM:  D:13, FE012902
TM:S:14, (65535), FD021101
TM: C:16, 00000010, FD022000, 0
TM:  D:17, FE012902
TM: C:19, 0000000C, FD022001, 0
TM: C:21, 00000010, FD022002, 0
TM:  D:22, FE012902
TM: C:24, 0000000C, FD022003, 0
OK
```

Where:

TM:S	Indicates the start of a BLE Service whose starting attribute handle is the integer value after the second ':' in that line.

The next integer parameter (in brackets)	The last attribute handle in that service.
Last eight-digit hex number	The UUID handle supplied by the firmware **Note:** *This is not the index mentioned in the AT+UUID command description.*

TM: C	Indicates the start of a BLE Characteristic
The integer after the second ':'	The handle for the value attribute
The next integer	Eight-digit hex value that denotes the characteristic properties (see command AT+GSCB for details)
The next eight-digit hex number	The UUID handle supplied by the firmware
The final decimal number	Is always 0. Intended as a place holder for the *Included Service UUID Handle*. **Note:** *We have not yet encountered an Included Service. We will add this functionality as needed.*

TM: D	Indicates the start of a BLE Descriptor that belongs to a Characteristic (such as CCCD)
The integer after the second colon (:)	Its attribute handle
Next hex number	The UUID handle supplied by the firmware. The last four digits of the UUID are the 16-bit adopted UUID if the first four digits are FE01. For example, if the last four digits are 2902, it is a CCCD. This means that you can use the attribute handle with the AT+GCWC command to write an enable/disable notify/indicates for the characteristic to which it belongs.

The host processing the TM responses know there are no more to come when it receives either an OK or ERROR message.

Possible Responses

```
OK
ERROR
TM:S
TM:  C
TM:   D
```

3.3.21 AT+GCFA

Command	**`AT+GCFA hIdx, uS, x, uC, y <,uD, z>`**
Description	This is a GATT client-related command.

Use this command to search for the handle of the value attribute of a Characteristic or the attribute handle of a descriptor attached to a characteristic in the peer connected on the handle identified by hIdx.

<uD, z>	(Optional) When this is absent, it implies that the search is for the value handle of a characteristic. When present, it implies that the search is for the descriptor. `OK` or `ERROR` terminates this command. If a characteristic or descriptor is found, the FC or FD responses have been received respectively.
uS uC uD	These are the UUID index that were used to pre-create a UUID handle using the command AT+UUID.
x y z	The 0-based instance index of the appropriate entity in the remote GATT table. For example, if x=1, y=2, and z=0, it means search for the second instance of a service with the UUID *uS*. In that service, search for the third instance of the characteristic with UUID *uC*; and in that characteristic, look for the first instance of the descriptor with UUID *uD*. **Note:** Typically, GATT tables do not have multiple instance services.

The main use of such a command is to locate a characteristic or descriptor in a server device to obtain the attribute handle so that it can be subsequently used in read/write requests using commands AT+GCRD, AT+GCWA, AT+GCWC.

This command immediately responds with OK or ERROR and, at some time subsequent, the asynchronous response FC or FD is received

When the attribute handle specified in the FC or FD is 0, it implies that the object was not found in the remote GATT table.

Possible Responses

```
OK
ERROR
FC
FD
```

3.3.22 AT+GCRD

Command	**`AT+GCRD hIdx, hAttr, nOffset`**
Description	This is a GATT client-related command.

It is used to read the content of a remote attribute starting at offset specified within that attribute. For example, if the attribute contains *Hello World,* setting nOffset to 6 results in *World* being read.

hIdx	The connection handle of the server from which it reads
hAttr	The attribute of the handle that was extracted using either AT+GCTM or AT+GCFA commands

This command immediately responds with OK or ERROR and at some time subsequent, the asynchronous response AR is received.

If the read was successful, then an AR response is received which contains the data. If the read failed (for example, if the attribute does not exist or it requires the connection to be authenticated), then the AS response is received. In rare occasions, an AB could also be received if, for example, the module is low in memory.

Possible Responses

```
OK
ERROR
AR
AS
AB
```

3.3.23 AT+GCWA

Command

`AT+GCWA hIdx, hAttr, hexDataString`

Description

This is a GATT client-related command.

It is used to write data to an attribute in a remote GATT table and expects an acknowledgement which will be received as an asynchronous response "AW" after the terminating "OK" response.

hIdx	The connection handle of the server from which it reads
hAttr	The attribute of the handle that was extracted using either AT+GCTM or AT+GCFA commands
hexDataString	A string consisting of only hexadecimal characters which must be an even number in length. It is converted to binary before writing to the peer.

It always writes to offset 0 in the destination attribute.

If the attribute rejects the write because say the connection is not encrypted, then the AW will have the appropriate status value.

Possible Responses

```
OK
ERROR
AW
```

3.3.24 AT+GCWC

Command

AT+GCWC hIdx, hAttr, hexDataString

Description

This is a GATT client-related command.

It is used to write data to an attribute in a remote GATT table; it does not expect an acknowledgement after the terminating OK response. If the command fails to write the value, then there will eventually be a disconnection because the link supervision timer will timeout.

hIdx	The connection handle of the server from which it reads
hAttr	The attribute of the handle that was extracted using either AT+GCTM or AT+GCFA commands
hexDataString	A string consisting of only hexadecimal characters which must be an even number in length. It is converted to binary before writing to the peer.

It always writes to offset 0 in the destination attribute.

If the attribute rejects the write because say the connection is not encrypted, then the AW will have the appropriate status value.

Possible Responses

```
OK
ERROR
```

3.3.25 AT+GSMD, AT+GSCB, AT+GSCE, AT+GSSB, AT+GSSE

Command

```
AT+GSMD m, rdRights, wrRights, len
AT+GSCB uC, prop, mVal <,mCccd<,mSccd>>
AT+GSCE hexDataString
AT+GSSB uS
AT+GSSE
```

Description

These are GATT server-related commands used to populate the local GATT server table with services, characteristics, and descriptors.

A characteristic can have properties like read/write and CCCD and/or SCCD descriptors which may or may not require authentication.

When adding a characteristic, those attributes must be specified. You can achieve this by using a metadata object which must be pre-created using the AT+GSMD command. Just like UUID handles management, this app provides for an array of metadata objects that are referenced using the index *m* in the range 0 to 3.

AT+GSMD is used to create a metadata object in array index *m* and creates an opaque integer value that contains the read and write which can be any one of these values:

0	No access
1	Open
2	Encrypted with no man-in-the-middle (MITM) protection
3	Encrypted with man-in-the-middle (MITM) protection

Once the metadata object is created its index can be used to refer to in any command (like AT+GSCB) that needs it.

AT+GSSB	Used to define the start of a service which has a UUID that was pre-created using the AT+UUID command
AT+GSSE	Used to define the end of a service so that a new Service can be added using AT+GSSB.
AT+GSCB	Used to define a characteristic which can have a CCCD and/or SCCD descriptors attached to it.

If the arguments mCccd and mSccd are not supplied, then the characteristic will have neither. To add a SCCD but not a CCCD, use the syntax **`,,mScc`** where the empty field between the two commands conveys that desire.

The parameter *uC* is the index of a UUID handle was pre-created using AT+UUID; and *prop* is a bit mask whose value is in the range 1 to 63 (0x3F) which are the properties as per the definition in the Bluetooth Specification. The following are the properties:

0	Broadcast-capable (Sccd descriptor must be present)
1	Can be read by the client
2	Can be written by the client without an ACK
3	Can be written (ACK is sent back)
4	Can be notifiable (Cccd descriptor must be present)
5	Can be indicatable (Cccd descriptor must be present)

AT+GSCE	Used to commit the new characteristic and *hexDataString* supplies the initial value (after conversion to binary). If CCCD or SCCD descriptors are specified, then the initial values are 0. This command responds with an integer value in the \nNN\r format (an integer value in the range 0 to N). This command will respond with an integer value in format "\nNN\r" which is an integer value in the range 0 to N. This integer value is an index value into an array of handles which MUST be noted by the host as associated with the newly created characteristic which is referenced in the commands AT+GSWC, AT+GSNO, and AT+GSIC. Think of this index value as an identifier.

Possible Responses

```
OK
ERROR
```

3.3.26 AT+GSIC

Command `AT+GSIC i, hexDataString`

Description

This is a GATT server-related command and is used to send a value indication if the client has enabled indications via the referenced characteristic's CCCD.

i	The characteristic identifier that was returned by the AT+GSCE command
hexDataString	The data that is first converted to binary and is then sent as an indication to all clients that enabled them.

When the indication is acknowledged by the client, it results in an asynchronous AK message.

Possible Responses

```
OK
ERROR
AK
```

3.3.27 AT+GSNO

Command `AT+GSNO i, hexDataString`

Description

This is a GATT server-related command and is used to send a value indication if the client has enabled indications via the referenced characteristic's CCCD.

i	The characteristic identifier that was returned by the AT+GSCE command
hexDataString	The data that is first converted to binary and is then sent as an indication to all clients that enabled them.

Possible Responses

