

User Guide

BL652 Development Kit

Version 1.4

REVISION HISTORY

Version	Date	Notes	Approver
1.0	12 Aug 2016	Initial Release	Jonathan Kaye
1.1	19 Aug 2016	Updates to J12 and J6 pins	Raj Khatri
1.2	6 Sept 2016	Updates to nAutorun settings	Raj Khatri
1.3	14 Oct 2016	Updates to JTAG Signals and wiring	Raj Khatri
1.4	15 Nov 2016	Fixes to vSP (Virtual Serial Port) Modes and OTA (Over the Air) smart BASIC Application Download section.	Raj Khatri

CONTENTS

1	Overview	5
2	Laird BL652 Development Kit Part Numbers	5
3	Package Contents.....	5
4	BL652 Development Kit – Main Development Board.....	6
4.1	Key Features.....	6
5	Understanding the Development Board.....	8
5.1	BL652 Default Configuration and Jumper Settings.....	10
6	Functional Blocks	11
6.1	Power Supply	12
6.1.1	Additional Power Option (for BL652 only) – Coin Cell.....	13
6.2	Reset Button	14
6.3	SWD (JTAG) Interface.....	14
6.4	Four-wire UART Serial Interface	15
6.5	UART Mapping	16
6.5.1	UART Interface Driven by USB.....	16
6.5.2	UART Interface Driven by External Source	16
6.6	nAutoRUN Pin and Operating Modes	18
6.7	vSP (Virtual Serial Port) Modes and OTA (Over the Air) <i>smart</i> BASIC Application Download.....	19
7	Software.....	20
8	Breakout Connector Pinouts.....	21
8.1	J40, J44, J29, J41, J1, J5 SIO (Special Input/Output Sockets) Breakout Connectors	21
8.2	Arduino Connector for Plugging in an Arduino Shields	24
8.2.1	Analog Input Buffer and Attenuator Circuit (U3)	26
8.3	Additional Peripherals/Sensors	27
8.3.1	Temperature Sensor	27
8.3.2	I2C Sensor (RTC Chip)	28
8.3.3	SPI Device EEPROM	29
8.3.4	Push Button and LED Connected to BL652.....	30
8.3.5	NFC External Antenna Connector and NFC Antenna RF Matching Circuit	31
8.3.6	Optional External Serial SPI Flash IC.....	32
8.3.7	Optional 32.76 kHz Crystal	33
9	Other Features.....	34

9.1 Current Consumption Measurement..... 34

10 Appendix 36

10.1 Coin Cell Insertion 36

10.2 Coin Cell Removal 36

11 Additional Documentation..... 37

1 OVERVIEW

The Laird DVK-BL652 development kit provides a platform for rapid wireless connectivity prototyping, providing multiple options for the development of Bluetooth Low Energy (BLE) plus Near Field Communication (NFC) applications.

The Laird BLE development kit is designed to support the rapid development of applications and software for the BL652 series of BLE modules featuring Laird’s innovative event driven programming language – *smartBASIC*. More information regarding this product series including a detailed module user’s guide and *smartBASIC* user guides are available on Laird’s BL652 product page: <http://www.lairdtech.com/products/bl652-ble-module>

2 LAIRD BL652 DEVELOPMENT KIT PART NUMBERS

Part number: DVK-BL652-SA/DVK-BL652-SC

Applicable to the following BL652 module part numbers:

- BL652-SA-xx Bluetooth Smart v4.2 + NFC module – integrated antenna featuring *smartBASIC*
- BL652-SC-xx Bluetooth Smart v4.2 + NFC module – external antenna featuring *smartBASIC* (FCC)

3 PACKAGE CONTENTS

All kits contain the following items:

Development Board	The development board has the required BL652 module soldered onto it and exposes all available hardware interfaces.
Power Options	<ul style="list-style-type: none"> ▪ USB cable – Type A to micro type B. The cable also provides serial communications via the FTDI USB – RS232 converter chip on the development board. ▪ DC barrel plug with clips for connection to external power supply (7-12Vdc) ▪ 3x AAA battery holder fitted on underside of development board ▪ Coin-cell holder (for powering BL652 module only, not the development board) fitted on underside of development board
Two-pin jumpers for pin headers (5)	Five jumpers for 2.54 mm pitch headers used on DVK-BL652 development board.
Fly leads (6)	Supplied to allow simple connection of any BL652 module pin (available on Plated Through Holes on J29, J40, J41, J44, and headers J5, J36 to any Arduino pin (available on Plated Through Holes on J15, J16, J22, J23)
External BLE dipole antenna	External dipole antenna, 2 dBi, 2.4-2.5 GHz (Laird part #0600-00057) with integral RF coaxial cable with 100 mm length and IPEX-4 compatible RF connector.
NFC antenna	Supplied with the DVK-BL652-SC development board only. Laird NFC flexi-PCB antenna.
Web link card	Provides links to additional information including the BL652 user guide, firmware, terminal utilities, schematics, quick start guides, firmware release notes and more. Note: Sample <i>smartBASIC</i> applications are available to download from the Laird BL652 applications GitHub webpage

4 BL652 DEVELOPMENT KIT – MAIN DEVELOPMENT BOARD

This section describes the BL652 development board hardware. The BL652 development board is delivered with the BL652 series module loaded with integrated *smartBASIC* runtime engine firmware. The development board comes with a preloaded sample *smartBASIC* application – *\$autorun\$.devkit.sample.app.sb*. This app can be accessed at the Laird GitHub repository at <https://github.com/LairdCP/BL652-Applications>. Use the BL652 Quick Start Guide for additional information. The guide is accessible from the following link: www.lairdtech.com/BL652-Quick-Start

Note: By default, the module starts running the preloaded development board sample application (*\$autorun\$.devkit.sample.app.sb*) at power-up.

Applications in *smartBASIC* are simple and easy to develop for any BLE application. Sample *smartBASIC* applications scripts are available to download from the Laird GitHub repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications>.

The BL652 development board is a universal development tool that highlights the capabilities of the BL652 module. The development kit is supplied in a default configuration which should be suitable for multiple experimentation options. It also offers a number of header connectors that help isolate on-board sensors and UART from the BL652 module to create different configurations. This allows you to test different operating scenarios. The development board also has support for plugging in 3rd party Arduino Shield boards.

The development board allows the BL652 series module to physically connect to a PC via the supplied USB cable for development purposes. The development board provides USB-to-Virtual COM port conversion through a FTDI chip – part number FT232R. Any Windows PC (XP or later) should auto-install the necessary drivers; if your PC cannot locate the drivers, you can download them from <http://www.ftdichip.com/Drivers/VCP.htm>

4.1 Key Features

The BL652 development board has the following features:

- BL652 series module soldered onto the development board
- The following power supply options for powering the development board:
 - USB (micro-USB, type B)
 - External DC supply (7-12V)
 - AAA batteries (three AAA battery holder fitted on underside of development board)
- Regulated 3.3V for powering the BL652 module. Optional regulated 1.8V for powering the BL652 module via selection switch
- Power supply option for coin-cell (CR2032) operation of the BL652 module ONLY (not development board)
- USB to UART bridge (FTDI chip)
- BL652 UART can be interfaced to:
 - USB (PC) using the USB-UART bridge (FTDI chip)
 - External UART source (using IO break-out connectors J1 when the development board is powered from a DC jack or AAA batteries)
 - Arduino connector by use of an analog switch to route the BL652 UART
- Current measuring options (BL652 module only):
 - Pin header (Ammeter)
 - Current shunt monitor IC (volt meter or oscilloscope)
 - Series resistor for differential measurement (oscilloscope)

- IO break-out 2.54 mm pitch pin header connectors (plated through-holes) that bring out all interfaces of the BL652 module – UART, SPI, I2C, SIO [DIO or AIN (ADCs)], PWM, FREQ, NFC – and allow for plugging in external modules/sensors.
- Pin headers jumpers that allow the on-board sensors(I2C sensor, LEDs, Arduino SPI interface, etc.; and the USB UART FTDI bridge) to be disconnected from BL652 module (by removing jumpers).
- Three on-board sensors:
 - Analog output temperature sensor
 - I2C device (RTC chip)
 - SPI device (EEPROM)
- Two buttons and two LEDs for user interaction
- NFC antenna connector on-board development board for use with supplied flexi-PCB NFC antenna
- Optional external 32.768 kHz crystal oscillator. Not required for operation of the BL652; is disconnected by open solder-bridges by default.
- Optional external serial (SPI) flash IC. Not required for operation of the BL652; is disconnected by open solder-bridges by default.
- One analog buffer (provides a 3.3:1 attenuation) used when an analog source is at 5 volts into development board.
- Arduino connectors – Allow for plugging of Arduino shield boards.

Note: The DVK-BL652 development board is **not** an Arduino shield, but is an Arduino base board (similar to the Arduino UNO).

- Arduino connector test points – All Arduino connector signals brought out to plated through-holes (2.54 mm pitch). Allow any Arduino connector signal (D0-D13 or A0-A5) to be connected to any BL652 module using fly leads for maximum flexibility
- Arduino connector signals wired to BL652 via headers or series resistors
- Arduino connectors orientation at 90 degrees to the development board long dimension, allowing larger Arduino shields to hang off the side of development board so not interfering with the mounted external antenna or the BL652 module (the BLE chip antenna).
- Access to BL652 JTAG – also known as Serial Wire Debug (SWD) Interface
- On-board SWD (JTAG) programmer circuitry
- *smartBASIC* runtime engine FW upgrade capability:
 - Via UART (using the FTDI USB-UART)
 - Via SWD (JTAG) using on-board JTAG programmer circuitry on the DVK-BL652
- *smartBASIC* application upgrade capability:
 - Via UART (using the FTDI USB-UART)
 - Via OTA (Over-the-Air)

