*smart*BASIC App Migration Guide

BL65x *smart*BASIC Applications

Version 1.0

The scope of this guide relates to changes required to migrate *smart*BASIC apps written for previous versions of firmware to use functions and events made available in the latest release.

REVISION HISTORY

Version	Date	Notes	Contributor(s)	Approver
1.0	08 Mar 21	Initial release	Kieran Mackey	Jonathan Kaye

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

CONTENTS

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

1 Overview

The latest releases of *smart*BASIC firmware have made changes to the API from previous versions. These changes provide security enhancements as well as the ability to be alerted when connection parameters change (which may be due to memory requirements, regulatory requirements, or limitations of the remote device) This document explains the changes required to migrate *smart*BASIC apps to the latest release. Table 1 details affected versions:

Table 1 Previous and new BL65x firmware releases

Product	Previous Firmware	Release Date	New Firmware	Release Date
BL652	v28.10.7.2	May 2020	v28.11.8.0	March 2021
BL653, BL653µ	v30.1.1.2*	Oct 2020	v30.2.3.0	Jan 2021
BL654, BL654PA	v29.4.6.6	June 2020	v29.5.7.2	Jan 2021

* BL653 also had a release v30.2.2.0 in Jan 2021 that contained a partial implementation of these changes

The Summary of Changes section provides a summary of all changes made in the above versions of firmware. App Migration details changes that may affect *smart*BASIC applications written for previous versions of firmware and how these can be adapted for the latest changes.

2 Nomenclature

The font `courier new` is used to indicate *smart*BASIC functions

3 Glossary

Table 2: Glossary

Term	Definition
Bluetooth Address	There are four types of addresses defined for Bluetooth LE: ▪ **Public IEEE Format** – Manufacturer specific IEEE Bluetooth address and company identifiers can be purchased through the IEEE Registration Authority. ▪ **Random Static** – Burned into radio silicon upon shipment or can be randomly generated to a new value at each power cycle. ▪ **Random Private Resolvable** – Used in bonded devices and requires the Identity Resolving Key (IRK) be shared during phase 3 of the pairing procedure. This changes periodically based on a timer or other method. This is the default use case for iOS devices and for Android devices with Lollipop or newer. ▪ **Random Private Non-Resolvable** – Shared between bonded devices for use during reconnection. This changes with each connection
Channel Map	The 2.4 GHz transmission frequency of Bluetooth Low Energy is subdivided into 40 channels (3 for advertising, 37 for data). The channel map states which of these channels are allowed for transmission on a device.
Connection Handle	This is a *smart*BASIC term for the ID that is unique to each active Bluetooth LE connection
Connection Interval	Time between connection events. Each connection event is an exchange of data between Bluetooth devices.
Event Length	Event length is the maximum time the radio can be active during a connection interval. See CFG 219 Event Length for more details.
GATT Table	In a GATT server, this is a table of all attributes accessible to remote devices. In a GATT client, this is a table of all attributes used by the remote Bluetooth device that is found during service discovery. ***Note:*** *GATT server/client roles are independent from GAP peripheral/central roles.*
LESC	Bluetooth Low Energy Secure Pairing – This is a more secure pairing method introduced to Bluetooth in version 4.2. See [C] for information on Bluetooth LE pairing.
SoftDevice	Nordic Semiconductor Bluetooth stack that makes up part of *smart*BASIC firmware.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

4 SUMMARY OF CHANGES

4.1 New Functions

Please see the associated user's guide for the respective module and firmware version for details on the following functions. Refer to BL652 [E], BL653/BL653µ [J], and BL654/BL654PA [O] *smart*BASIC extension guides.

4.1.1 BleGetConnHandleFromResolvedAddr()

Used to get a connection handle from a resolved Bluetooth address (bond required)

4.1.2 BleGetResolvedAddrFromConnHandle()

Used to get a resolved address from a connection handle (bond required)

4.1.3 BleGetAdvertisingAddress$()

Returns the currently active advertising address including private resolvable/non-resolvable addresses

4.1.4 BleSecMngrAuthFailPref()

Sets configuration if a device should be disconnected on authorisation failure (pairing, bonding, or encryption failure)

4.1.5 BleTxPowerGet()

Gets the current or default transmit power for a role/connection. The function requires parameters *nRole* for the role type and *nHandle* for connection handle when requesting the power for an active connection. The following table (Table 3) defines these roles.