```
OK
ERROR
```

3.3.28 AT+GSWC

Command `AT+GSWC i, hexDataString`

Description

This is a GATT server-related command and is used to set a new value for the characteristic identified by 'i'. If the characteristic is created with a property bit set for readable, then a remote GATT client is able to read this new value when it next polls it.

	'i' refers to the characteristic identifier that was returned by the AT+GSCE command and hexDataString is the data that is first converted to binary and then sent as an identification to ALL the clients that have enabled them.
Possible Responses	`OK` `ERROR`

3.3.29 AT+LADV

Command	**`AT+LADV <advType <,advIntvlMs>>`**
Description	Start adverts which are non-VSP related. If the optional parameters are missing, then default values are used. S register 108 is used for the advType and S register 208 for advIntvlMs. **Note:** These default values are cached on powerup/reset. If the S registers are changed, there must be an AT&W and then a reset. If this command is received when in VSP mode, it exits to non-VSP mode and remains in that new mode. Also see the AT+EADV command which is used to send extended adverts, for example LECODED.
Possible Responses	`OK` `ERROR`

3.3.30 AT+LADVX

Command	**`AT+LADVX`**
Description	Stop all adverts. **Note:** An incoming connection is established, then adverts are automatically stopped and a new AT+LADV command is required to restart adverts. At any time, use the command Ati2016 to determine the current status of advertising. Bit 0 is set if the module is advertising.
Possible Responses	`OK` `ERROR`

3.3.31 AT+LCON

Command	**`AT+LCON <L> address`**
Description	Make a non-VSP connection, with a slave latency of 0, to the device identified by *address* which is a 14-digit hex string (such as *000016A40B1623*). To make a VSP connection, use the ATD command. If 'L' is present, then the connection attempt will be on LECODED PHY On connection, the *connect* response contains parameters (which are detailed in the next chapter which describes all responses) but, because there can be multiple non-VSP connections, the parameters must be identified. A number between 1 and N is provided in that response so that it can be subsequently used to interact with the device on that connection. **Note:** The handle is 1 and above. 0 is used internally for special use to identify the one VSP connection that is possible. It uses the following S reg values to expedite the connection: ▪ 300 – Minimum connection interval ▪ 301 – Maximum connection interval ▪ 206 – Link supervision timeout ▪ 110 – Connection timeout (wait this long for the peer to accept) To change these values prior to initiating a connection, use the command ATSxxx=yyyy which is also described in this guide. The AT parser is then suspended until either a *connect*, *discon*, or *ERROR* response is sent. For example, if the address specified is not exactly a 14-digit hexstring then the ERROR response is sent.

Possible Responses	`connect…`
	`discon`
	`ERROR`

3.3.32 AT+LDSC

Command	**`AT+LDSC hIdx`**
Description	Use this command when in non-VSP mode to drop a connection (identified by the integer *hIdx*, that was supplied in the *connect* response). It is a value in the range 1 to N and initiates a disconnection. Later, after an OK response is sent, the actual disconnection occurs. At that time the discon message is sent.
Possible Responses	`OK` `ERROR`

3.3.33 AT+LENC

Command	**`AT+LENC hIdx`**
Description	Use this command when in non-VSP mode to encrypt a connection (identified by the integer *hIdx* that was supplied in the *connect* response). It is a value in the range 1 to N and initiates the negotiation with the peer for the connection to go encrypted. Later, after an OK response is sent, the *encrypt hIdx* message is sent.

Possible Responses	`OK`
	`encrypt hIdx`
	`ERROR`

3.3.34 AT+LMTU

Command	**`AT+LMTU hIdx`**
Description	This is a GATT client related non-VSP command, use this command to request the desired attribute MTU size from the remote GATT server. By default, the ATT_MTU size is 96 which is defined by #define DEFAULT_DLE_ATTRIBUTE_SIZE in $LIB$.defines.BL65x.sb and can be adjusted using S reg 219.

Possible Responses	`OK`
	`ERROR`
	`MT`

3.3.35 AT+LSCN

Command	**`AT+LSCN <timeout_sec <,"escaped_pattern"<,rssi<,scanType>>>>`**
Description	Start scanning for adverts. All parameters are optional and, if missing, the default value for timeout is obtained from S register 106, and *escaped_pattern* is set to an empty string and RSSI is set to -128. If in VSP mode of operation and the timeout_sec is set to 0, then it exits from VSP operation mode into non-VSP mode. It stays in that mode, otherwise the AT parser is suspended for the timeout value specified while scanning is in progress. ‘scanType’ is a bitmask where bit 0 is set to scan for primary adverts on 1MPHY, bit 1 for primary adverts on LECODED (if both are set then scanning will happen on both PHYs). Bit 2 is set for extended scanning into the secondary channels (if bit 1 is set, then this is forced) and bit 3 is set for passive scanning otherwise clear for active scanning where scan request packets will be sent if an advert is scannable.

Possible Responses	`OK`
	`ERROR`
	`AD0:...`
	`AD1:...`

3.3.36 AT+LSCNX

Command	**`AT+LSCNX`**
Description	Stop scanning.
Possible Responses	`OK` `ERROR`

3.3.37 AT+LPHY

Command	**`AT+LPHY hIdx`**
Description	This is a GAP related non-VSP command, use this command to request the desired PHY from the remote device as per the start-up flags, which can be adjusted using S reg 100.

Possible Responses	`OK`
	`ERROR`
	`PU`
	`PF`

3.3.38 AT+LVSP

Command	**`AT+LVSP`**
Description	When in non-VSP mode, this command sets the module into VSP mode. This means if the VSP service is not already installed in the GATT table, it will be installed.
Possible Responses	`OK` `ERROR`

3.3.39 AT+OOBL

Command	**`AT+OOBL`**
Description	Used to retrieve the local LESC OOB data from the underlying stack to pass to the remote radio via OOB means.

Possible Responses	`OL:local_address, oob_hash, oob_rand`
	`ERROR`

3.3.40 AT+OOBR

Command	**`AT+OOBR remote_address, oob_hash, oob_rand`**
Description	Used to submit remote address, OOB_hash, and OOB_rand to the underlying stack for use in future LESC OOB pairing. These values should be received from the remote device via OOB means.
Possible Responses	`OK` `ERROR`

3.3.41AT+PAIR

Command	**`AT+PAIR hIdx`**
Description	Use this command when in non-VSP mode to initiate a pairing with the device on the connection identified by the index handle *hIdx*. Later, if OK is sent and if pairing is successful, then the asynchronous response *encrypt* is sent. Also, if the pairing i/o capability S register 107 is not JustWorks, then there are other intervening responses related to authentication which require a response, such as: ▪ showcode hIdx code ▪ comparecode hIdx code ▪ passkey? hIdx ▪ oobkey? hIdx ▪ lescoob? hIdx ▪ xxkey? hIdx The response commands to these asynchronous responses are detailed in the later section related to responses. Because this is all event-driven using responses, it is sufficient to act accordingly when the events happen as detailed. At any time, the command ATI2009 returns the number of devices in the trusted device bond database. In VSP mode, if via S register 102 an encrypted VSP connection was enforced, and S107 is set to 0 (i.e. JustWorks) and the device is not trusted, then it allows a pairing during the connection initiated by ATD.
Possible Responses	`OK` `ERROR`

3.3.42 AT+PCFG

Command	**`AT+PCFG mode`**
Description	Changes the setting of the pair confirmation requests which is a bitmask value.

Mode bitmask values are as follows:

Bit	Value	Description
	0	No confirmation, always accept automatically
0	0x1	Confirm if existing bond exists with device (central only)
1	0x2	Confirm if existing bond exists with device (peripheral only)
2	0x4	Confirm if key exchange less than requested (central only)
3	0x8	Confirm if key exchange different than requested (central only)
4	0x10	Confirm if LESC is preferred but remote device does not support it
5	0x20	Confirm if MITM is preferred but remote device does not support it
6	0x40	Confirm if OOB is preferred but remote device does not support it
7	0x80	Confirm if bond is preferred but remote device does not support it (central only)
8	0x100	Confirm if bond is preferred but remote device did not request it (peripheral only)
9	0x200	Confirm if remote device maximum encryption key size is less than the local device maximum encryption key size
10	0x400	Confirm if supplied pairing address is different from connected address and there is a bond entry for the supplied address (Note that this will trigger an 'OB' event rather than a 'CP' event)
11	0x800	Modifies existing bond confirmation (bits 0 and 1) so that no confirmation will not be thrown if the device pairing/bonding is using the bond details that already exist in the bond database
12-14		*Reserved for future use*
15	0x8000	Always confirm

Where mode is an integer value in the range 0 to 32768 (0x8000)

Possible Responses

```
OK
ERROR
```

3.3.43 AT+PCNF

Command	**`AT+PCNF hIdx, accept`**
Description	Replies to an incoming pair confirmation event (CP) to either accept or decline it, where 'hIdx' is the connection and 'accept' is 0 to decline or 1 to accept.

Possible Responses