5 UNDERSTANDING THE DEVELOPMENT BOARD

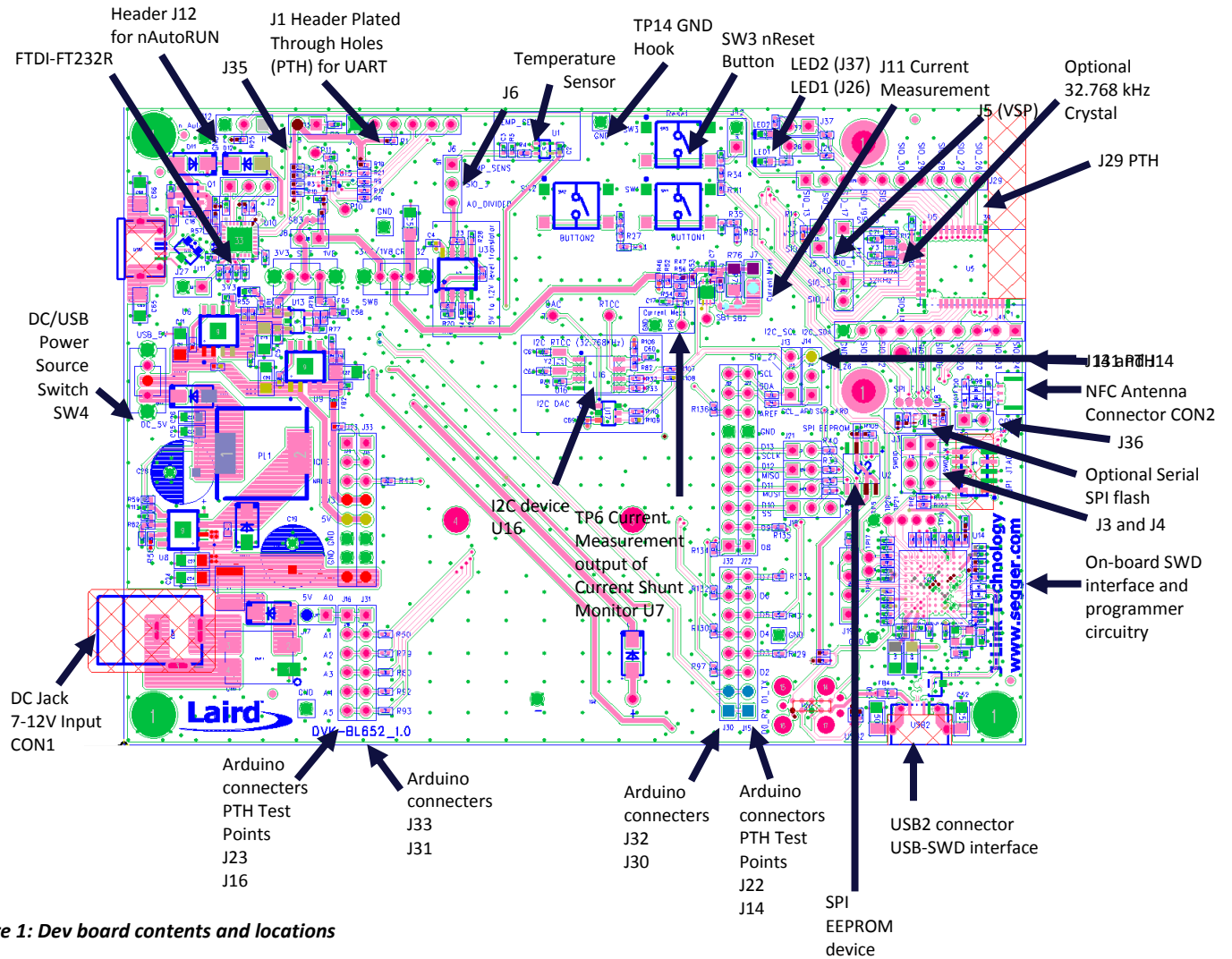


Figure 1: Dev board contents and locations

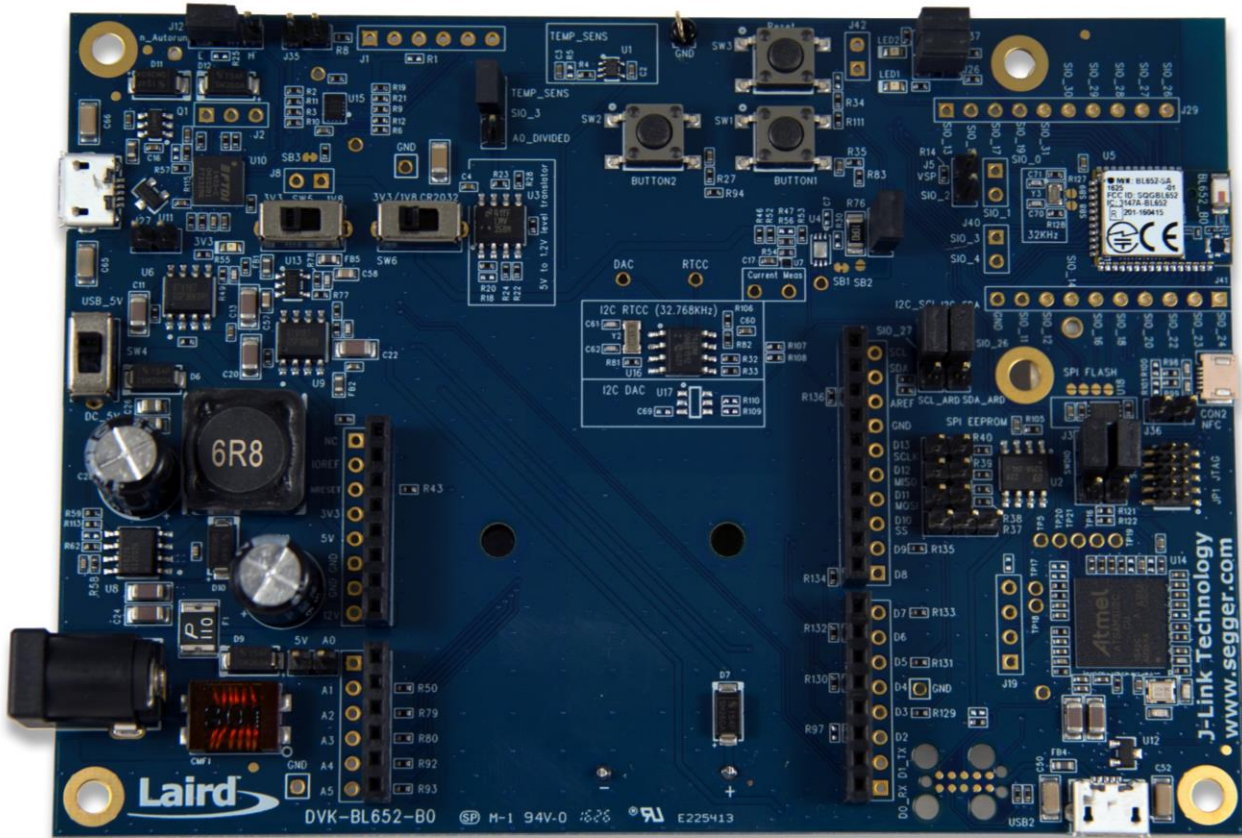


Figure 2: Development board DVK- BL652 (fitted with BL652-SA module for example)

5.1 BL652 Default Configuration and Jumper Settings

Important! To ensure correct out-of-the-box configuration, the BL652 development board must be set according to [Figure 3](#).

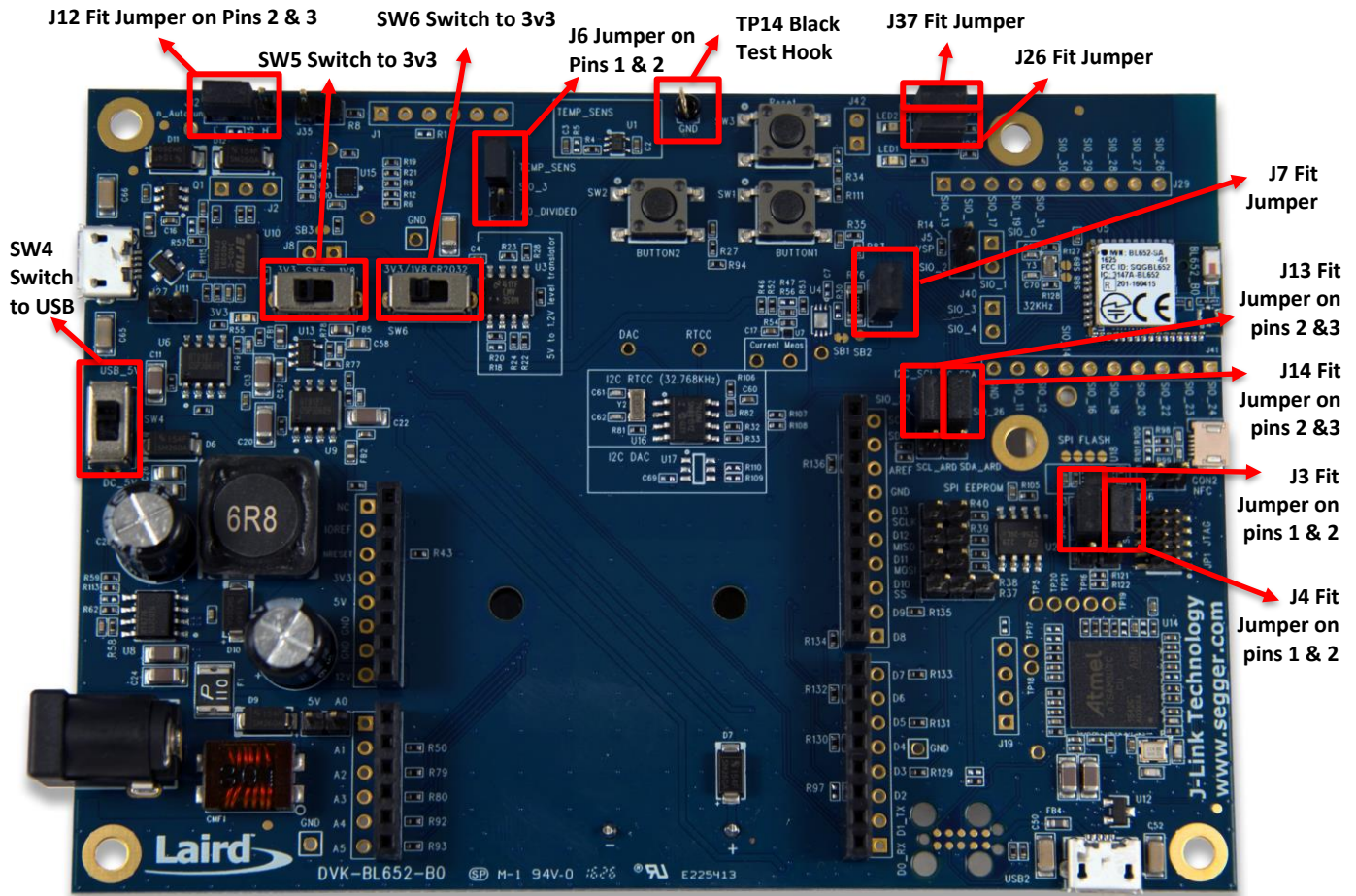


Figure 3: Correct DVK- BL652 development board jumper and switch settings

6 FUNCTIONAL BLOCKS

The BL652 development board is formed by the major functional blocks shown in Figure 4.