Table 3: Role definitions

Role	nRole Value	Description
Scanning	0	Transmission power for scanning
Advertising	1	Transmission power for advertising
Pairing	2	Transmission power for pairing
Default connection	3	Default transmission power for connections
Active connection	4	Transmission power for connection of provided connection handle

4.1.6 BleGetChannelMap()

Returns the currently active channel map for a connection

4.1.7 BleSecMngrPairConf()

Used to set what conditions trigger a pairing confirmation event being thrown

4.1.8 BlePairingInfo()

Used to return details on the device and pairing requested during a pairing request with a device; only used on incoming pairing request event

4.1.9 BlePairingResponse()

Used to respond to an incoming pairing request event to accept or decline it; only used on incoming pairing request event

4.1.10 BleGetCurConnSecInfo()

Returns information on the current security level of a connection

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

4.1.11 BleOverwriteBond()

Used in the bond overwrite event (EVBLEOVERWRITEBOND) to accept or decline allowing the bond overwrite; only used on incoming bond overwrite event

4.2 New Events

Refer to the associated user's guide for the respective module and firmware version for details on the following events.

4.2.1 EVBLE_PASSKEY

Replaces BLE_EVBLEMSGID_DISPLAY_PASSKEY from EVBLEMSG and includes connection handle. This event is thrown when there is a Bluetooth LE pairing event in progress that requires the entry/acceptance of a passkey

4.2.2 EVBLE_AUTHKEY

Replaces BLE_EVBLEMSGID_AUTH_KEY_REQUEST from EVBLEMSG and includes connection handle. This event is thrown when an authorisation key is requested to be given by the remote device

4.2.3 EVPACKETLENGTHLIMITED

New event for if either this device or the other device is too constrained by memory limitations of either device and cannot use the requested packet length

4.2.4 EVTXPOWERLIMITED

New event for BL654PA only which indicates if the connection power must be lowered to remain within regulatory laws. This can happen when a remote central device limits the Bluetooth LE channel map which the local peripheral device is unable to deny

4.2.5 EVBLEAUTHINFO

New event which includes authorisation details on a successful or failed authorisation attempt with a remote device. This can be thrown on a central or peripheral device once phase three of the pairing process is complete

4.2.6 EVBLESECREQUEST

New event which can be seen on central devices when a remote device requests it to either authenticate, pair, or disconnect (and now BL65x devices can trigger this as peripheral devices using `BlePair()`)

4.2.7 EVBLEOVERWRITEBOND

New event which is used when a remote device attempts to bond and specifies a different bonding address then the connection address of a device which already exists in the bond database, indicating a possible intrusion attack

4.2.8 EVBLEMSG

The EVBLEMSG event is triggered under multiple conditions and an ID parameter is also passed to indicate the cause of the event. The following event IDs were added:

- BLE_EVBLEMSGID_AUTHENTICATION_SUCCESSFUL (29) which triggers if authentication is successful
- BLE_EVBLEMSGID_CONFIRM_PAIRING (30) which triggers when a pairing request needs to be confirmed (BlePairingInfo() and BlePairingResponse() are used at this point

  Note: Pairing confirmation is enabled by default.

- BLE_EVBLEMSGID_CONFIRM_TIMED_OUT (31) which triggers whenever a pairing confirmation request is received by the local device, but the local device fails to respond in time. The timeout is set to 25 seconds

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

4.3 New Features

4.3.1 Automatically Discover 128-bit UUIDs (AT+CFG 220)

Allows the automatic discovery of UUIDs from a remote device (such as when scanning a GATT table). It is enabled by default. Setting the value of CFG 220 to *1* allows the discovery of UUIDs outside those defined by the Bluetooth Special Interest Group (Bluetooth SIG) without needing to register the base UUID with the `BleHandleUuid128()` function. This means UUIDs that do not have the Bluetooth SIG base UUID of 00000000-0000-1000-8000-00805F9B34FB go in the GATT table when the service discovery function `BleDiscServiceFirst()` is called.