```
OK
ERROR
```

3.3.44 AT+PKEY

Command	**`AT+PKEY nnn`**
Description	If pairing I/O capability is appropriately set via S register 107 so that this module has keyboard capability for use in authenticated pairing, then this command can be used to issue a static passkey to the underlying stack for use during a pairing procedure in a future connection. It allows for a use case similar to what PIN codes provided in classic Bluetooth before simple secure pairing was introduced in v2.1. Where nnn is an integer value in the range 0 to 999999 **Note:** The pairing still uses LESC Diffie-Hellman based exchanges, but the only difference is that instead of a random number this static value is used. **Note:** Repeated pairing attempts using the same pre-programmed passkey makes pairing vulnerable to MITM attacks.
Possible Responses	`OK` `ERROR`

3.3.45 AT+PRSP

Command

```
AT+PRSP hIdx, [Y|y|N|n]
AT+PRSP hIdx, 32HexDigitNumber
AT+PRSP hIdx, nnn
```

Description

If pairing I/O capability is appropriately set via S register 107 so that this module has a user-interface to expedite an authenticated pairing (as opposed to Just Works), then during the pairing process (which is initiated by the AT+PAIR command), if the peer device also has pairing capability, then the variant of this command to use is as per the response as follows:

- *showcode hIdx code* :: AT+PRSP hIdx, [Y|y|N|n]
- *comparecode hIdx code* :: AT+PRSP hIdx, [Y|y|N|n]
- *passkey? hIdx* :: AT+PRSP hIdx, nnn
- *oobkey? hIdx* :: AT+PRSP hIdx, 32HexDigitNumber
- *lescoob*? hIdx :: AT+OOBR hIdx, oob_hash, oob_rand
- *xxkey? hIdx* :: AT+LDSC hIdx

Where hIdx is the same value as per supplied in the response from the module:

- [Y|y|N|n] – One of those four single characters to imply a Yes or No.
- Nnn – An integer value in the range 0 to 999999
- 32HexDigitNumber – A hexadecimal string consisting of exactly 32 characters.

If *xxkey?* was received which will be unexpected, then the best action is to disconnect and exists to future proof the device just in case a future Bluetooth specification adds a new type of pairing authentication mechanism.

Possible Responses

```
OK
ERROR
```

3.3.46 AT+SFMT

Command

```
AT+SFMT <frmt>
```

Description

<frmt> can take the range 0..1
When AT+LSCN is used to scan for adverts it will display each advert in a default format where only the device name from the advert data is displayed. That default format is specified by frmt=0 and will be the default value if <frmt> value is not provided.

If frmt=1 then the full advert/scan report data is displayed in hex format.

Please note that user is free and encouraged to add more formats

Possible Responses `OK`
`ERROR`

3.3.47 AT+SIOC

Command **`AT+SIOC sionum, func, subfunc`**

Description The module has many digital and analog I/O pins. They are referred to as Signal Input/Output (SIO for short) pins. They are configurable to be digital or analog and can be in or out and some can even have special functions like PWM (pulse width modulation) or even Frequency output.

Individual pins are configured using this command.

Sionum is a value in the range 1 to N which identifies the signal pin number

Func is as follows:

1	Digital_IN
2	Digital_OUT
3	ANALOG_IN

Subfunc are values that further qualify the *Func* value. Refer to the module user guide for additional details and description of the GPIOSETFUNC() function.

Possible Responses `OK`
`ERROR`

3.3.48 AT+SIOR

Command **`AT+SIOR sionum`**

Description Once a signal pin is configured using the AT+SIOC command, if it was configured as a digital_in or analog_in, the command retrieves the current value – 0 or 1 for digital and a 0 to N for analog.

The integer value returned has a starting \n character and ending \r character.

Possible Responses `OK`
`ERROR`

3.3.49 AT+SIOW

Command **`AT+SIOW sionum, val`**

Description Once a signal pin is configured using the AT+SIOC command, if it was configured as a digital_out, this command sets the current value which will be 0 or 1.

Possible Responses `OK`
`ERROR`

3.3.50 AT+TXPO

Command **`AT+TXPO`**
`AT+TXPO hIdx`

Description This can be used to query the default transmission power for connections or the active transmission power for an active connection (if provided), where hIdx is an optional parameter and specifies the connection handle to query the active transmission power for.

Possible Responses	`OK`
	`ERROR`
	`TXPO`

3.3.51 AT+UUID

Command	**`AT+UUID u, 16bitUuid`**
	`AT+UUID u, 32HexDigitNumber`
	`AT+UUID u, 16bitUuid, v`

Description	*BLE makes wide use of UUIDs (universally unique identifiers) which are 128-bit (16-byte) random values. These values can be cumbersome to manage as string objects and so the module firmware exposes a concept of a 32-bit integer value which is a handle to an internal 16-byte buffer that contains the actual value.* *The smartBASIC application exposing the AT interface functionality extends that concept by using an array of integer variables to store those handles provided by the firmware. Those firmware handles are never exposed, but instead an index value* 'u' *is.* The 'u' in these three variants of the command is the index into that integer array. Think of there being a bunch of mailboxes numbered 0 to N (see MAX_UUID_HANDLES in the source code) which are your scratchpads to load UUID handles into (using these commands) as and when you need to supply a UUID into any of the AT commands that require a UUID. For example, the command AT+GSSB takes a parameter which is one of these 0 to N indices. The value for 'u' shall always be in the range 0 to N, where N is 15 at the time of writing and can be modified by changing the #define for MAX_UUID_HANDLES. The command variant "AT+UUID u, 16bitUuid" is used to create a handle from a Bluetooth SIG adopted 16-bit UUID and store it in the array index 'u'. The value 16bitUuid shall be in the range 0 to 0xFFFF. The command variant "AT+UUID u, 32HexDigitNumber" takes the 32-character hexadecimal string and converts that into a handle and stores it in the array index 'u'. The command variant "AT+UUID u, 16bitUuid, v" takes the '16bitUuid' which is a value in the range 0 to 0xFFFF and creates a sibling of the handle stored in array index v and stores in array index 'u'. By sibling, it is meant that the base UUID of the handle stored in array index 'v' is used to create the new UUID.
Possible Responses	`OK` `ERROR`

3.3.52 ATD

Command	**`ATD <L> address`**
Description	Make a VSP connection, with a slave latency of 0, to the device identified by 'address' which is a 14-digit hex string like for example 000016A40B1623. To make a non-VSP connection use command AT+LCON.

If 'L' is present, then the connection attempt will be on LECODED PHY

It uses the following S register values to expedite the connection:

300	Minimum connection interval
301	Maximum connection interval
206	Link supervision timeout
110	Connection timeout (wait this long for peer to accept)

To change these values prior to initiating a connection use the command ATSxxx=yyyy which is also described in this guide.

Then the AT parser is suspended, until either a "CONNECT" or a "NOCARRIER" response is sent. Please note this is one of the few commands that is NOT terminated by an OK or ERROR response.

If for example, the address specified is not exactly a 14-digit hex string then the NOCARRIER response will be sent.

Possible Responses

```
CONNECT…
NOCARRIER
```

3.3.53 ATI

Command	**`ATI n`**
Description	Get read-only information identified by integer n. A value is returned that could be an integer or a string that starts with a \n and ends with a \r.

The following information is returned with identifier 'n' as stated (refer to user guide for the module for more n values in the section related to the function SYSINFO):-

Reg	Description
0	The value of #define AT_RESPONSE_0 in the source code. E.g: "BL654"
3	The firmware version of the module (see n=23 for version of app)
4	The Bluetooth address of the module as a 14-digit hex string
10	The value of #define AT_RESPONSE_10 in the source code. E.g: "Laird Connectivity, (c) 2020"
11	Will return 1 if low power UART operation variant of this application has been loaded.
23	The version of the smartBASIC application, which is the value of #define LibVer
33	The version of the smartBASIC application, which is the value of #define AppVer and can be changed by the customer in top level .sb file
42	The value of the current state of a state machine implemented by the app.
50	Count of NFC Coil energise/deenergise events.

		Will be an odd number if the coil is still energised. When this count is read, all bits except bit 0 is reset. Wraps to 0 after 2^31
	51	Count of the number of times the NFC Tag has been read. When this count is read, it will be reset Wraps to 0 after 2^31
	2009	Number of devices in the trusted device bond database
	2012	Maximum number of devices that can be saved in the trusted device bond database
Possible Responses	`OK` `ERROR`	

3.3.54 ATS

Command