PSU Block and Current Measurement Block

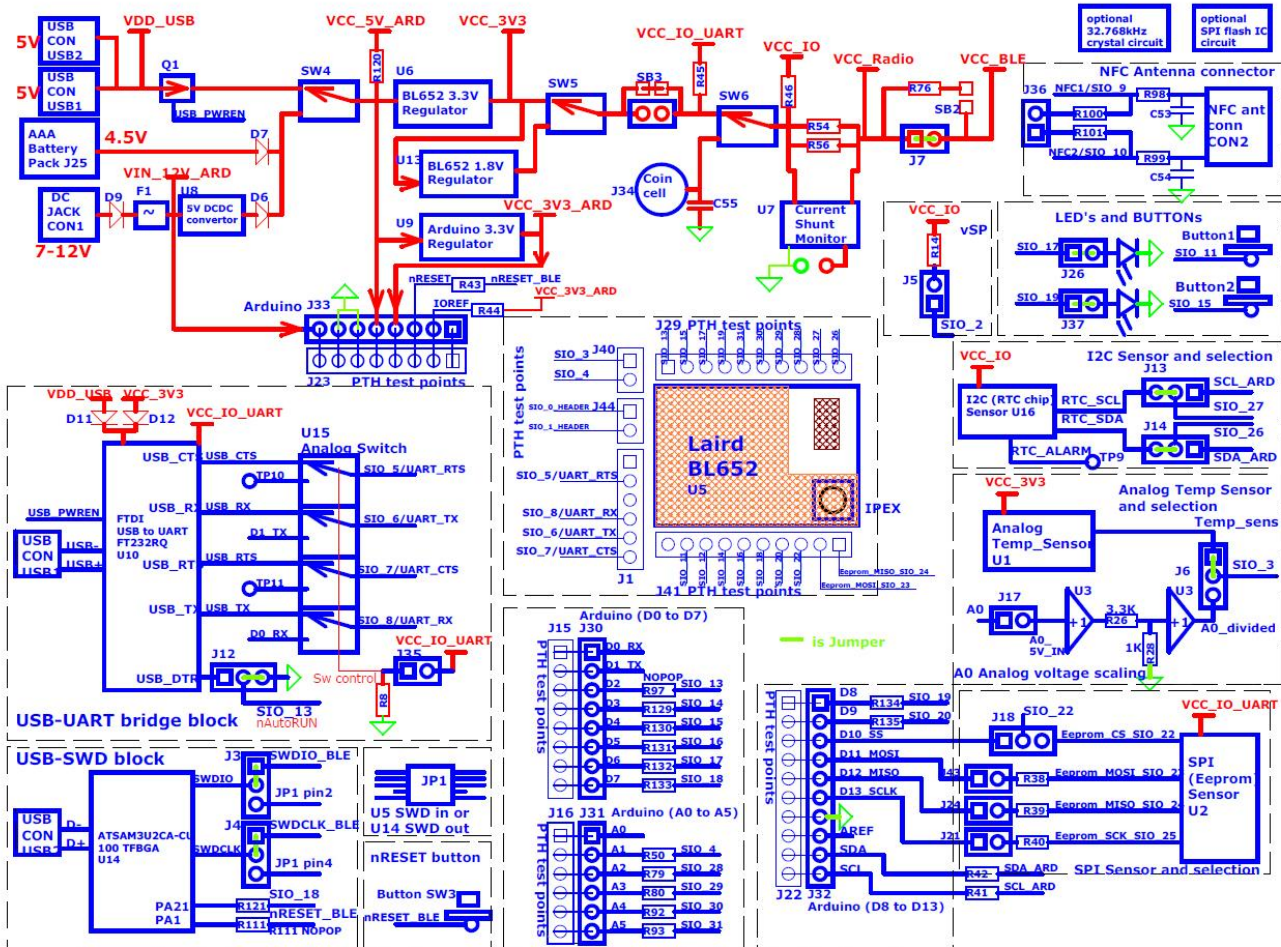


Figure 4: DVK- BL652 block diagram

6.1 Power Supply

Figure 5 shows the DVK- BL652 development board Power Supply block.

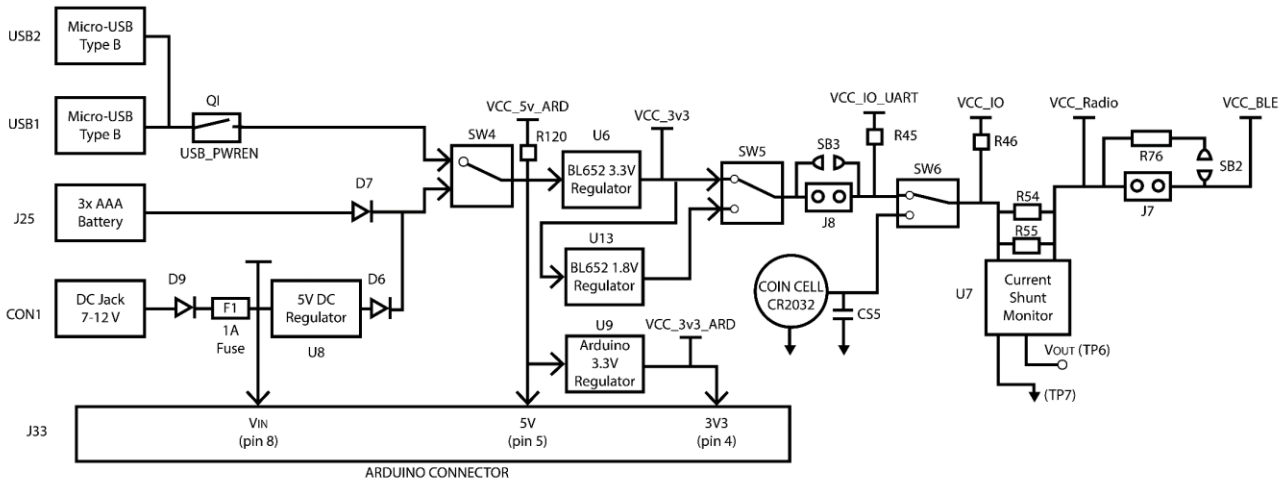


Figure 5: DVK-BL652 power supply

There are three options for powering the development board:

- USB type micro-B connector (USB1)
- External DC supply (7-12V), into DC jack connector (CON1),
- AAA batteries – Three AAA battery holder (J25) fitted on underside of development board

The power source fed into the DC jack (CON1) (which is then regulated by a DCDC to 5V) or three AAA batteries (J25) is combined together through diodes (diode-OR) and fed to the selection switch SW4. SW4 selects the power source between either the USB or the DC jack (5V-regulated)/AAA.

The 5V from the USB or the 5V from DCDC output/AAA batteries is regulated down to 3.3 V with an on-board regulator (U6) on the development board.

The development board also has a 1.8V regulator allowing for the possibility to power the BL652 module from a 1.8V rail.

Switch SW5 selects between the regulated 3.3V and regulated 1.8V. Default position of SW5 is to select regulated 3.3V.

Table 1: Dev board power source and switch positions

Development Board Power Source	Switch Positions		
	SW4	SW5	SW6
USB (USB1)	Position USB	Position 3V3 always	Position 3V3/1V8 always
DC jack (CON1) or AAA battery (J7)	Position DC	Position 3V3 always	Position 3V3/1V8 always
USB (USB2) (Note 1)	Position USB	Position 3V3 always	Position 3V3/1V8 always

Note: The development board DVK-BL652 has on-board circuitry to allow access to BL652 SWD interface (via USB connector USB2). Use USB2 only to power the development board when BL652 SWD interface is needed. Refer to [SWD Interface](#). When USB2 is used, USB1 does not need to be used for DC power.

The CR2032 coin cell voltage is not regulated but is fed directly to the BL652 module supply pin. Switch SW6 selects between the regulated 3V3V/1V8 and coin cell. The coin cell powers only the BL652 module directly (on the development board); this is power domain VCC_Radio and through R46 provides power to power domain VCC_IO.

The Arduino connector (J33) receives the following:

- 7-12V from the DC jack (CON1) directly into the Arduino connector J33 pin 8 (Vin_12V_ARD) via protection diode (D9) and 1A fuse (F1).
- 5V is generated from the on-board DCDC regulator (U8) on the development board into the Arduino connector J33 pin 5 (VCC_5V_ARD). The U8 7-12V input is taken from DC jack (CON1).
- 3.3V generated from a separate regulator (U9) is used to supply the Arduino connector J33 pin 4, 3.3V domain only (VCC_3V3_ARD).

On the development board, the power circuitry is as follows:

- VCC_3V3 – Supplies power to the FTDI chip as well as temperature sensor (U1).
- VCC_IO_UART – Supplies the FTDI chip IO and all other sensors and circuitry.
- VCC_IO – Supplies the I2C RTC chip (U16). The use case for powering this is: The RTC chip can be configured so that, after the pre-determined time, the RTC chip outputs (via RTC_ALARM pin) a transition level that can be used to wake up the BL652 module up from deep sleep.
- VCC_Radio – Supplies the BL652 series module only. Current measuring block – the current shunt monitor IC (U7) – on the development board only measures the current into power domain VCC_Radio (that is current going into header J7 pin1).
- VCC_BLE – supplies the BL652 series module only and is to the current that has come out of the current measuring block on the development board on header connector J7pin2.
- VCC_12V_ARD – Supplies the Arduino connector (J33) only.
- VCC_5V_ARD – Supplies the Arduino connector (J33) only and the Analog buffer IC (U3) attenuator circuit.
- VCC_3V3_ARD – Supplies the Arduino connector (J33) only.

6.1.1 Additional Power Option (for BL652 only) – Coin Cell

The coin cell powers **only** the BL652 module directly via SW6 (on the development board – power domain VCC_Radio) and optionally (through 46, fitted by default) provides power to domain VCC_IO. Jumpers J3 and J4 MUST both be fitted between pins 2 and 3 for coin cell operation of the BL652. Leaving J3 and J4 jumpers fitted between pins 1 and 2 connects the DVK-BL652 on-board SWD (JTAG) circuitry to the BL652, holding the BL652 in SWD (JTAG) mode and increasing the current drawn by the BL652 by ~20uA.

Refer to the [Appendix](#) for the correct method of coin cell insertion and removal.

6.2 Reset Button

The development board has a reset button (SW3). The Reset is active low (SW3 pushed down). To view its location on the board itself, refer to [Figure 2](#).