4.3.2 Sysinfo

New indexes were added to get information about the state of firmware, memory, and hardware:

- Sysinfo

2045	Returns SoftDevice start-up code. Result code can be decoded by UwTerminalX. A return of 0 means SoftDevice started up successfully
2046	Returns SoftDevice RAM (in bytes) required for the active configuration
2047	Returns SoftDevice RAM (in bytes) required for the desired configuration. See Configuration memory for more details
2053-2056	Static NFC UIDs 0-3
2090	Bluetooth LE maximum Packet Length (active in SoftDevice)
2091	Bluetooth LE connection Event Length (active in SoftDevice)
2092	Bluetooth LE maximum central connections (active in SoftDevice)
2093	Bluetooth LE maximum peripheral connections (active in SoftDevice)
2094	Bluetooth LE maximum combined connections (active in SoftDevice)

- Sysinfo$

0	Static device descriptor (product family, BL652, BL653, or BL654)
2057	Returns full NFC UID as string

4.4 Changed Features

Refer to BL652 [E], BL653/BL653µ [J], and BL654/BL654PA [O] *smart*BASIC extension guides for the respective module and firmware version for details on the following changes.

4.4.1 Function: BleBondMngrErase()

This was previously a sub-routine but is now a function and returns a status code. This can now indicate if the index of the bond to erase is invalid.

4.4.2 Function: BleBondingIsTrusted()

Additional bit values added for LESC 128-bit key and if signing is required. LESC 128-bit keys can be used when using OOB (Out of Band) pairing or passkey pairing. Signing is required for authenticated pairing. See [C] for more information about pairing methods.

An existing parameter for this function, *keyInfo* is extended for new bit values. The bitmask values are as follows with new bits highlighted in bold:

Bit 0	Set if MITM is authenticated
Bit 1	Set if it is a rolling bond and can be automatically deleted if the database is full and a new bonding occurs
Bit 2	Set if an IRK (identity resolving key) exists

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Bit 3	Set if a CSRK (connection signing resolving key) exists
Bit 4	Set if LTK as slave exists
Bit 5	Set if LTK as master exists
Bit 6	LESC key
Bit 7	LESC high-security mode key – 128-bit LESC key
Bit 8	Signing is required

4.4.3 Event: EVBLEMSG

The EVBLE event gets passed a numerical ID parameter to indicate what triggered the event. Two of these IDs have been deprecated:

- BLE_EVBLEMSGID_DISPLAY_PASSKEY (9) deprecated and should no longer be used. Event EVBLE_PASSKEY should be used instead
- BLE_EVBLEMSGID_AUTH_KEY_REQUEST (11) deprecated and should no longer be used. EVBLE_AUTHKEY should be used instead.

4.4.4 Configuration Value 219 *Event Length*

The configuration key *Set aside time per connection* (set with `AT+CFG 218`) was removed. It is replaced with *Event Length* which is set using `AT+CFG 219`

Event Length is active radio time per connection interval, between 2-800. Each unit is 1.25 milliseconds. See CFG 219 Event Length for more details

4.4.5 Other Changes

- Bluetooth LE connection/scan periods were previously set to a minimum of 1 second. This is lowered to 100 ms
- In some instances, depending on module configuration, the SoftDevice could have failed to start up because of using too much memory or other issues. The firmware now automatically reduces the configuration so the SoftDevice can start. Use `sysinfo(2047)` to get required memory for desired configuration and `sysinfo(2046)` for memory used in active configuration
- Peripheral devices can send security requests to central devices which requires them to pair, authenticate, or disconnect. This is performed using BlePair() which previously returned an error
- Bonds are no longer erased when upgrading firmware, however, bonds should be manually erased when upgrading from firmware versions prior to those listed in Table 1 as new bits have been added which were not supported in older firmware versions
- BL653 only – In previous firmware versions, if automatic crystal discovery was enabled (`AT+CFG 210 0`) then the module would fail to start. This is fixed.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

5 App Migration

This section details changes that may be required to keep existing apps functioning.