```
ATS n=m
ATS n?
ATS n=?
```

Description

These commands are to set and get values, as well as the range of valid values, for any S register (respectively) where the S register is identified by the integer value n. When setting, m is the new value.

For the ‘n?’ and ‘n=?’ variants the returned value will be enclosed in \n and \r and will be sent before the OK. The max and min integer values returned by ‘n=?’ are separated by a comma.

Note when setting the value that it will not be retained over a power cycle or a warm reset triggered using the ATZ command. See the AT&W command to make all changed values permanent.

The following S Registers are defined:

Register	Description
100	*Start-up Flags* ▪ Bit 0: Set to VSPConnectable - hence populates GATT table and starts adverts ▪ Bit 1: Ignored if bi t0 is 1 otherwise start advertising with no timeout ▪ Bit 2: Ignored if bit 0 is 1 otherwise start scanning with no timeout ▪ Bit 3: Set for max bidirectional throughput of about 127kbps, otherwise half that. ▪ Bit 4: Use Data Length Extension (#define DLE_ATTRIBUTE_SIZE) in smartBASIC application ▪ Bit 5: Phy Rate 00 – 1MPHY 01 – Long Range – 125kbps 10 – RFU : will set 1 MPHY 11 – 2MPHY
101	*TxPower_dBm* (see module user guide for valid values) Note for all BL654 AT applications, maximum settable TxPower is +18dBm but this will be capped to +8dBm on BL654 and +14dBm on BL654PA if LE Coded PHY is being used or if a remote central device has forced a channel map change.
102	*Encryption Requirement for incoming VSP connections* ▪ Bit 0: Enable(1)/Disable(0) ▪ Bit 1: (MITM(1) /NoMITM(0)

103	*Device Name Format in adverts and Gap Service (valid values 0 to 7)* If this value is 0 then the name will be “DEVNAME” which is specified using the string S Register command AT%Sn=s where n is 0 and the command AT%S is described elsewhere in this section. If this value is non-zero, then the name will be “DEVNAME-HH..HH” where the number of HH is exactly double the value in this register and those hex digits correspond to the rightmost hex characters of the Bluetooth address. For example, if the Bluetooth address is 0123456789ABCD and this register contains 3, then the device name is “LAIRD”, and the advertised device name will be “LAIRD-89ABCD”. Also note that if the combined string is greater than the value set for ‘#define MaxDevNameSize’ in the *smart*BASIC source code (which you are free to modify) then the name will be the rightmost that many characters.
104	This is the slave latency that will be negotiated when connected as a slave. This negotiation will start about 5 seconds after the connection is made.
105	*FlagsAD* This is the flags bit in the Flags AD element when adverts are started. It specifies general or limited discoverability. If not sure, use default value.
106	*Scan Timeout in seconds* When starting scans for adverts using the AT+LSCN command, if the timeout value is omitted then the value in this register will be used.
107	*I/O Capability to use during the initial negotiation when pairing.* This specifies the user interface that is available to expedite a pairing. ‘Just Works’ pairing implies there is no user interface and so the resulting encryption key will not be authenticated and so not immune to MITM (man-in-the-middle) attack. Valid values are- ▪ 0=Just Works ▪ 1=Disp with Y/N ▪ 2=Kboard only ▪ 3=Disp Only ▪ 4=Kboard+Disp
108	*Idle Advert Type* This specifies the advert type to use advertising in non-VSP mode. ▪ 0=ADV_IND (Connectable and will respond to scan requests) ▪ 1=ADV_DIRECT_IND (connectable but only from specific device) ▪ 2=ADV_SCAN_IND (Not Connectable, but responds to scan req’s) ▪ 3=ADV_NONCONN_IND (Not Connectable, ignores scan req’s) If this is changed, then a save using AT&W is required and will only take effect after the next power cycle or warm reset.
109	*Pin to use to control VSP command mode or low power UART operation.* If this is -1, then to drop a VSP connection, the ^^^^ escape sequence needs to be sent, otherwise the state of the pin is used to disconnect. That pin must be high to allow connection to continue. If there is an outgoing connection attempt this pin is low then the connection will not be allowed and similarly for incoming connection, on connection, this pin is low, then an immediate disconnection will be requested. For low power UART operation, this GPIO line is monitored and when low the module is allowed to automatically close the UART after and idle period that is set via SRegister 213

110	*Connection Timeout in seconds* When making an outgoing connection using the command ATD or AT+LCON, this Sreg specifies the maximum time for connectable adverts from the device to be connected to.
111	*Number of '^' characters to send over the UART to trigger a disconnect* If Sreg109 is -1 then multiple '^' character can be used, interspaced by delays' to disconnect when there is a VSP connection. The delay is specified by Sreg 210.
112	*Active or Passive Scan Type* Set to 0 for passive scanning and 1 for active. Active scanning means that if an advert is received with type ADV_IND or ADV_SCAN_IND the it will send a scan_request so that the advertiser sends a scan_response which contains a further 31 bytes made of AD elements. By default, this is set for active scanning.
113	*Scan RSSI minimum in dBm* When scanning for adverts, each incoming advert is reported with the RSSI that was received at. If the RSSI of that advert is less than specified by this S reg then it will not be reported to the host connected at the UART. This allows the host to filter adverts based on how weak the signal is (that is how far away it is usually). The default setting is -120 and so given that the receive sensitivity is around -100 it implies that all adverts no matter how weak, if received, will be reported to the host.
114	*Link Supervision Timeout (Seconds) as Slave* This is the link supervision timeout that will be requested for an incoming connection after 5 seconds if the connection interval is not in the required range. This value is written to the GAP service on power up.
115	*Minimum Encryption Key Length* This can be between 7 and 16. Essentially at pairing this information will be determined and saved in the trusted device bond database. In future if a service requires a minimum key length for data exchange, and the connection is encrypted, if the length of the key for that encryption in less that this value then data exchange cannot happen.
116	*MITM (man-in-the-middle) for Encryption Required* This is used by a central role device when it wishes to start encryption. If this set to 1, then it implies that the encryption request shall only succeed if the stored key was authenticated when the most recent pairing happened. Valid values are 0 for no MITM requirement and 1 for required.
117	*DCD Output for VSP Connection* Used to select pin for DCD output, LOW when there is a VSP connection. Default pin is used if not set.
118	*VSP Incoming Max Cached Packets in VSP RX Buffer* Default value is 8 See S reg 219 and 307 to adjust throughput performance
126	*Max Connections as Master/Central* Can be up to 16. Default is 8.

127	*BLE Transmission Buffers* Can be up to 6. Default value is 6.
137	*Polarity of Connection Indication Pin and Encrypted Connection Indication Pin* Select the polarity for pins specified in S registers 138 and 139. Bits 0-1 are being used as follows: Bit 0: connection indication pin (S register 138) 1=high active (default) 0=low active Bit 1: encrypted connection indication pin (S register 139) 1=high active (default) 0=low active Valid values are 0 to 3.
138	*Connection Indication Pin* Select the pin to use to indicate that a non-VSP connection is active. The pin will remain active as long as there is at least one non-VSP connection active. Polarity of the pin can be selected with S register 137.
139	*Encrypted Connection Indication Pin* Select the pin to use to indicate that a non-VSP connection has been encrypted. The pin will remain active as long as there is at least one encrypted non-VSP connection active. Polarity of the pin can be selected with S register 137.
1xx	*Free to be used.* Currently ‘xx’ is 19 to 25 and 28-36. Valid values are -128 to +127
200	*VSP Encryption Disconnect Timeout (milliseconds)* If a VSP service is specified with encryption requirement, then on a VSP connection a timer is started. If that timer times out before the connection goes encrypted, then the slave will initiate a disconnection. This is a form of resilience to a denial-of-service attack, in which a device just connects and then does nothing to prevent legitimate users from connecting. The timer is cancelled as soon as the connection goes encrypted.
201	*VSP Advert Interval (milliseconds)* When starting adverts for incoming VSP connections, this specifies the advert interval to use.
202	*UART Transmit Buffer Size*
203	*UART Receive Buffer Size* The size of the UART transmit and receive ring buffers. 0 means use default.
204	*VSP Transmit Buffer Size*
205	*VSP Receive Buffer Size* The size of the VSP transmit and receive ring buffers.
206	*Link Supervision Timeout in milliseconds* When making an outgoing connection using ATD or AT+LCON this S reg specifies the link supervision timeout to use in the connection request.
207	*Appearance (Optionally used in Adverts)* This specifies the value to use in the Appearance AD element in an advert. A value of 0 implies that the Appearance AD element will not be added to the advert report.

208	*Idle Advert Interval in milliseconds* When advertising in non-VSP mode, this specifies the default advert interval. Also used when not supplied in the AT+LADV command.
209	*GATT Client memory size* Use this to specify the memory the GATT client will reserve for itself when it is opened for any GATT client activity. Only modify this if memory becomes tight due to many smartBASIC variables being declared. Adjustment of this S reg will be rare.
210	*VSP Escape Character Minimum Inter-Character Spacing (milliseconds)* When ^ is used to drop a VSP connection, this specifies the minimum delay that has to exist between consecutive ^ characters for a disconnection to be triggered. This is so that normal data traffic containing a train of ^ characters does not induce a disconnection. See S reg 111 which is used to specify the number of consecutive ^ characters needed to trigger the disconnection.
211	*Scan Interval in milliseconds*
212	*Scan Window in milliseconds* When a scan for adverts in initiated these registers specifies the interval and window respectively, for scanning. The ratio of window over interval specifies the duty cycle. When both are set to the same value the duty cycle is 100% and so here is minimal probability that an advert report will be missed. However, setting 100% duty cycle implies the radio receiver is ON all the time and so will result in maximum power consumption. Setting the ratio as low as possible reduces power consumption but at the expense of missing adverts.
213	*UART Idle Time in milliseconds for low power UART operation* When the low power version of this application is loaded into the module, if the module detects that there is no UART activity for this period of time AND the ‘keep UART open’ input line is low, then it will automatically close the UART. If there is incoming data over the air that needs to be conveyed to the host, then the UART is automatically opened regardless of the status of the ‘keep UART open’ input line.
219	*DLE Attribute Size* Valid values 20…244, Default value is 96 set using #define DEFAULT_DLE_ATTRIBUTE_SIZE in LIB.defines.BL65x.sb See S reg 118 and 307 to adjust throughput performance
2xx	*Free to be used.* Currently ‘xx’ is 14 to 18. Valid values are -32768 to +32767
300	*Minimum Connection Interval in microseconds*
301	*Maximum Connection Interval in microseconds* When making an outgoing connection using ATD or AT+LCON, these specify the minimum and maximum intervals that is acceptable for the connection interval. A range needs to be specified to give the stack flexibility in arranging the optimal connection intervals when there are multiple connections. If you are going to only have a VSP connection and so know that the radio is not going ‘object’ it is possible to set both these values to the same value and in that case, you should get the value you require. When the connection is established it is reported using the CONNECT response which will supply the actual interval negotiated by the stack with the peer.

302	*UART Baud rate* This specifies the baud rate to use for commands and data transfer. After setting, a power cycle or a warm reset will be required.
303	*VSPTxUUID*
304	*VSPRxUUID*
305	*VSPMdmInUUID*
306	*VSPMdmOutUUID* These are values in the range 0x0 to 0xFFFF and are the 16-bit UUID offsets to use for the VSP service. Changing this will mean that mobile apps supplied by Laird Connectivity to interact with VSP will stop working as they will not find the expected UUIDs. Only change this if you really need to. A good reason would be to make the VSP private to you and so other devices expecting the standard Laird Connectivity UUID will not work and so yet another way to restrict access to your device.
307	*BLE Connection Event Length* Maximum number of packets that can be transmitted per connection per connection interval. Can be set up to 800 in all modules but is limited in BL654PA to 9 when TXPOWER is set to +18dBm. Default is 12.
3xx	*Free to be used.* Currently 'xx' is 08 to 09. Valid values are -2147483648 to +2147483647

Possible Responses	`OK` `ERROR`

3.3.55 ATZ

Command	`ATZ`
Description	Restart the module by performing a warm reset.
Possible Responses	`OK`

3.4 Responses

To simplify reception of messages in the receiving device, each message starts with a \n character and ends with a \r character, and may contain additional embedded \n characters, where \n is the linefeed character with ASCII code 0x0A and \r is the carriage return characters with ASCII code 0x0D.

After stripping the \n start character, each response will start with a unique 2-character sequence to help the host decode the response quicker in a stateless manner.

Some responses are synchronous which mean they are used to terminate a command so that the command parser can process more commands.

3.4.1 Response: Synchronous and Terminating

When a host receives these responses, it can issue new commands and expect them to be processed immediately.

3.4.1.1 CONNECT