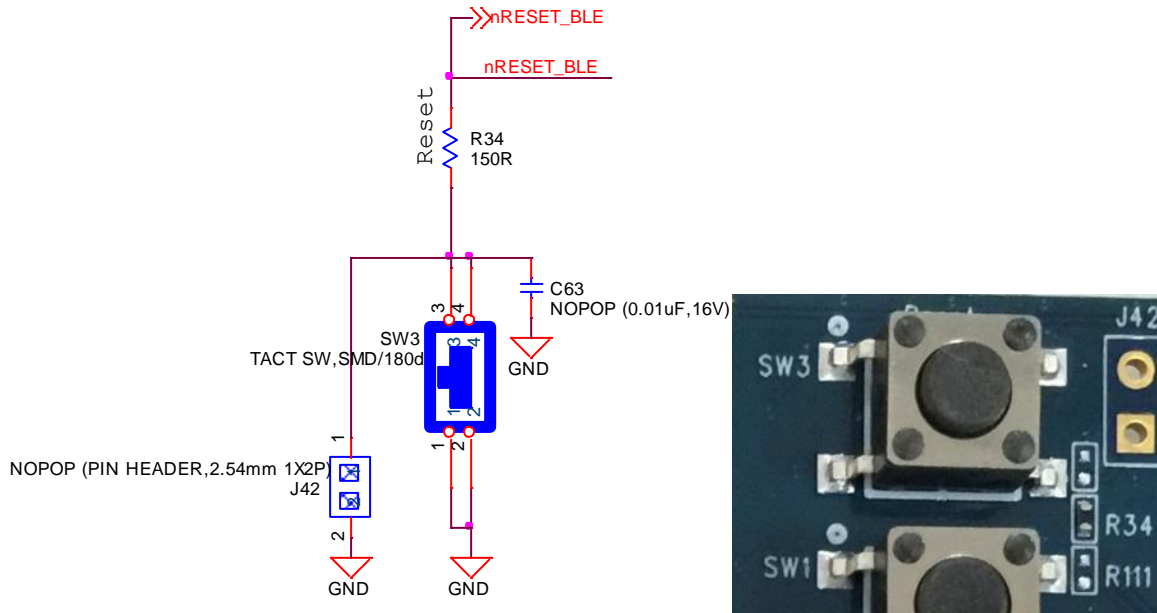


Figure 6: Reset button schematic and location diagram

6.3 SWD (JTAG) Interface

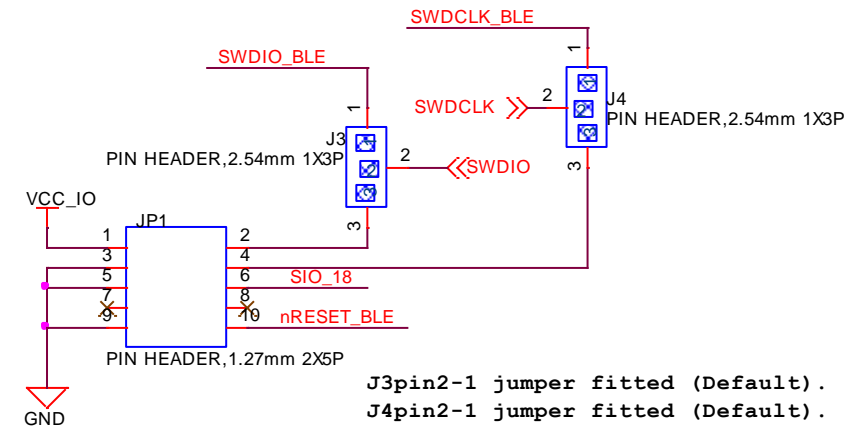
The development board provides access to the BL652 module two-wire SWD interface on JP1. This is REQUIRED for customer use, since the BL652 module supports *smartBASIC* runtime engine firmware over JTAG (as well as over UART)

Laird recommends you use JTAG (2-wire interface) to handle future BL652 module firmware upgrades. You MUST wire out the JTAG (2-wire interface) on your host design (four lines should be wired out, namely SWDIO, SWDCLK, GND and VCC). Firmware upgrades can still be performed over the BL652 UART interface, but this is slower (60 seconds using UART vs. 10 seconds when using JTAG) than using the BL652 JTAG (2-wire interface).

Upgrading *smartBASIC* runtime engine firmware or loading *smartBASIC* applications also can be done using the UART interface.

For those customers (using Nordic SDK) that require access to BL652 SWD (JTAG) interface, the development board (DVK-BL652) (see [Figure 1](#)) has on-board circuitry to allow access to BL652 module SWD interface (via USB connector USB2).

[Figure 7](#) shows the SWD on board circuitry routing via J3 and J4 header connectors. When connector USB2 is used for programming over the SWD (JTAG), J3 and J4 (three-pin headers) jumpers MUST be fitted between pins 1 and 2 for both J3 and J4. This is required to connect the two-wire SWD (JTAG) interface from U14 to the BL652 SWD (JTAG) interface.



SEGGER J-Link Lite Cortex M-9 JTAG/SWD Emulator 10-pin connector. Top view.

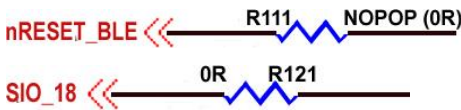
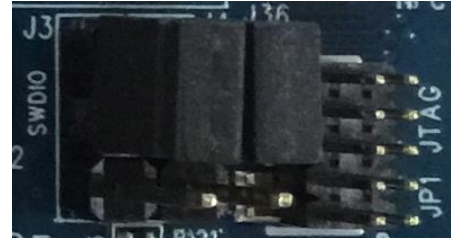


Figure 7: USB to SWD onboard circuitry routing

Table 2 displays the four signals running from Atmel MCU U14 SWD interface (plus SIO_18 and nReset_BLE) to the BL652 module SWD interface (plus SIO_18 and nReset_BLE).

Table 2: USB U4 USB-SWD to BL652 SWD signal routing connections

U4 (Atmel MCU) Net SWD Interface	Route SWD Interface from U4 to BL652 Module (and pin)	Comments
SWDCLK	SWDCLK_BLE (pin 6)	Fit jumper in J4 pin 2-1 (default)
SWDIO	SWDIO_BLE (pin 5)	Fit jumper in J3 pin 2-1 (default)
nRESET_BLE	nRESET_BLE (pin 7)	Via NOT Fitted (default) series resistor R111
SIO.18	SIO_18 (pin 9)	Via Fitted (default) series resistor R121

SIO_18 is a Trace output (called SWO, Serial Wire Output) and is not necessary for programming BL652 over the SWD interface.

nReset_BLE is not necessary for programming BL652 over the SWD interface.

6.4 Four-wire UART Serial Interface

The development board provides access to the BL652 module four-wire UART interface (TX, RX, CTS, RTS) either through USB (via UT10 FTDI USB-UART convertor chip) or through a breakout header connector J1.

Note: The BL652 module provides four-wire UART interface on the HW and the other four signals (DTR, DSR, DCD, RI), which are low bandwidth signals, can be implemented in a *smartBASIC* application using any spare digital SIO pins.

6.5 UART Mapping

The UART connection on the BL652 series module and the FTDI IC are shown in [Table 3](#). [Figure 8](#) explains how the BL652 series module UART is mapped to the breakout header connector J1. These connections are listed in [Table 3](#).

Table 3: SIO/UART connections

BL652 SIO	BL652 Default Function	FTDI IC UART
SIO_6	UART_TX (output)	USB_RX
SIO_8	UART_RX (input)	USB_TX
SIO_5	UART_RTS (output)	USB_CTS
SIO_7	UART_CTS (input)	USB_RTS

Note: Additionally, SIO_13 (the nAutoRUN input pin on the module) can be driven by the USB_DTR output pin of the FTDI chip. This allows testing the \$autorun\$ application on boot without setting the autorun jumper on the development board. Autorun can be controlled directly from Laird’s UWTerminal using the DTR tick box.

6.5.1 UART Interface Driven by USB

- **USB Connector:** The development kit provides a USB Type Micro-B connector (USB1) which allows connection to any USB host device. The connector optionally supplies power to the development kit and the USB signals are connected to a USB-to-serial converter device (FT232R) when SW4 is set to the USB position.
- **USB – UART:** The development kit is fitted with a (U10) FTDI FT232R USB-to-UART converter which provides USB-to-Virtual COM port on any Windows PC (XP or later). Upon connection, Windows auto-installs the required drivers. For more details and driver downloads, visit the following website: <http://www.ftdichip.com/Products/FT232R.htm>.
- **UART Interface Driven by USB FTDI Chip:** In normal operation, the BL652 UART interface is driven by the FTDI FT232R USB-to-UART converter.

6.5.2 UART Interface Driven by External Source

- **UART Interface Driven by External UART Source:** The BL652 module UART interface (TX, RX, CTS, RTS) is presented at a 2.54 mm (0.1”) pitch header (J1). To allow the BL652 UART interface to be driven from the breakout header connector (J1), the following must be configured:
 - The development board must be powered from a DC jack (CON1) or AAA batteries (J25) and with switch SW4 in DC position.
 - The FTDI device must be held in reset. This is achieved automatically by removal of the USB cable (from connector USB1), placing SW4 in the DC position or fitting a jumper on J27.
 - Fit a jumper on J35 (to switch the Analog switch U15 and route BL652 UART to J1) when connecting an external UART source (for example FTDI USB-UART TTL (3.3V) converter cable) using J1. This isolates the BL652 UART from the on-board USB-UART FTDI device. By default, the jumper on J35 is not fitted, so by default BL652 UART is routed to U10 FTDI FT232R USB –UART converter.

Note: The BL652 UART signal levels always need to match the supply voltage, VCC_Radio, of the BL652.