5.1 Essential Changes

5.1.1 Event EVBLEMSG, ID BLE_EVBLEMSGID_DISPLAY_PASSKEY

Replace – this requires existing code to be modified

Use Case	Any application that uses a EVBLEMSG event with the ID **BLE_EVBLEMSGID_DISPLAY_PASSKEY** (9)
Details	EVBLEMSG **BLE_EVBLEMSGID_DISPLAY_PASSKEY** (9) should be removed. This should be replaced with event **EVBLE_PASSKEY** ***Note:*** *For security, it should be ensured that either Pass Key or Numerical Comparison are the same on both ends.*

Example

Before:

```
function HandlerBleMsg(BYVAL nMsgId AS INTEGER, BYVAL nCtx AS INTEGER) as integer
  if (nMsgId == BLE_EVBLEMSGID_DISPLAY_PASSKEY) then
    print "\nDisplay Pairing Passkey ";nCtx
  endif
endfunc 1
OnEvent  EVBLEMSG              call HandlerBleMsg
```

After:

```
function HandlerBlePasskey(BYVAL Conn, BYVAL Passkey, BYVAL Flags)
    if (Flags == 0) then
        //This is a passkey
        PRINT "\n +++ Pass Key Request (NOT Numerical Comparison), Handle: ";integer.h'Conn;", Key: ";Passkey
    elseif (Flags == 1) then
        //This is a numerical comparison
        PRINT "\n +++ Numerical Comparison Request (NOT Pass key), Handle: ";integer.h'Conn;", Comparison: ";Passkey
    endif
endfunc 1
OnEvent EVBLE_PASSKEY      call HandlerBlePasskey
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

5.1.2 Event EVBLEMSG, ID BLE_EVBLEMSGID_AUTH_KEY_REQUEST

Replace – this requires existing code to be modified

Use Case	Any application that uses a EVBLEMSG event with ID **BLE_EVBLEMSGID_AUTH_KEY_REQUEST** (11)
Details	EVBLEMSG **BLE_EVBLEMSGID_AUTH_KEY_REQUEST** (11) should be removed. This should be replaced with event **EVBLE_AUTHKEY.**

Example

Before:

```
function HandlerBleMsg(BYVAL nMsgId AS INTEGER, BYVAL nCtx AS INTEGER) as integer
  if (nMsgId == BLE_EVBLEMSGID_AUTH_KEY_REQUEST) then
    print "\n +++ Auth Key Request, type=";nCtx
  endif
endfunc 1
OnEvent  EVBLEMSG              call HandlerBleMsg
```

After:

```
function HandlerBleAuthkey(BYVAL Conn, BYVAL Type, BYVAL Flags)
    print "\nAuth Key Request, Handle: ";integer.h'Conn;", Type: ";Type
endfunc 1
OnEvent EVBLE_AUTHKEY      call HandlerBleAuthkey
```

5.1.3 Event EVBLEMSG, ID BLE_EVBLEMSGID_CONFIRM_PAIRING

New – new code must be added

Use Case	Any application that makes use of Bluetooth LE pairing
Details	An EVBLEMSG event with ID **BLE_EVBLEMSGID_CONFIRM_PAIRING** (30) is triggered when a pairing request must be confirmed or rejected.

Example

Note: The following excerpt only contains the EVBLEMSG event. Refer to the *smart*BASIC extension guide and example BlePairingResponse.sb for the full script.