```
CONNECT 0, address, interval, sprvsnTout, latency
```

The command ATD has successfully created a VSP connection to the device with Bluetooth ‘address’ where the connection interval is ‘interval’ which is in microseconds, ‘sprvsnTout’ is the link supervision timeout in microseconds and ‘latency’ is the slave latency.

The first parameter will always be 0 as that handle index is dedicated for VSP connections.

3.4.1.2 ERROR

`ERROR` **nn**

A command was not successfully actioned and ‘nn’ is an error code. Error Code are as follows:

01	Invalid S Reg number
02	Value supplied is out of range
05	Syntax Error
09	Invalid Address has been supplied
14	Command cannot be processed in current state
15	Unknown Command
33	Value supplied is not valid
46	GPIO specified is not available
47	Too few parameters supplied
48	Too many parameters supplied
49	Hex String is not valid
50	Save Fail
51	Restore Fail
52	VSP open fail
53	Invalid Advert Type
54	Invalid UUID
55	Service Not Ended
56	Characteristic Not Ended
57	Service Not Started
58	Too Many Characteristics
59	Characteristic Not Started
60	NFC Not Open
61	NFC NDEF Message Empty
62	Directed advert but peer address is missing
63	Invalid Channel Mask
64	Invalid Advert Reports
65	Invalid Advert Report Data
66	Invalid Advert Report Data Size
67	Invalid out of band (OOB) data
68	Newline character was expected but was not found
99	Functionality not coded

3.4.1.3 NOCARRIER

`NOCARRIER` i

The command ATD has failed to establish a VSP connection.

‘i’ is an integer number which is the error code describing the failure as per:

Value	Description
5	Authentication failure
6	PIN or link key missing
7	memory capacity exceeded
8	connection timed out
19	remote user terminated connection
20	remote user terminated connection due to low resources
21	remote user terminated connection due to power off
22	local user terminated connection
30	Invalid LMP parameters
31	Unspecified error
34	LMP response timed out
36	LMP PDU not allowed
58	Controller busy
59	Connection interval unacceptable
61	MIC failure
62	Connection failed to be established
80	BleConnect function returned an error
81	Invalid address
82	Command pin state
83	Too many connections
84	Timeout
85	Out of memory
86	Unencrypted
87	No VSP service
88	Expected pairing process input was not received
90	User disconnected
91	Authenticated link required

3.4.1.4 OK

`OK`

A command was successfully expedited.

3.4.2 Response: Synchronous & Not Terminating

When a host receives these responses, it cannot issue new commands and expects them to be processed immediately as a terminating response is still to come.

3.4.2.1 TM:S

TM:S:i, (j), HHHHHHHH

AT+GCTM command in progress and this specifies details of an attribute in a remote table that contains a Service attribute.

'i' is an integer number which is the attribute handle.

'j' is an integer number which is the last attribute handle in this service

'HHHHHHHH' is an 8-digit hex value corresponding to a UUID handle. If the first 4 HHHH is 'FE01' then it is a Bluetooth SIG adopted UUID and the 16-bit value is the next 4 digits.

3.4.2.2 TM: C

TM: C:i, 000000PP, HHHHHHHH, 0

AT+GCTM command in progress and this specifies details of an attribute in a remote table that contains a Characteristic attribute.

'i' is an integer number which is the handle for the value attribute.

'HHHHHHHH' is an 8-digit hex value corresponding to a UUID handle. If the first 4 HHHH is 'FE01' then it is a Bluetooth SIG adopted UUID and the 16-bit value is the next 4 digits.

The final '0' is for future use and is related to included services.

Note the one space between the first ':' and the 'C'

3.4.2.3 TM: D

TM: D:i, HHHHHHHH

AT+GCTM command in progress and this specifies details of an attribute in a remote table that contains a Descriptor attribute.

'i' is an integer number which is the attribute handle.

'HHHHHHHH' is an 8-digit hex value corresponding to a UUID handle. If the first 4 HHHH is 'FE01' then it is a Bluetooth SIG adopted UUID and the 16-bit value is the next 4 digits.

Note the two spaces between the first ':' and the 'D'

3.4.2.4 ADAD

ADAD:advSet, address

This response is emitted after the AT+ADAD command has been used to query the advertising address and contains the requested advertising address, if no advertising set was provided then the value of 'advSet' will be '*'

3.4.2.5 TXPO

TXPO:hIdx, power

This response is emitted after the AT+TXPO command has been used to query the transmission power and contains the requested value, if no connection handle was provided then the value of 'hIdx' will be '*', the resultant power level is in dBm.

3.4.2.6 CSEC

CSEC:hIdx, flags, secMode, secLevel

This response is emitted after the AT+CSEC command has been used to query the connection security and contains the requested details, 'flags' is in hexadecimal format and consist of the following bitmask values:

Bit	Value	Description
0	0x1	Device is in central mode for connection
1	0x2	A bond exists in the bond database for this address
2	0x4	The connected device is encrypted using the data in the bond database
3	0x8	Encryption for this connection is enabled
4	0x01	MITM for this connection is enabled
5	0x20	LESC for this connection is enabled

'secMode' refers to the security mode of the connection and security level refers to the security level of the connection, which can be one of the following:

Mode	Level	Description
1	1	Open (no security/encryption)
1	2	Encrypted link, no MITM protection
1	3	Encrypted link with MITM protection
1	4	LESC 128-bit key encrypted link with MITM protection
2	1	Signed or encrypted link, no MITM protection
2	2	Signed or encrypted link with MITM protection

3.4.2.7 ENCRYPT

ENCRYPT

The command ATD is in progress and has reached the encrypted state before final confirmation which will be the "CONNECT" response.

3.4.3 Response: Asynchronous

A host must be designed to expect any of these responses at any time. To help with enabling a host to be as stateless as possible, all these responses have a unique 2 letter starting sequence to quickly determine what it means and how it gets processed.

3.4.3.1 AB

AB:hIdx, respcode

This is triggered when the AT+GCRD command attempts to read the content of an attribute in a remote GATT table and it is successful, but it fails to store that content locally. RespCode is a value that can referenced in the Laird Connectivity utility UwTerminalX.

3.4.3.2 AD

AD0:t addr14hex rssi "name"
AD1:t addr14hex rssi "name"
ADE:t addr14hex rssi "name"
ADS:t addr14hex rssi "name"

These messages happen asynchronously when scanning for advertisements.

The 'AD1' variant is when scanning using the AT+LSCN command while waiting for an incoming VSP connection.

'ADE' response is when an extended advert report has arrived.

'ADS' response is when an extended scan report has arrived.

't' is the advert type which will be 0 to 3 as per the Bluetooth specification where 0 implies that advert is connectable and is always 0 (and has no meaning) when the response is 'ADE' or 'ADS'

'addr14hex' is a hex string exactly 14 characters long that is the address present in the advert and the first 2 characters are used to determine the type (such as resolvable, static, etc.).

'rssi' is the RSSI of the received packet and will usually be a value between abut -30 and -100. Lower values indicate weaker signal.

' "name" ' is the device name if it has been supplied in the advert.

None of the other AD elements are displayed. If you wish for that information to also be displayed, it is encouraged that the supplied smartBASIC application be modified as required, or to use the AT+SFMT 1 command to force all the data in an advert to be sent in the response as a hex string. Be aware that extended adverts can have a payload as large as 255 bytes and so in that case the response will be at least double that.

See function HndlrAdvReport00(). This function is called each time an advert report is received (look for the 'print' statement).