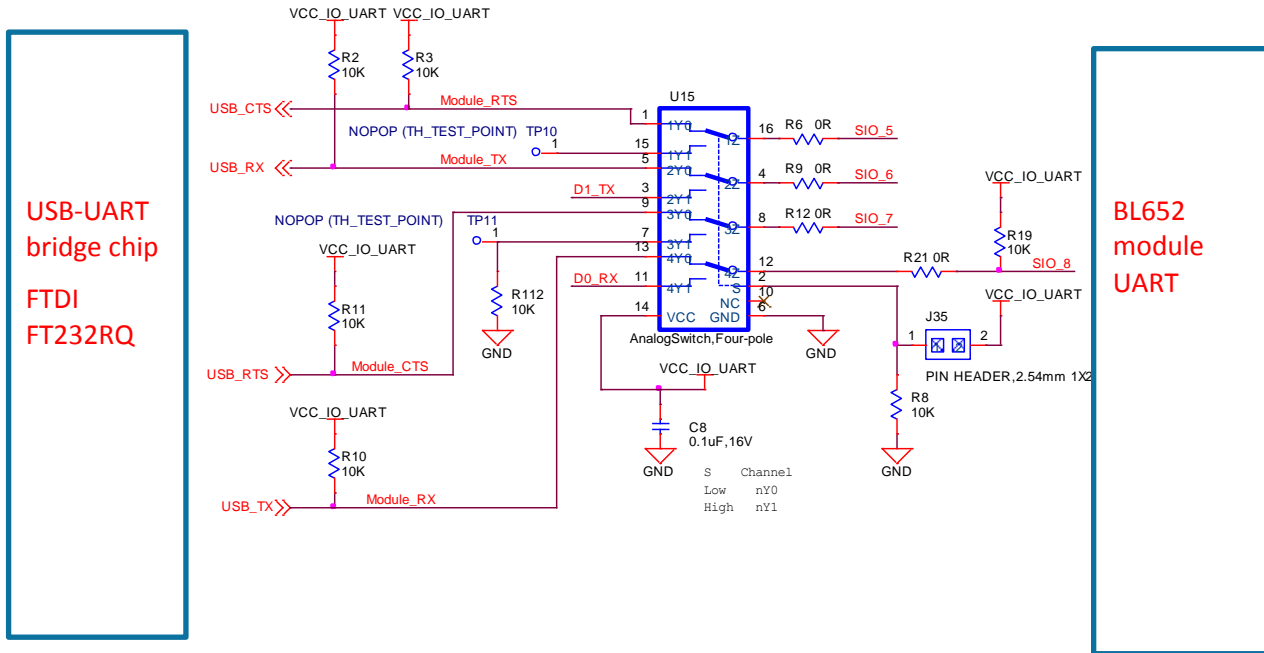


Figure 8: USB to UART interface and header to UART interface

J1 pinout is designed to be used with FTDI USB-UART TTL (3.3V) converter cables (found at <http://www.ftdichip.com/Products/Cables/USBTTLSerial.htm>). One example is FTDI part TTL-232R-3V3.

If the BL652 on the development board is powered from 1.8V supply, then the 1.8V version of the FTDI USB-UART cable would need to be used. UART signal levels always need to match the supply voltage, VCC_Radio, of the BL652.

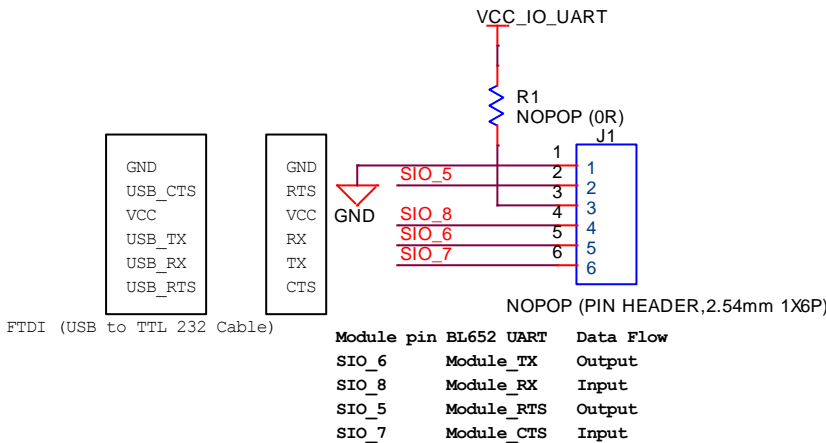


Figure 9: J1 wiring to match FTDI USB-UART cable (TTL-232R-3V3 cable)

Fit a jumper in J35 (to switch the Analog switch U15 and route BL652 UART to J1) when connecting an external FTDI USB-UART TTL (3.3V) converter cable using J1.

Fitting a jumper in J35 also allows the BL652 UART to be routed to Arduino connector (J30).



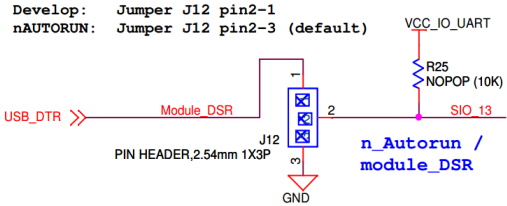
6.6 nAutoRUN Pin and Operating Modes

On the development board, the USB_DTR output (FTDI chip U10) from the PC is wired to BL652 module pin SIO_13 (pin 28) which is the nAutoRUN pin.

Note: *smartBASIC* runtime engine FW checks for the status of nAutoRUN during power-up or reset. The nAutoRUN pin detects if the BL652 module should power up into Interactive/Development Mode (3.3 V) or Self-contained Run mode (0V). The module enters Self-contained Run mode if the nAutoRUN pin is at 0V and an application called \$autorun\$ exists in the module's file system, then the *smartBASIC* runtime engine FW executes the *smartBASIC* application script automatically; hence the name *Self-contained Run mode*.

Tying nAutoRUN to 3.3V inhibits the \$autorun\$ application from running. As an alternative to using USB_DTR, the J12 three-pin header allows a jumper to be fitted to select between the two operating modes.

Table 4: BL652 nAutoRUN header

nAutoRUN Pin	BL652 Operating Mode (pin28, nAutoRUN Mode/SIO_13)		
	Interactive/Development Mode (SIO_13 set High Externally)	Self-contained Run Mode (nAutoRUN mode) (SIO_13 Low Internally)	Circuit
J12 Jumper Position			<p>Develop: Jumper J12 pin2-1 nAutoRUN: Jumper J12 pin2-3 (default)</p> 
Develop Jumper on J12 pins 2-1	nAutoRUN (default) Jumper on J12 pins 2-3 BL652 has internal pull-down enabled, jumper in J12 in 2-3 can also be left off		

The J12 header connector allows the USB_DTR signal from the FTDI chip to be disconnected from the BL652.

To connect the BL652 nAutoRUN pin SIO_13 (pin 28) to PC FTDI USB_DTR line via the J12 header connector, do the following:

- Fit the jumper into the J12 (pin 2-1) header connector to allow the PC (using UwTerminal) to control nAutoRUN pin (SIO_13).

To disconnect the BL652 nAutoRUN SIO_13 (pin 28) from the PC FTDI USB_DTR line, do the following:

- Remove the jumper on header connector J12 pin 2-1. Then nAutoRUN can be controlled by inserting the jumper onto J12 (pin 2-3) as shown in Table 4 (this is the default). The BL652 by default has pull-down enabled on the SIO_13 (nAutoRUN) pin, so the jumper into J12 (pin 2-3) is optional.

6.7 vSP (Virtual Serial Port) Modes and OTA (Over the Air) *smart*BASIC Application Download

The OTA feature makes it possible to download *smart*BASIC applications over the air to the BL652. To enable this feature, SIO_2 must be pulled high externally.

On the development board, header connector J5-pin1 brings out the BL652 SIO_2; J5-pin 2 brings out VCC_IO. To pull BL652 SIO_2 high (to VCC_IO), fit jumper into header J5.

Note: When SIO_2 is high, ensure that SIO_13 (nAutoRun) is NOT high at same time, otherwise you cannot load the *smart*BASIC application script.

This section discusses VSP Command mode through pulling SIO_2 high and nAutoRUN low. Refer to the documentation tab of the **BL652 product page** <http://www.lairdtech.com/products/bl652-ble-module>.

Figure 10 shows the difference between VSP Bridge to UART mode and VSP Command mode and how SIO_02 and nAutoRUN must be configured to select between these two modes.

- **VSP Bridge to UART mode** takes data sent from phone or tablet (over BLE) and sends to BL652 to be sent out of the BL652 UART (therefore data not stored on BL652).
- **VSP Command mode** takes data sent from phone or tablet and sends to BL652 which will interpret as an AT command and response will be sent back. The OTA Android or iOS application can be used to download any *smart*BASIC application script over the air to the BL652 because a *smart*BASIC application is downloaded using AT commands.

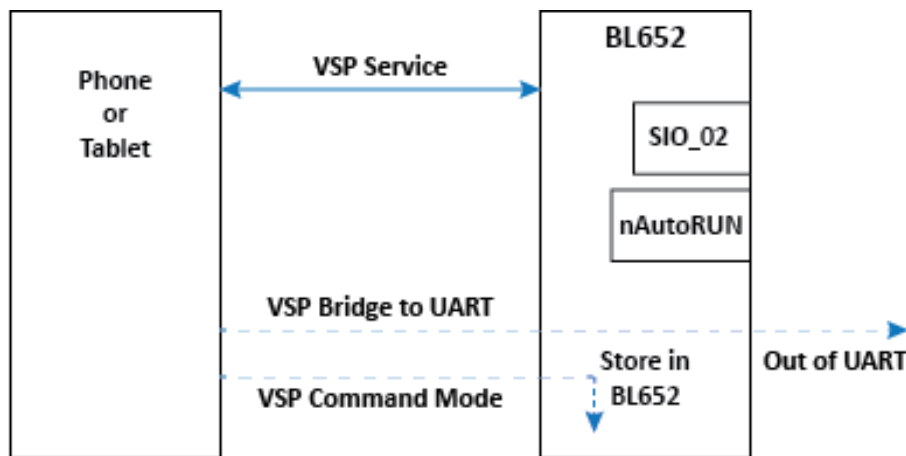


Figure 10: Differences between VSP bridge to UART mode and VSP Command mode

Table 5: vSP modes

Mode	SIO_02 and Jumper position J5	nAutoRUN (SIO_13) and Jumper position J12
VSP Bridge to UART mode	High by fitting jumper in J5	High by fitting jumper in J12 pin 2-1 and untick DTR box in UwTerminal
VSP Command mode	High by fitting jumper in J5	Low by fitting jumper in J12 pin 2-3 and tick DTR box in UwTerminal (the DTR box is ticked by default in UwTerminal)

SIO_02 High (externally) selects the VSP service. When SIO_02 is High and nAutoRUN is Low (externally), this selects VSP Command mode. When SIO_02 is High and nAutoRUN is High (externally), this selects VSP Bridge to UART mode.

When SIO_02 on module is set HIGH (externally), VSP is enabled and auto-bridged to UART when connected. However, for VSP Command mode, auto-bridge to UART is not required. With SIO_02 set to High and nAutoRUN set to Low, the device enters VSP Command mode and you can then download the *smartBASIC* application onto the module over the air from the phone (or tablet).

7 SOFTWARE

The development board connects the BL652 module to a virtual COM port of a PC or other device. From a PC, you can communicate with the module using Laird’s UwTerminal application (version 7.20 or newer for Windows) or UwTerminalX (a cross platform equivalent of UwTerminal available for Windows, Mac, and Linux). Both utilities allow connection to serial devices using any combination of the communications parameters listed in Table 6.