```
function HndlrBleMsg(byval nMsgId as integer, byval nCtx as integer)
    if (nMsgId == BLE_EVBLEMSGID_CONFIRM_PAIRING) then
        print "Confirm pairing (";nCtx;")\n"
        rc = BlePairingInfo(nCtx, initiator, flags, ioCap, minKeySize, maxKeySize, keyExchange)
        print "  Initiator: ";initiator;", Flags: ";integer.h'flags;", IOCap: ";ioCap;", Key size: ";minKeySize;"-";maxKeySize;", Key exchange: ";integer.h'keyExchange;"\n"
// Applications should implement their own system for accepting or declining pairing. This is an example only and should not be used in production code.
        rc = BlePairingResponse(nCtx, 1, 0)
        print "Accepted pairing\n"
    elseif (nMsgId == BLE_EVBLEMSGID_CONFIRM_TIMED_OUT) then
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
    print "Confirm pairing request timed out (";nCtx;")\n"
  endif
endfunc 1
OnEvent  EVBLEMSG              call HandlerBleMsg
```

5.1.4 Event EVBLEMSG, ID BLE_EVBLEMSGID_CONFIRM_TIMED_OUT

New – new code must be added

Use Case	Any application that makes use of Bluetooth LE pairing
Details	An EVBLEMSG event with ID **BLE_EVBLEMSGID_CONFIRM_TIMED_OUT** (31) is called when a pairing confirmation has timed out.

Example

See example BLE_EVBLEMSGID_CONFIRM_PAIRING

5.1.5 Function BlePairingResponse(connHandle, nAccept, nKeyExchange)

New – new code must be added

Use Case	Any application that makes use of Bluetooth LE pairing
Details	This should be used to accept or decline an incoming pairing request. This should be used in response to EVBLEMSG event with ID **EVBLEMSG BLE_EVBLEMSGID_CONFIRM_PAIRING**. New function **`BlePairingInfo()`** can be used to get information about the connecting device. Function **`BleBondingIsTrusted()`** can be used to determine whether a bond should be accepted or declined.

Example

For an example, see EVBLEMSG BLE_EVBLEMSGID_CONFIRM_PAIRING

5.1.6 CFG 214 High Bandwidth Mode

Replace – this requires existing code to be modified

Use Case	Applications for BL65x that set *High bandwidth mode* (CFG 214) ***Note:*** *This configuration was removed from the BL654 firmware in version v29.4.6.6.*
Details	See Event Length

5.1.7 CFG 218 Set Aside Time Per Connection

Replace – this requires existing code to be modified

Use Case	Applications for BL65x that set *Set aside time per connection* (CFG 218)
Details	See Event Length

5.1.8 CFG 219 Event Length

New – new code must be added

Use Case	Applications that want to raise/lower the radio active time. This can be used to increase throughput at the expense of power consumption Applications that maintain multiple active connections
Details	Event Length replaces previous configuration *Set aside time per connection*. It also completely obsoletes earlier configuration *High bandwidth mode*.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

	Event length is the active radio time for a single connection within a connection interval. It is measured in units of 1.25 milliseconds. The minimum value for event length is 2 so the minimum time is 2.5 milliseconds. The maximum value is 800 so the maximum time is 1000 milliseconds. Changing the event length can have several affects: ▪ Raising the event length to the same value as connection interval uses the maximum amount of radio time for that connection. This increases data throughput but also increases power consumption. ▪ Raising the event length above connection interval has no further effect. ▪ Lowering the event length lowers active radio time and therefore also lowers throughput and power consumption. ▪ Lowering the event length allows more radio time for multiple active connections or scanning events. ▪ For situations where a device has more than one active connection, the connection interval should be at least the event length multiplied by the maximum number of active connections. Having a higher event length increases memory required for the desired configuration. See section *Configuration memory* for more details. See Nordic documentation on *Suggested intervals and windows* for Event Length and connection interval recommendations.

Example

In this example, a *Set aside time per connection* value of 7500 µs is replaced with an equivalent event length of 6.

Before:

```
NvCfgKeySet(218, 7500)
```

After:

```
NvCfgKeySet(219, 6)
```

5.1.9 Event EVBLEOVERWRITEBOND, Function BleOverwriteBond()

New – new code must be added

Use Case	Any application that uses Bluetooth LE bonds and wants to protect against intrusion attacks
Details	An event EVBLEOVERWRITEBOND is called when a remote device attempts to bond and specifies a different bonding address than the connection address of a device which already exists in the bond database. This indicates a possible intrusion attack and can be responded to with the **`BleOverwriteBond()`** function to either accept or decline the new bond. ***Note:*** *GATT functionality is restricted until the bond is either accepted or declined. Bluetooth LE services protected by security requirements are not able to be read.*

Example

```
function HandlerBleOverwriteBond(BYVAL Conn, BYVAL Flags, BYVAL Address$)
  //A possible intrusion attack has occurred. Do not accept new bond
  dim rc
  print "\nBond overwritten, Handle: ";integer.h'Conn;", Flags: ";integer.h'Flags;", Address: ";strhexize$(Address$)
  rc = BleOverwriteBond(Conn, 0)
endfunc 1
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

5.2 Recommended Changes

The following changes are recommended for users who want to change default behavior or make use of extra features.