3.4.3.3 AE

AE:

When adverts are started with AT+EADV and the 'maxCount' parameter is non-zero, then this async response is sent when those many adverts have been sent and advertising is automatically stopped.

3.4.3.4 AK

AK:i

An indication that was initiated using the command AT+GSIC has been acknowledged and 'i' is the index of the characteristic that was indicated, and to recap 'i' was provided when the characteristic had been entered into the local GATT table using the command AT+GSCE.

3.4.3.5 AR

AR:hIdx, offset, hexDataString

This is triggered when the AT+GCRD command is used to read the content of an attribute in a remote GATT table and it successfully reads it. Here, 'hIdx' is the connection handle index, 'offset' is the offset that was requested when the read was requested and 'hexDataString' is the data in hex string format.

3.4.3.6 AS

AS:hIdx, erStatus

This is triggered when the AT+GCRD command is used to read the content of an attribute in a remote GATT table and it fails. Here, 'hIdx' is the connection handle index, 'erStatus' is the reason for the failure and will be an integer value as follows:

Hex	Dec	Description
0x0001	1	Unknown or not applicable status
0x0100	256	Invalid error code
0x0101	257	Invalid attribute handle
0x0102	258	Read not permitted
0x0103	259	Write not permitted
0x0104	260	Used in ATT as invalid PDU
0x0105	261	Authenticated link required
0x0106	262	Used in ATT as request not supported
0x0107	263	Offset specified was pas the end of the attribute
0x0108	264	Used in ATT as insufficient authorisation
0x0109	265	Used in ATT as prepare queue full
0x010A	266	Used in ATT as attribute not found
0x010B	267	Attribute cannot be read or written using read/write blob requests
0x010C	268	Encryption key size used is insufficient
0x010D	269	Invalid value size
0x010E	270	Very unlikely error
0x010F	271	Encrypted link required
0x0110	272	Attribute type is not a supported grouping attribute
0x0111	273	Encrypted link required
0x0112	274	Reserved for Future Use – Range 1 begin
0x017F	383	Reserved for Future Use – Range 1 end
0x0180	384	Application range begin
0x019F	415	Application range end
0x01A0	416	Reserved for Future Use – Range 2 begin
0x01DF	479	Reserved for Future Use – Range 2 end
0x01E0	480	Reserved for Future Use – Range 3 begin
0x01FC	508	Reserved for Future Use – Range 3 end

0x01FD	509	Profile and Service Error: (CCCD) improperly configured
0x01EE	510	Profile and Service Error: Procedure already in progress
0x01FF	511	Profile and Service Error: Out of range

3.4.3.7 AW

AW:hIdx, status

This is triggered when the AT+GCWA command is used to write the content of an attribute in a remote GATT table and demonstrates the outcome of that attempt. Here, 'hIdx' is the connection handle index, 'status' is an integer value which will be 0 for success otherwise a value as listed in the section for the "AS" response.

3.4.3.8 CC

CC:i, newValue

This message happens asynchronously when a remote GATT client writes into a CCCD descriptor of one of the local characteristics identified by 'i', which was provided as a result of AT+GSCE when the characteristic was created and committed. The parameter 'newValue is an integer.

See responses 'WR' and 'SC' when the Characteristic Value and Sccd are written.

3.4.3.9 CONNECT

CONNECT 0, address, interval, sprvsnTout, latency

For a device waiting for an incoming VSP connection, this is an asynchronous message to confirm that a connection is fully setup from a device with Bluetooth 'address' where the connection interval is 'interval' in microseconds, 'sprvsnTout' is the link supervision timeout in microseconds and 'latency' is the slave latency.

The first parameter will always be 0 as that handle index is dedicated for VSP connections.

Note: Lower case 'connect' implies a non-VSP connection.

3.4.3.10 connect

connect hIdx, address, interval, sprvsnTout, latency

For a device waiting for an incoming non-VSP connection this is an asynchronous message to confirm that a connection is setup from a device with Bluetooth 'address' where the connection interval is 'interval' in microseconds, 'sprvsnTout' is the link supervision timeout in microseconds and 'latency' is the slave latency.
The first parameter 'hIdx' is the handle index which are non-zero and dedicated for non-VSP connections.

Note: An upper case CONNECT implies a VSP connection.

3.4.3.11 discon

discon hIdx, reason

This indicated that the connection identified by the handle hIdx has been dropped and the reason for disconnection is specified by the integer value 'reason'. See source code for the *smart*BASIC application for 'reason' values by searching for the string "CONN_ERROR_"

3.4.3.12 encrypt

encrypt hIdx

This indicates that the connection identified by the handle hIdx has entered the encrypted state.

3.4.3.13 FC

FC:hIdx, hAttr, props

This is triggered by the AT+GCFA command to search for a characteristic's attribute handle. 'hIdx' is the connection handle index, 'hAttr' is handle of the attribute if found, otherwise it will be 0. 'Props' is the property bitmask of that characteristic.

Note: hAttr==0 if characteristic not found.

3.4.3.14 FD

FD:hIdx, hAttr

This is triggered by the AT+GCFA command to search for a descriptor's attribute handle. 'hIdx' is the connection handle index, 'hAttr' is handle of the attribute if found, otherwise it will be 0.

Note: hAttr==0 if descriptor not found.

3.4.3.15 IN

IN:hIdx, hAttr, hexDataString

This message happens asynchronously when a remote GATT server sends this device a notification or an indication where 'hIdx' identifies the server connection, 'hAttr' is the handle of the attribute that got updated with the new data in 'hexDataString' which is hexadecimal format.

Note: If it is an indication then a GATT acknowledgement has been automatically sent.

3.4.3.16 OL

OL:hAddr, hHash, hRand

While pairing if the I/O capability Sreg107 is appropriate and the other end also has a user interface, then this could be sent to the host to request a 32 hex character OOB (out of band) string which it then submits using the AT+PRSP command. 'hAddr' is the address of the remote requesting device, 'hHash' is the hash, 'hRand' is the random number.

3.4.3.17 passkey

passkey? hIdx

While pairing if the I/O capability Sreg107 is appropriate and the other end also has a user interface, then this could be sent to the host to request an integer value in the range 0 to 999999 which it then submits using the AT+PRSP command. 'hIdx' identifies the server connection.

3.4.3.18 oobkey

oobkey? hIdx

While pairing if the I/O capability Sreg107 is appropriate, then this could be sent to the host to request an out-of-band (OOB) key which it then submits using the AT+PRSP command. 'hIdx' identifies the server connection.

3.4.3.19 lescoob

lescoob? hIdx

While pairing if the I/O capability Sreg107 is appropriate, then this could be sent to the host to request an out-of-band (OOB) LESC key which it then submits using the AT+OOBR command. 'hIdx' identifies the server connection.

3.4.3.20 xxkey

xxkey? hIdx

This response indicates an unsupported authentication type during paring, where 'hIdx' identifies the server connection.

3.4.3.21 RING

RING address, [U|T]

When waiting for a VSP connection, this message is the first indication to the host that a connection is in progress from a device with Bluetooth 'address'.

The [U|T] implies either a 'U' which implies that the 'address' is not in the trusted device bond database or a 'T' which implies the incoming VSP connection is from a trusted device.

3.4.3.22 SC

SC:i, newValue

This message happens asynchronously when a remote GATT client writes into a SCCD descriptor of one of the local characteristics identified by 'i', which is provided as a result of AT+GSCE when the characteristic was created and committed. The parameter 'newValue is an integer.

See responses 'WR' and 'CC' when the Characteristic Value and CCCD are written.

3.4.3.23 showcode