Table 6: UwTerminal/UwTerminalX communication parameters for BL652

Port (Windows)	1 to 255
Port (Mac/Linux)	Any /dev/tty device (UwTerminalX only)
Baud Rate	1200 to 921,600 Note: Baud rate default is 115200 for BL652.
Parity	None
Data Bits	8
Stop Bits	1
Handshaking	None or CTS/RTS

Note: Baud rates higher than 115200 depend on the COM port capabilities of the host PC and may require an external USB – RS232 adapter or ExpressCard – RS232 card

The benefits of using UwTerminal/UwTerminalX include the following:

- Continually displayed status of DSR, CTS, DCD, and RI
- Direct control of DTR on the host PC via a check box
- Direct control of RTS, if CTS / RTS Handshaking is disabled when UWTerminal is launched
- Sending UART BREAK signals. Following provides explanation UART Break.
(https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter#Break_condition)
- BASIC tab provides standalone testing and development of *smartBASIC* applications and allows UwTerminal operation to be automated (UwTerminal only)
- Additional built-in features (right click in Terminal tab screen) to accelerate development including Automation and various XCompile / Load / Run options for downloading *smartBASIC* applications into the BL652.

Note: Full details on *smartBASIC* are available in the *smartBASIC* User Guide available at the Laird product page for BL652 (<http://www.lairdtech.com/products/bl652-ble-module>) along with a document giving a basic introduction to UwTerminal. A help file is included with UwTerminalX that gives an overview of the program.

Tip: If the module returns a four hex digit error code: In UwTerminal, select those four digits, right-click, and select **Lookup Selected ErrorCode** (select **Lookup Selected Error-Code (Hex)** if using UwTerminalX). A description of the error is then printed on screen.

The following are the differences between UwTerminal and UwTerminalX:

- UwTerminal is Windows only whilst UwTerminalX is cross platform
- UwTerminal and UwTerminalX have the same overall functionality but some UwTerminal functions are not available in UwTerminalX including File Player, BASIC tab, multiple file downloading at once, and communication over TCP port.

8 BREAKOUT CONNECTOR PINOUTS

8.1 J40, J44, J29, J41, J1, J5 SIO (Special Input/Output Sockets) Breakout Connectors

Access to all 32 BL652 series module signal pins (SIO's = Signal Input /Output) is available on header connectors J40, J44, J29, J41, J1, J5 (2.54 mm pitch headers).

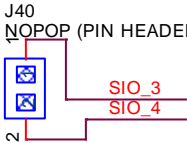
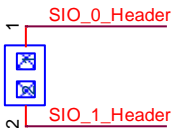
Note: The BL652 module signal pins designation SIO (Signal Input /Output).

- DEFAULT type is DIO (Digital Input or Output) or UART (on fixed pins)
- ALTERNATE type is either AIN (Analog Input ADC), I2C, SPI, DIO (on fixed pins), PWM, FREQ, and NFC
- DIO or AIN functionality is selected using the GpioSetFunc() function in *smartBASIC*
- I2C, UART, SPI controlled by xxxOPEN() functions in *smartBASIC*
- SIO_5 to SIO_8 are DIO by default when \$autorun\$ app runs on power up
- SIO_9 and SIO_10 are NFC pins by default, can be set to alternative function SIO using the GpioSetFunc() function in *smartBASIC*

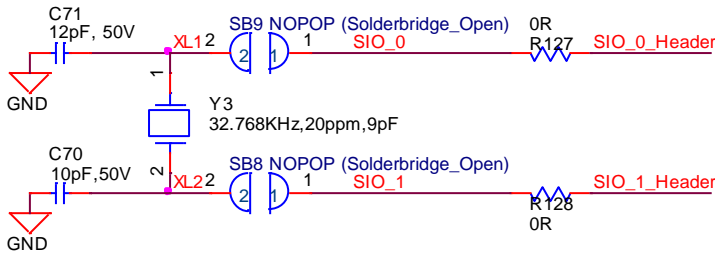
These breakout connectors can interface to a wide array of sensors, the BL652 is user configurable through the *smartBASIC* application script to change each SIO pin from the default function (DIO, UART) to alternate functions (AIN (ADC), I2C, SPI, DIO), PWM, FREQ, and NFC. The BL652 development kit incorporates additional fly-lead cables inside the box to enable simple, hassle-free testing of these multiple interfaces.

Table 7 shows the BL652 module pins that are brought out to plated through Holes (suitable for 2.54 mm pitch headers).

Table 7: Module pins exposed by plated through holes

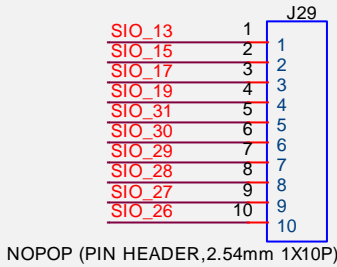
Plated Through Holes or Header Connector		BL652 Module Signals Exposed
J40 	J40 NOPOP (PIN HEADER, 2.54mm 1X2P)	BL652 pin plated holes for access
		SIO_3 SIO_4
J44 	J44 NOPOP (PIN HEADER, 2.54mm 1X2P)	SIO_0
		SIO_1

Plated Through Holes or Header Connector | **BL652 Module Signals Exposed**



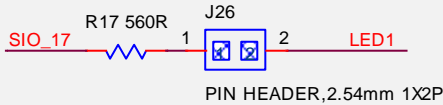
J44 connects to SIO_0 and SIO_1 via OR resistors R127 and R128. By default, the optional external 32.768 kHz crystal circuit is not connected to BL652 as SB8 and SB9 are open.

J29



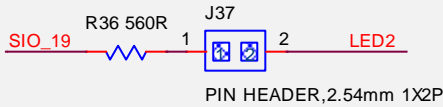
BL652 pin plated holes for access
SIO_13
SIO_15
SIO_17
SIO_19
SIO_31
SIO_30
SIO_29
SIO_28
SIO_27
SIO_26

J26



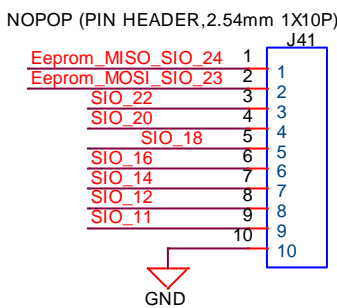
J26 Connects SIO_17 to LED1
J26 jumper fitted (default).

J37



J37 Connects SIO_17 to LED2
J37 jumper fitted (default).

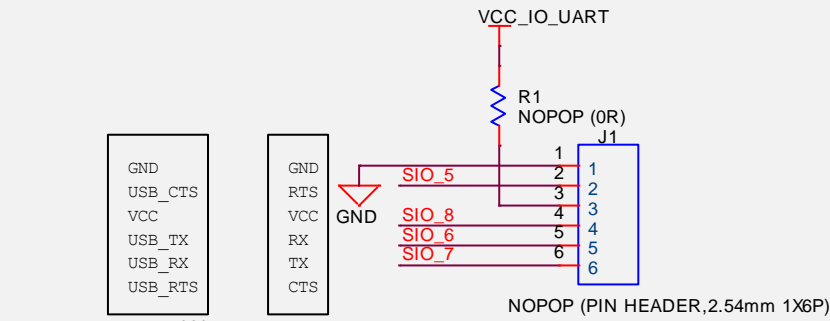
J41



BL652 pin plated holes for access
SIO_24
SIO_23
SIO_22
SIO_20
SIO_18
SIO_16
SIO_14
SIO_12
SIO_11

Plated Through Holes or Header Connector **BL652 Module Signals Exposed**

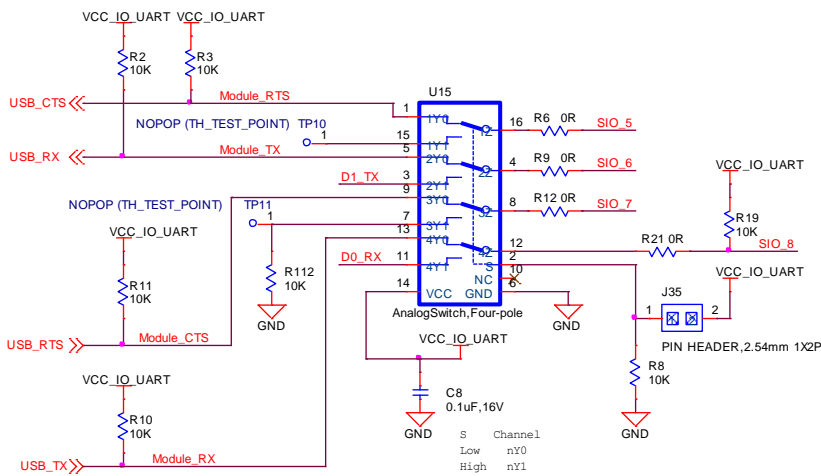
J1



Module pin	BL652 UART	Data Flow
SIO_6	Module TX	Output
SIO_8	Module RX	Input
SIO_5	Module RTS	Output
SIO_7	Module CTS	Input

Serial Port plated holes for access

J35



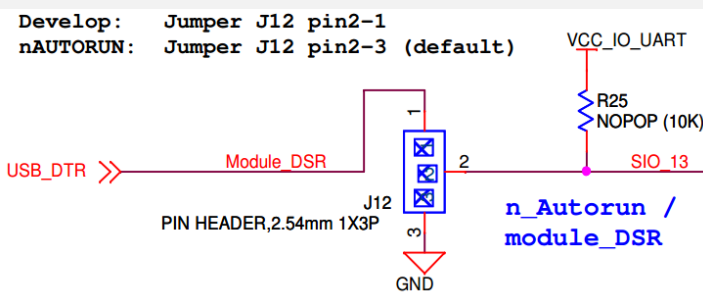
Jumper in J35 selects between BL652 UART routed to FTDI or Arduino:

- No Jumper on J35 (default)**
- Routes SIO_5 (RTS) to FTDI CTS
- Routes SIO_6 (TX) to FTDI RX
- Routes SIO_7 (CTS) to FTDI RTS
- Routes SIO_8 (RX) to FTDI TX

Jumper on J35

- Routes SIO_5 (RTS) to TP10
- Routes SIO_6 (TX) to Arduino D1_TX
- Routes SIO_7 (CTS) to TP11
- Routes SIO_8 (RX) to Arduino D0_RX

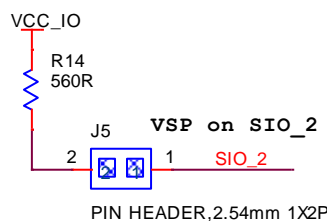
J12



Develop: Jumper J12 pin2-1
nAUTORUN: Jumper J12 pin2-3 (default)

Connects SIO_13 (nAutoRUN) to FTDI DTR
Default jumper fitted in J12 pin 2-3

J5



SIO_2
Can be used to pull-up SIO_2 to VCC_IO

Default No Jumper fitted on J5 SIO_2

Plated Through Holes or Header Connector	BL652 Module Signals Exposed
<p>J13</p> <p>J14</p> <p>J13pin2-3 jumper fitted (Default). J14pin2-3 jumper fitted (Default).</p>	<p>J13 Routes SIO_27 (I2C_SCL) to RTC_SCL device or Arduino_SCL pin J13 pin 2-3 jumper fitted (default). J14 Routes SIO_26 (I2C_SDA) to RTC_SDA device or Arduino_SDA pin J14 pin 2-3 jumper fitted (default).</p>
<p>J6</p> <p>Temp Sensor: Jumper J6 pin2-1 (default)</p>	<p>J6 routes SIO_3 to Temp Sensor or the Analog Buffer attenuator (5V to 1.2V) output J6 pin 2-1 jumper fitted (default)</p>

8.2 Arduino Connector for Plugging in an Arduino Shields

The DVK-BL652 development board is NOT an Arduino Shield, but is an Arduino base board (like the Arduino UNO).