5.2.1 Function BleSecMngrPairConf()

New – new code must be added

Use Case	Any application that makes use of Bluetooth LE pairing and wants to change default behaviour of when to confirm pairing requests
Details	Use this function to change pair confirmation settings. This function is passed one parameter, nPairConfirmation. This parameter is a bitmask of events that triggers a pairing confirmation. Bits 1, 2, 10, and 11 are set by default. The following are the bit values:

0	No confirmation (automatically accept)
Bit 0	Confirm if there is an existing bond with device (central)
Bit 1	Confirm if there is an existing bond with device (peripheral)
Bit 2	Confirm if key exchange less than requested (central)
Bit 3	Confirm if key exchange different (including exchanging additional keys, central)
Bit 4	Confirm if LESC not supported by remote device and configuration is set to use it
Bit 5	Confirm if MITM not supported by remote device and configuration is set to use it
Bit 6	Confirm if OOB not supported by remote device and configuration is set to use it
Bit 7	Confirm if bond was requested but remote device does not support or want to bond (central)
Bit 8	Confirm if remote device did not ask to bond and configuration is set to use bonding (peripheral)
Bit 9	Confirm if remote device maximum encryption key size is less than the maximum set in the configuration
Bit 10	Confirm if the remote device is attempting to overwrite a bond that might be for a different device ***Note:*** *This triggers the EVBLEOVERWRITEBOND event, not the EVCONFIRMPAIRING event*
Bit 11	Modifies existing bond confirmation (bits 1 and 2) so that the confirmation event is not thrown if the device pairing/bonding is using the bond details that already exist in the database
Bits 12- 14	*Reserved.* Should be set to 0.
Bit 15	Always confirm

Example

```
//Set pairing confirmation to all
rc = BleSecMngrPairConf(0x8000)
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

6 TROUBLESHOOTING

6.1 My devices won't pair

Ensure that event EVBLEMSG is implemented and that it is using BlePairingResponse() with event ID BLE_EVBLEMSGID_CONFIRM_PAIRING (30) to confirm pairing.

6.2 My app doesn't compile

Visit the appropriate GitHub page listed below to check that you're using the appropriate applications release with your version of *smart*BASIC firmware. Packages of samples applications for previous releases are available for BL652 [G], BL653/BL653µ [L] and BL654/BL654PA [Q]

6.3 Configuration memory

This release allows more flexibility for checking active device configuration and checking memory resources used.

Configuration memory is affected by event length, maximum packet size, and the maximum number of supported connections. Having all three set high may mean the configuration gets dynamically reduced to not overflow memory. Use the following steps to check if the configuration has been reduced with suggestions to help restore a desired configuration:

- Use sysinfo(2046) to check memory used in the active configuration.
- Use sysinfo(2047) to check memory required for the desired configuration. If sysinfo(2046) is less than sysinfo(2047) then configuration has been reduced to not overflow memory.
- The active value for event length can be checked by using sysinfo(2091)
- Lowering AT+CFG 206 (max connections as central) can be used to reduce memory requirements and keep a desired Event Length.

See section Sysinfo on how to query the active configuration for individual settings.

7 REFERENCES

7.1 Bluetooth Low Energy

Table 4: References for more Bluetooth Low Energy Information

Ref	Link
[A]	Article: *How to Set the Bandwidth on Bluetooth LE Link Connection*
[B]	Nordic Documentation on *Suggested intervals and windows*
[C]	Article: *A Basic Introduction to Bluetooth LE Security*

7.2 BL652

Table 5: Links to additional BL652 information

Ref	Link
[D]	BL652 product page
[E]	BL652 *smart*BASIC extensions manual: To be released with new firmware
[F]	BL652 *smart*BASIC sample applications
[G]	BL652 *smart*BASIC sample applications (previous releases)

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

7.3 BL653/BL653µ

Table 6: Links to additional BL653/ BL653µ information

Ref	Link
[H]	BL653 product page
[I]	BL653µ product page
[J]	BL653/ BL653µ *smart*BASIC extensions manual
[K]	BL653/ BL653µ *smart*BASIC sample applications
[L]	BL653/ BL653µ *smart*BASIC sample applications (previous releases)

7.4 BL654/BL654PA

Table 7: Links to additional BL654/ BL654PA information

Ref	Link
[M]	BL654 product page
[N]	BL654PA product page
[O]	BL654/ BL654PA smartBASIC extensions manual
[P]	BL654/ BL654PA smartBASIC sample applications
[Q]	BL654/ BL654PA smartBASIC sample applications (previous releases)

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610