showcode hIdx, passcode

While pairing, if the I/O capability Sreg107 is appropriate and the other end also has a user interface, then this could be sent to the host to display the integer value 'passCode' as a 6-digit decimal number with trailing 0's so that a 6-digit number is shown, where 'hIdx' identifies the server connection. This end needs to confirm with a Yes or No to complete the pairing and that is done using the command AT+PRSP. **Note that this is a passkey entry only and is not for LESC numerical comparison**, the two events are different, numerical comparison events will come through as comparecode (described below)

3.4.3.24 comparecode

comparecode hIdx, comparisonCode

While pairing, if the I/O capability Sreg107 is appropriate and the other end also has a user interface, then this could be sent to the host to display the integer value 'comparisonCode' as a 6-digit decimal number with trailing 0's so that a 6-digit number is shown, where 'hIdx' identifies the server connection. This end needs to confirm with a Yes or No to complete the pairing and that is done using the command AT+PRSP. **Note that this is an LESC numerical comparison entry only and is not for passkey entry**, the two events are different, passkey events will come through as showcode (described above)

3.4.3.25 scanned

scanned

When waiting for an incoming VSP connection it is possible to also scan for adverts for a specified interval using the command AT+LSCN which will then trigger advert report "AD". When the scan times out, this response will be sent to the host.

3.4.3.26 WR

WR:i, hexDataString

This message happens asynchronously when a remote GATT client writes into one of the local characteristics identified by 'i' which was provided as a result of AT+GSCE when the characteristic was created and committed. The parameter 'hexDataString' is the new data that was written into the characteristic.

See responses 'SC' and 'CC' when the SCCD and CCCD are written.

3.4.3.27 xxkey

xxkey

While pairing, if the I/O capability Sreg107 is appropriate and you receive this, then contact Laird Connectivity. This is to cater for a future pairing authentication scheme. The API allows for this as a hypothetical future scenario.

3.4.3.28 LL

LL:hIdx, txrxTimeReducedBy, txSizeReducedBy, rxSizeReducedBy

This message happens asynchronously when the packet length or transmission/reception time has been limited due to constraints of the local or remote device. The time value is in us and the size values are in bytes, 'hIdx' identifies the connection.

3.4.3.29 PL

PL:hIdx, txPower

This message happens asynchronously when the transmission power has been limited due to constraints of the local to remain in regulatory compliance. The power is measured in dBm, 'hIdx' identifies the connection. This event only exists when using the BL654PA module.

3.4.3.30 PI

PI:hIdx, code, source, hexFlags, hexSecurityModeLevels

This message happens asynchronously when a pairing has been successful or failed and includes details on it, 'hIdx' identifies the connection, 'code' indicates success if 0, otherwise failure (if a failure has occurred, all other fields should be ignored), 'source' is set to the source of the error which is 0 for the local device or 1 for the remote device, 'hexFlags' is a bitmask field in hex format with the following bit values:

Bit	Value	Description
0	0x1	Bonded
1	0x2	*Reserved for future use*
2	0x4	LESC pair/bond

'hexSecurityModeLevels' is a bitmask field in hex format with the following bit values:

Bit	Value	Description
0	0x1	Security mode 1, level 1: No security, open link
1	0x2	Security mode 1, level 2: Encrypted link, without MITM protection
2	0x4	Security mode 1, level 3: Encrypted link, with MITM protection
3	0x8	Security mode 1, level 4: Encrypted LESC link with 128-bit encryption key, with MITM protection
4	0x10	Security mode 2, level 1: Signed or encrypted link, without MITM protection
5	0x20	Security mode 2, level 2: Signed or encrypted link, with MITM protection

3.4.3.31 SR

SR:hIdx, hexFlags

This message happens asynchronously when a remote peripheral device has issued a security request, this should be responded to by pairing/authenticating/disconnecting using AT+PAIR, AT+LENC or AT+LDSC. 'hIdx' identifies the connection, 'hexFlags' is a bitmask field in hex format with the following bit values:

Bit	Value	Description
0	0x1	Bonding is supported
1	0x2	MITM is supported
2	0x4	LESC is supported
3	0x8	*Reserved for future use*
4	0x10	Key press generation events are supported

3.4.3.32 OB

OB:hIdx, hexFlags, address

This message happens asynchronously when a remote device has attempted to overwrite a bond in the bond database of another device. 'hIdx' identifies the connection, 'flags' is a bitmask corresponding to the following values:

Bit	Value	Description
0	0x1	The existing bond is an LESC bond, but new type is a legacy pairing/bond
1	0x2	The new request is a pair request whilst the existing was a bond
2	0x4	The existing bond has an IRK but the new pairing/bond does not
3	0x8	The existing bond and new pairings have IRKs, but they differ
4	0x10	The existing bond had a higher security level than the new pairing/bond request has
5	0x20	The existing bond supported signing using CSRK, but new pairing/bond does not,
6	0x40	The existing bond supported authentication using MITM, but the new pairing/bond does not

Address is the BLE address of the connection. This must be replied with by using the AT+BOVR command to accept or decline the action before additional commands can be used.

3.4.3.33 CP

CP:hIdx, initiator, hexFlags, ioCap, minKeySize, maxKeySize

This message happens asynchronously during a pairing/bond process when the process needs to be confirmed due to the confirm pairing status, which was set using AT+PCFG, this needs to be responded to by using the AT+PCNF command to accept or decline the pairing. 'hIdx' identifies the connection, 'initiator' indicates the device that started the process which will be 0 for the local device and 1 for the remote device, 'hexFlags' is a bitmask field in hex format with the following bit values:

Bit	Value	Description
0	0x1	Bonding is supported
1	0x2	MITM is supported
2	0x4	LESC is supported
3	0x8	OOB is supported
4	0x10	Key press generation events are supported

'ioCap' described the input/output capability of the device:

Value	Description
0	No input/output capability (just works)
1	Display with yes/no input
2	Keyboard only
3	Display only
4	Keyboard with display

'minKeySize' is the minimum size of the encryption key that the remote device supports in bytes (or 0 if encryption is not supported), 'maxKeySize' is the maximum size of the encryption key that the remote device supports in bytes.

3.4.3.34 MT

MT:hIdx, vspChunkLen

This message happens asynchronously after an MTU exchange has occurred, 'hIdx' identifies the connection, 'vspChunkLen' is the maximum size of on-air VSP messages in bytes.

3.4.3.35 PU

PU:hIdx, txPhy, rxPhy

This message happens asynchronously when a PHY update has succeeded, 'hIdx' identifies the connection, 'txPhy' and 'rxPhy' specify what the new PHY is for the connection as per:

Value	Description
1	1M PHY
2	2M PHY
3	*500Kbps LE Coded PHY – this is not supported*
4	125Kbps LE Coded PHY

3.4.3.36 PF

PF:hIdx, status

This message happens asynchronously when a PHY update has failed, 'hIdx' identifies the connection, 'status' is the error for the PHY request/update not being successful.

3.5 AT Commands (NFC Operation)

If the module is capable of NFC operation and the functionality is enabled at smartBASIC app compile time (the application filename will have ".nfc" appear in it) then this section describes the commands for NFC operation.
The compile time enabling bitmask value is 0x00200000 which needs to be provided via the following line which is at the top of the smartBASIC source code file, and will have been done for you already if you downloaded a file with ".nfc" in the filename:

```
#set $cmpif, 0xhhhhhhhh
```

These commands access the appropriate NFC smartBASIC functions as described in the appropriate modules' user guide.

When an active NFC coil energizes or de-energizes this module's NFC coil, then an asynchronous response is sent to the host, and if the tag is successfully read or written by the active NFC device then an appropriate asynchronous response is also sent. See a full description of these responses in the section "Responses (NFC Operation)" in the next sub-chapter

3.5.1 AT+NOPN

Command	`AT+NOPN max_ndef_buflen <,writeable>`
Description	Open the NFC interface and reserve `max_ndef_buflen` bytes of memory to create an NDEF message which can contain multiple records as long as there is memory to accommodate them. If the optional <,writeable> is not present then the tag is read only. If it is present and has a value of 1, then it will writable when the underlying firmware in the module allows that. When write capability is absent it will only open in read only mode. The argument `max_ndef_buflen` should be within the range 128 to 512 and it can be changed by modifying the values of the #defines `NFC_MIN_TAG_SIZE` and `NFC_MAX_TAG_SIZE` appropriately in the .sb file.
Possible Responses	`OK` `ERROR`

3.5.2 AT+NCLS

Command	`AT+NCLS`
Description	Close the NFC interface and all memory previously reserved is released
Possible Responses	`OK` `ERROR`

3.5.3 AT+NRST

Command	`AT+NRST`

Description	Reset the NDEF message buffer so that is marked as empty, so that a new message can be added using the commands AT+NRAT and AT+NRAG
Possible Responses	`OK` `ERROR`

3.5.4 AT+NRAT

Command	`AT+NRAT "lang", "message"`
Description	Add a Text type record to the NDEF message buffer that was made available via AT+NOPN. For this record the NTF type will be se to 0x01 and the 'type' field in the record header will be set to the string value "T". The ID field in the header will be set as empty. The "lang" argument is a language specifier formatted so that the first character is the length of the string specifying the language. So for example, to specify that the message is in English use "\02en" where the three character \02 sequence will be escaped into 2 and 'en' the abbreviation for English. The "message" argument is any message and you can add UTF-8 strings by adding appropriate escape sequence for non-printable bytes.
Possible Responses	`OK` `ERROR`

3.5.5 AT+NRAG

Command	`AT+NRAG tnf, "type", "id", "payload"`
Description	This command is used to add any record to the message as all the fields in the header and the payload can be explicitly specified. The tnf value in the first byte of the ndef record header shall be in the range 0 to 7 as per the NDEF specification. The "type" value is written to the type field in the header. The "id" value will populate the ID field in the header and can be specified as empty using an empty double quoted string "". The payload of the ndef record is populated by "payload". Note that any of the 3 string arguments can have non-printable characters by escaping such values with the three-character sequence \hh where hh are the 2 hex digits required to fully specify an 8-bit value.
Possible Responses	`OK` `ERROR`

3.5.6 AT+NCMT

Command	`AT+NCMT`
Description	Commit NDEF message buffer to the NFC stack so that it can be made available to a reader
Possible Responses	`OK` `ERROR`

3.5.7 AT+NSEN

Command	`AT+NSEN`

Description	Enable the NFC coil so that an active reader/writer can access the ndef message that was committed using the most recent AT+NCMT command
Possible Responses	`OK` `ERROR`

3.5.8 AT+NSDS

Command	`AT+NSDS`
Description	Disable the NFC coil so that an active reader/writer cannot access the ndef message so that a new message can be committed if required.
Possible Responses	`OK` `ERROR`

3.6 Responses (NFC Operation)

This section describes all the responses generated by the module related to NFC operation, and only if that feature is compile-time enabled when the smartBASIC application is loaded into the module.

To simplify reception of messages in the receiving device, each message starts with a \n character and ends with a \r character.

After stripping the \n start character, each response will start with a **unique** 2-character sequence to help the host decode the response in a stateless manner.

Some responses are synchronous which mean they are used to terminate a command so that the command parser can process more commands and some that are asynchronous, meaning they can happen at any time.
However please note that if there is an ongoing VSP connection and so data is being transparently bridged between the UART and air-interface, then all NFC related asynchronous messages are suppressed and the only way to know after the VSP connection ceases is via the counts returned by ATI50 and ATI51

3.6.1 Response: Synchronous and Terminating

When a host receives these responses it can issue new commands and expect them to be processed immediately.

3.6.1.1 ERROR

ERROR nn

A command was not successfully actioned and 'nn' is an error code. Error Code are as follows:-

- 60: NFC Not Open
- 61: NFC NDEF Message Empty
- XX: Other Generic Errors (See earlier section)

3.6.1.2 OK

OK

A command was successfully expedited.

3.6.2 Response: Synchronous and Not Terminating

When a host receives these responses, it cannot issue new commands and expects them to be processed immediately as a terminating response is still to come.

None have been defined yet

3.6.3 Response: Asynchronous

A host must be designed to expect any of these responses at any time. To help with enabling a host to be as stateless as possible, all these responses have a unique 2 letter starting sequence to quickly determine what it means and how it gets processed.

3.6.3.1 NS

NS:state

This message is asynchronously sent when an active NFC device energizes or de-energizes this modules NFC coil. The 'state' will be 1 for energies and 0 for de-energies.

3.6.3.2 NR

NR

These messages happen asynchronously when the tag committed using AT+NCMT has successfully been read by an active NFC reader.

3.7 Compile Time Default Behaviour

This AT interface behaviours is supplied in source format and the reader is free and encouraged to modify and enhance as desired.

Behaviour may be altered by altering the values of #defines at the top of the source code file called "$autorun$.AT.interface.xxx.yyy.zzz.sb" which includes the file "LIB.AT.interface.sb".

Where xxx.yyy.zzz is some descriptive text to help you maintain several versions of the application in your source repository.

Noteworthy defines are as follows:

`ATI_RESPONSE_0`

This is a small string which is returned for command ATI0

`ATI_RESPONSE_10`

This is a small string which is returned for command ATI10

`MAX_CONNECTIONS`

Currently set to 8, but you can reduce it to ease the pressure on memory usage

`MAX_CHARACTERISTICS`

Currently set to 24 and that defines the maximum number characteristics that can be added using the AT+GSCE command.

`CONN_INTERVAL_MIN_ASPERIPH_US`
`CONN_INTERVAL_MAX_ASPERIPH_US`

These are minimum and maximum connection intervals as a peripheral. The module will accept anything that is provided, and it will not trigger a connection parameter renegotiation.

`NFC_MIN_TAG_SIZE`
`NFC_MAX_TAG_SIZE`

These are minimum and maximum buffer size that the NFC interface can be opened with to save one or more NDEF messages in.

MaxDevNameSize

Maximum allowable size of the advertised device name which should not be set to larger than 20.

MaxCmdStringSize

Maximum allowable size of a single AT command line in terms of characters which includes the terminating \r character.

#set $cmpif, 0xhhhhhhhh

This allows compile time switches to be manipulated.

This allows for code to be compiled out to make code space. For example, if you don't need to use the GATT client table map command, it can be done by clearing the appropriate bit in 0xhhhhhhh. Please examine the comments around that line.

3.8 Low Power UART Operation

This application is by its very nature requires a host to control it by sending AT commands over the UART interface given it operates like a modem where the data is relayed over a virtual serial connection in a BLE connection.

The UART interface that is embedded inside the microcontroller at the heart of the Laird Connectivity module consumes about 250 to 350 microamps when it is open.

Laird Connectivity has shown that it is possible to operate the module in **doze** mode so that the total current consumption can be as low as sub 10 microamps.

BLE is a low power radio technology and the radio chip has been optimised so that in-between radio events it can go to sleep and so a typical power profile can be shown to be doze current of sub 10uA and then about 8000 microamps when there is a radio event which can be of duration from a few 10s of microseconds to over 1000 microseconds and the radio event can occur as quickly as 7500 microseconds and as slow as over 4000000 microseconds. This shows that the duty cycle of low to high power provides for overall low average current consumption.

When the AT Interface application is loaded in the module, it by necessity requires that the UART is in operation and so the average quiescent current is going to be in the region of 250 to 350 microamps instead of the expected sub 10 microamps.

If your use case is such that there will be occasional traffic over the UART interface, then it is possible to enable a smartBASIC cross-compile switch (as in, you can do that) so that it will operate such that it will close the UART most of the time (or you select a file to download that has ".lpuart" in the filename as that compile switch is pre-enabled for you).

This requires a **cooperative** existence with the host which means an extra GPIO line connected between the host and the module is used to manage the open/close operation of the module's UART.

This GPIO, which is a digital output from the UART host, which for convenience in this guide, will be called the 'Keep UART Open' line and by default it is connected to the module's GPIO input line 24 and for your convenience can be changed via SRegister 109 using the command ATS109=X where X is the new GPIO line. Note that Sregister 109 is also used to specify the 'drop connection' line when in fast connection mode. This implies that low power operation is only available in normal mode where it is possible to get throughputs well in excess of 92kbps which is the maximum achievable when the baudrate on the UART is set to the default 115200.

The AT Interface app has been crafted so that if it sees the 'Keep UART Open' high then it will NOT try to close the UART automatically, otherwise if there has been no UART activity for a default time of 5 seconds then it will automatically close the UART to reduce the current consumption. While the UART is closed, if there is incoming data from over BLE that needs to be relayed to the host then it will automatically open the UART and send the data and start a shutdown timer. The default timeout of 5 seconds can be changed via the SRegister 213 and after a change it will require saving using AT&W as the SRegister is only read on power up or a reset invoked by the command ATZ.

When the UART is automatically shut down, it will de-assert the RTS line which is a signal to the serial port in the host that it should stop sending data. If the host sees that the module's RTS is de-asserted (which it will detect via its own CTS input line) and that it has set the 'Keep UART Open' line low, then it can set that line high which will result in the RTS line being reasserted after the module reopens the UART and so that data can be received by the module.

In summary, low power operation is only available in normal throughput operation and requires an additional GPIO line output from the host that conveys a 'Keep UART Open' command to the module and RTS line from the module should be monitored for serial port status.

To download the low power version of the application to the module please contact Laird Connectivity for appropriate settings for compile time “#set $cmpif” value so that the bit 0x00400000 is set or download the variant of the application that has the text “.lpuart” in the filename