The four Arduino connectors (J30, J31, J32 and J33) on the development board allow Arduino Shields to be plugged in.

- All Arduino connector signals are brought out to Plated-through Holes (2.54mm pitch) J15, J16, J22, J23. This allows any Arduino connector signal (D0-D13 or A0-A5) to be connected to any BL652 module SIO using fly leads for maximum flexibility.
- All Arduino connector signals (D0-D13 and A0-A5) are connected to the BL652 module via series resistors (560R), allowing easy disconnection. Table 6 shows the Arduino connector signals and mapping to BL652 module SIO pins.
- Arduino connectors orientation are at 90 degrees perpendicular to the long dimension, allowing larger Arduino Shields to hang off side of the board without interfering with a mounted external antenna or the BL652 module (the BLE chip antenna).

There are Arduino pins that may be used for special cases on the development board:

- Arduino pin IOREF** on development board (J33 pin2), is connected to 3.3V domain (VCC_3.3V_ARD) via 0R resistor (R44). Arduino IOREF allows Arduino shields to adapt to the voltage provided from the board, Since DVK-BL652 is sending 3.3V up (from the development board) to the IOREF, the Arduino documentation states that a properly configured Shield should respect our logic levels as a function of this pin.

Note: BL652 module PINS DO NOT SUPPORT 5V IO. Do not connect greater than 3.3V IO from Arduino Shields or others as DVK-BL652 does not have level translators.

If accidentally a shield with 5V IO were plugged in, there are series resistors on the DVK-BL652 on all Shield IO lines to provide very limited protection against an inappropriate logic level (something greater than 3.3V).

These series resistors provide the voltage drop as current flows through, activating the ESD protection diode in the BL652 module.

- **Arduino RESET** pin on development board (J33 pin3), is connected to BL652 nRESET pin (U5 pin22) via 0R resistor (R43).
- **Arduino AREF** is wired out from Arduino connector J32 to plated through holes on J22, which is next to the Arduino shield connector J32, and is also wired to SIO_2 via R136 (560R). AREF is supplied by a Shield board and is an input to the Arduino base board to indicate the maximum expected value of the analog signal. The BL652 module does not support this function.
- **Arduino D2 pin** wired out from Arduino connector J32 to plated through holes on J22, which is next to the Arduino shield connector J32. Arduino D2 pin is not wired to BL652 (SIO_13) as series resistor R97 is not fitted.

Table 8: Arduino connectors signals and mapping to BL652 SIO signals

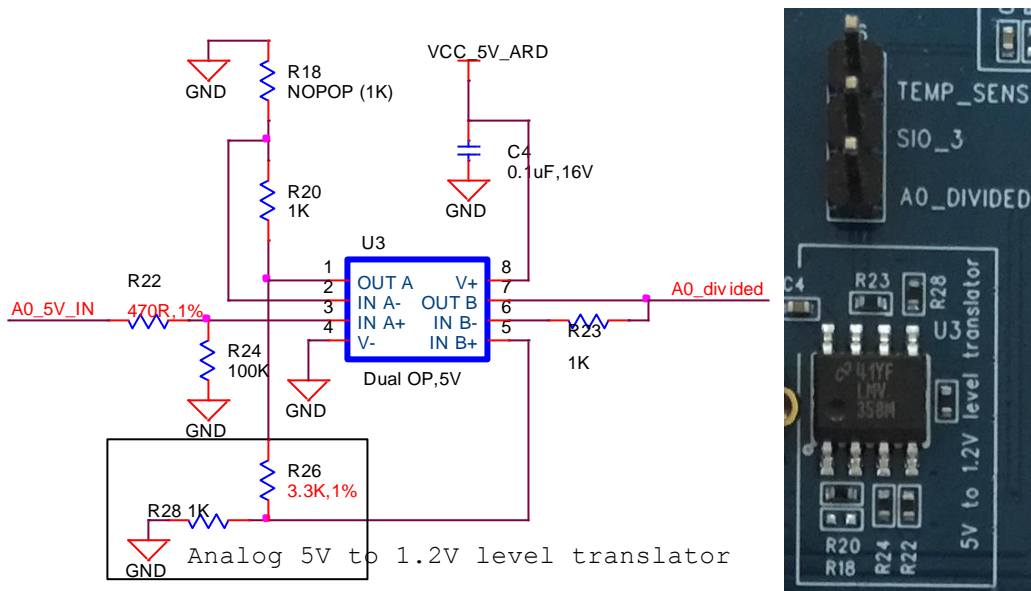
J#	Arduino Connectors and Plated Through Holes (Test points)	Arduino signals
J30	<p>NOPOP (560R) R97 SIO_13 R129 560R SIO_14 R130 560R SIO_15 R131 560R SIO_16 R132 560R SIO_17 R133 560R SIO_18</p> <p>D0 (RX) D1 (TX) D2 D3 D4 D5 D6 D7</p> <p>HEADER, FEMALE, 2.54mm, 1X8P NOPOP (PIN HEADER, 2.54mm 1X8P)</p>	<p>Arduino female header J30.</p> <p>J15 is plated through holes for accessing signals on J30</p>
J29	<p>SIO_31 R93 560R A5 SIO_30 R92 560R SIO_29 R80 560R A4 SIO_28 R79 560R A3 SIO_4 R50 560R A2 A1 A0</p> <p>HEADER, FEMALE, 2.54mm, 1X6P</p> <p>NOPOP (PIN HEADER, 2.54mm 1X6P)</p>	<p>Arduino female header J31.</p> <p>J16 is plated through holes for accessing signals on J30.</p> <p>J17 Connects Arduino A0 pin to Analog Input Buffer (U3). J17 jumper NOT fitted (default).</p>
J32	<p>SIO_19 R134 560R D8 SIO_20 R135 560R D9 SIO_2 AREF R136 560R Eeprom_CS_SIO_22 R37 560R Eeprom_MOSI_SIO_23 R38 560R Eeprom_MISO_SIO_24 R39 560R Eeprom_SCK_SIO_25 R40 560R SDA_ARD R42 560R SCL_ARD R41 560R</p> <p>J18 PIN HEADER, 2.54mm 1X3P</p> <p>J32 D8 D9 D10 SS D11 MOSI D12 MISO D13 SCLK</p> <p>J22 D8 D9 D10 SS D11 MOSI D12 MISO D13 SCLK AREF SDA SCL</p> <p>HEADER, FEMALE, 2.54mm, 1X10 NOPOP (PIN HEADER, 2.54mm 1X10P)</p>	<p>Arduino female header J32.</p> <p>J22 is plated through holes for accessing signals on J30.</p> <p>J18 Connects SIO_22 (SPI CS) to EEPROM (U2) or to Arduino D10 (for use as SPI Slave Select). J18 jumper NOT fitted (default)</p> <p>J43 Connects SIO_23 (SPI MOSI) to Arduino D11. J43 Jumper NOT fitted (default).</p>
J21		
J24		
J43		
J18		

J#	Arduino Connectors and Plated Through Holes (Test points)	Arduino signals
		<p>J24 Connects SIO_24 (SPI MISO) to Arduino D12. J24 Jumper NOT fitted (default).</p> <p>J21 Connects SIO_25 (SPI SCK) to Arduino D13. J21 Jumper NOT fitted (default).</p>
J33		<p>Arduino female header J33.</p> <p>J23 is plated through holes for accessing signals on J33.</p>

8.2.1 Analog Input Buffer and Attenuator Circuit (U3)

Figure 11 shows the Analog Buffer circuit that accepts a 0V to 5V analog input signal from Arduino shield pin A0 and scales it down to an acceptable range of 0V to 1.2V set by potential divider R26 (3.3 kOhms) and R28 (1 kOhms) with a gain of 0.23(=1/ (1+3.3)).

Max Input (Volts) (A0_5V_IN)	R26 (kOhm)	R28 (kOhm)	Output (Volts) (A0_divided)
5	3.3	1	1.16



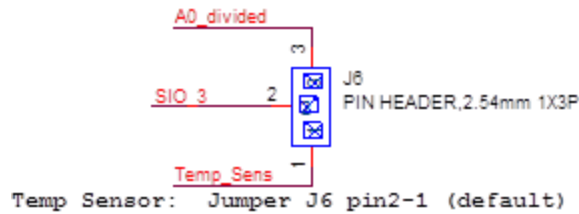


Figure 11: Analog Buffer schematic and PCB

The selection jumper on J6 connects either Temperature Sensor analog output or the output of the Arduino analog input buffer to SIO_3. By default, no jumper is fitted in J6 in either position. To select the Arduino analog input buffer to SIO_3, fit the jumper on J7 to short pins 2-3.

8.3 Additional Peripherals/Sensors

The BL652 development board provides for simple and hassle free connectivity to a wide range of sensors, but also includes several on-board sensors and options to enable a developer to test functionality straight out of the box.

In the *smartBASIC* application code written to use sensors on the development board (including the Temperature sensor (U1) – analog output, Analog Input Buffer (U3) – analog output, SPI EEPROM (U2), I2C RTC chip (U16), LED1(D1), LED2(D2) Button1(SW1), and Button2(SW2)) the SIO pins direction and type must be set in the *smartBASIC* application to override the defaults in the BL652 firmware.

For more information on these sample applications, see GitHub *smartBASIC* sample applications repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications>

8.3.1 Temperature Sensor

The temperature sensor (U1) by default is connected to the BL652 module as jumper on J6 pin 2-1, bridges TEMP_SENS and SIO_3.

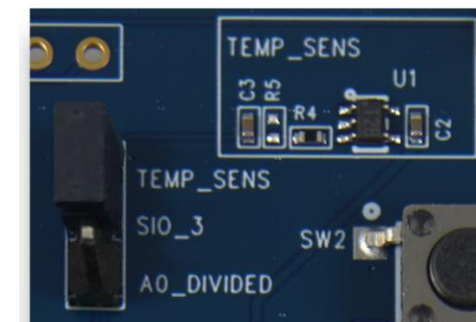
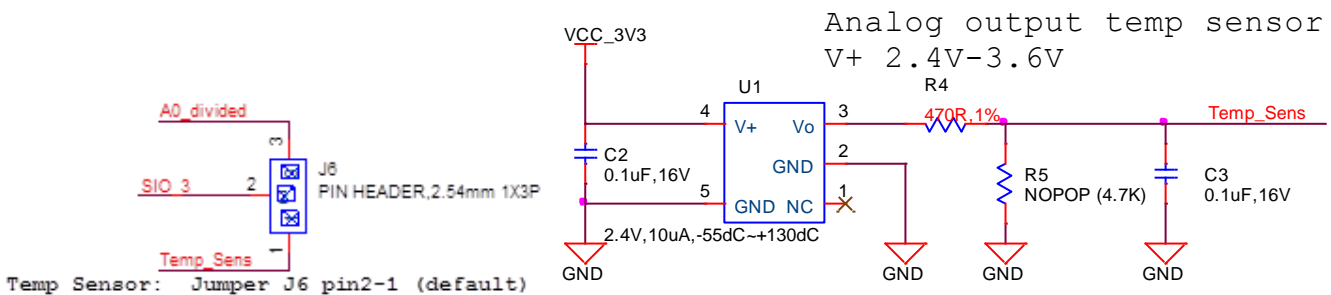


Figure 12: Temperature sensor schematic and PCB

The on-board temperature sensor (TI LM20BIM7 - www.ti.com/lit/ds/symlink/lm20.pdf) has an Analogue output that can be connected to BL652 module pin SIO_3; but since the LM20BIM7 has an analogue output, the BL652 module SIO_3 digital pin (DIO) must be configured as AIN analogue input (ADC). To configure the SIO_3 pin from DIO pin to Alternate function AIN, see the example file “*ts.temperature.sensor.sb*” in the GitHub *smartBASIC* sample applications repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications>

Key specifications of the LM20BIM7 are as follows in [Table 9](#).

Table 9: LM20BIM7 Specifications

Output type	Analogue output
Accuracy at 30°C	±1.5°C ±4°C (max)
Accuracy at 40°C to +85°C	approx. ±2.5°C ±5°C (max)
Power supply voltage range	+2.4 V to 5.5 V
Current Drain	10 uA (max)
Output impedance	160 Ohms (max)

The LM20BIM7 datasheet states the relationship of Temperature (T) to Voltage output (Vo) can be approximated as a linear equation (for temperature range of -40°C to +85°C):

$$Vo(mV) = -11.67mV/°C \times T + 1858.3$$

gives the following calculated Vo versus temperature:

Table 10: LM20BIM7 Temperature to Voltage Output relationship

Temperature (T)	Typical Voltage
+80°C	+924.7mV
+70°C	+1041.4mV
+60°C	+1158.1mV
+50°C	+1274.8mV
+40°C	+1391.5mV
+30°C	+1508.2mV
+20°C	+1624.9mV
+10°C	+1741.6mV
+0°C	+1858.2mV
-10°C	+1975.0mV
-20°C	+2091.7mV
-30°C	+2208.4mV

8.3.2 I2C Sensor (RTC Chip)

The I2C RTC chip (U16) allows the BL652 I2C interface to be tested. The output of the RTC chip (U16) is on the I2C bus and is by default connected to the BL652 module via jumpers on J13 pins 2-3 and J14 pins 2-3.

Table 11: I2C RTC chip BL652 I2C signal mappings

I2C RTC EEPROM (U16)	BL652 module (U5) SIO	Comments
(U16 pin6) RTC_SCL	(U5 pin38) SIO_27	Fit jumper on J13 pins 2-3 to select
(U16 pin5) RTC_SDA	(U5 pin37) SIO_26	Fit jumper on J14 pins 2-3 to select

Jumper on J13 pins 2-1 routes the BL652 I2C_SCL signal to Arduino connector (J32). Jumper on J14 pins 2-1 routes the BL652 I2C_SDA signal to Arduino connector (J32).

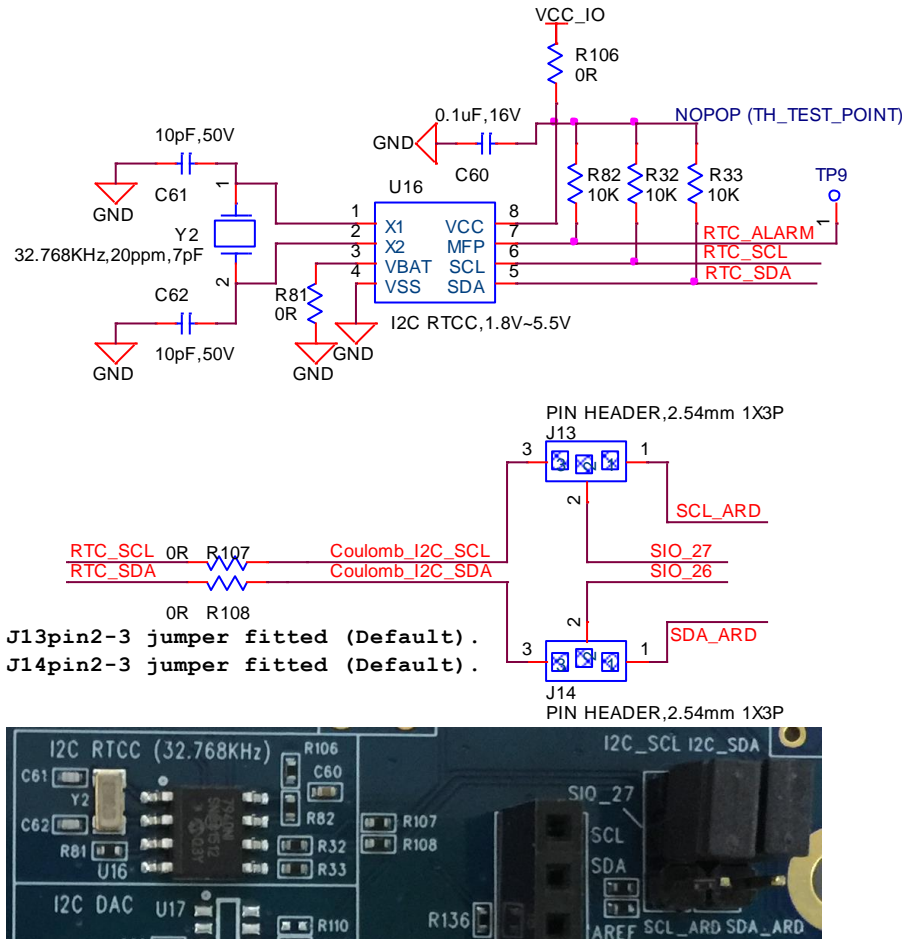


Figure 13: I2C device RTC chip schematic and PCB

To test the BL652 I2C interface, use *smartBASIC* application “*rtcs.erver.sb*” in the GitHub *smartBASIC* sample application repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications>. This application runs on the BL652 and can be used with an Android phone (requires an app such as nRF connect, https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=en_GB) or another BT900/BL620/BL652 loaded with “*rtcc.lient.sb*”.

The *smartBASIC* application “*rtcs.erver.sb*” is a BLE RTC server, and it advertises the current time (which it gets from the I2C RTC chip (U4)).

8.3.3 SPI Device EEPROM

The SPI EEPROM device (U2) is connected to the BL652 SPI pins **directly**. The 3-pin header J18 connects SIO_4 (SPI SS) to EEPROM (U2) or to the Arduino D10 (for use as SPI Slave Select). By default, the BL652 Module SIO_22 (used as the SPI_CS) is connected to EEPROM (U2) slave select line via J8 header with a jumper fitted on J18 pins 2-3. [Table 12](#) lists signal mappings how the SPI EEPROM (U2) is wired to BL652 SIO pins.

Table 12: SPI EEPROM to BL652 SPI signal mappings

SPI EEPROM (U2)	BL652 (U5) SIO	Comments
(U2pin6) Eeprom_SCK_SIO_25	SIO_25	

SPI EEPROM (U2)	BL652 (U5) SIO	Comments
(U2pin2) Eeprom_MISO_SIO_24	SIO_24	
(U2pin5) Eeprom_MOSI_SIO_23	SIO_23	
(U2pin1) Eeprom_CS_SIO_22	SIO_22 (via header J18)	Fit jumper in J18 pins 2-3 to select, then configure SIO_22 as an output and drive output low in <i>smartBASIC</i> application to select SPI slave (SPI EEPROM U2)

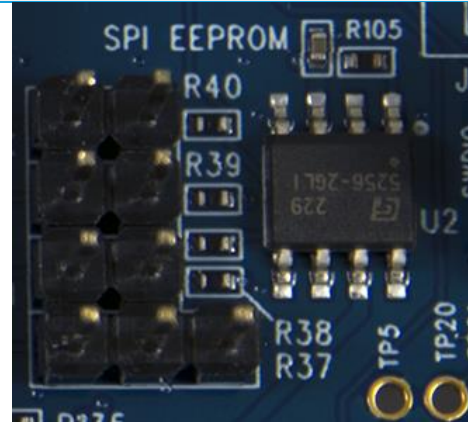
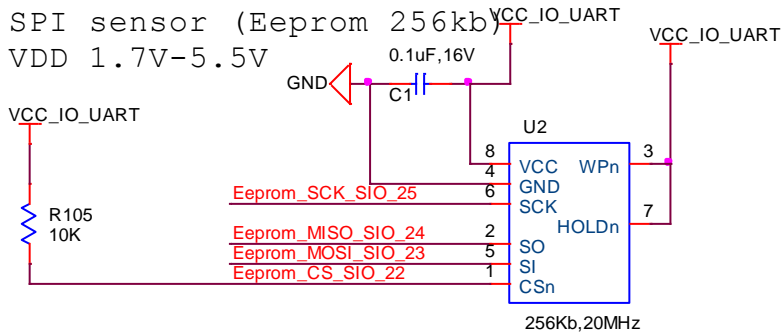


Figure 14: SPI EEPROM schematic and PCB

For a working example of the BL652 SPI interface using the SPI EEPROM (U2), a *smartBASIC* application for this will be available in the future in the GitHub *smartBASIC* sample application repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications>

8.3.4 Push Button and LED Connected to BL652

The two push buttons and two LED's on the DVK-BL652 are connected to dedicated SIO's of the BL652 module.

Table 13: LED's and Buttons to BL652 SIO signal mappings

Part	SIO	Comments
LED1 (D1)	SIO_17 (via header J26)	To connect LED1 to SIO_17, Fit jumper in J26
LED2 (D2)	SIO_19 (via header J37)	To connect LED1 to SIO_19, Fit jumper in J37
Button 1 (SW1)	SIO_11 (via series resistor R83)	
Button 2 (SW2)	SIO_15 (via series resistor R94)	

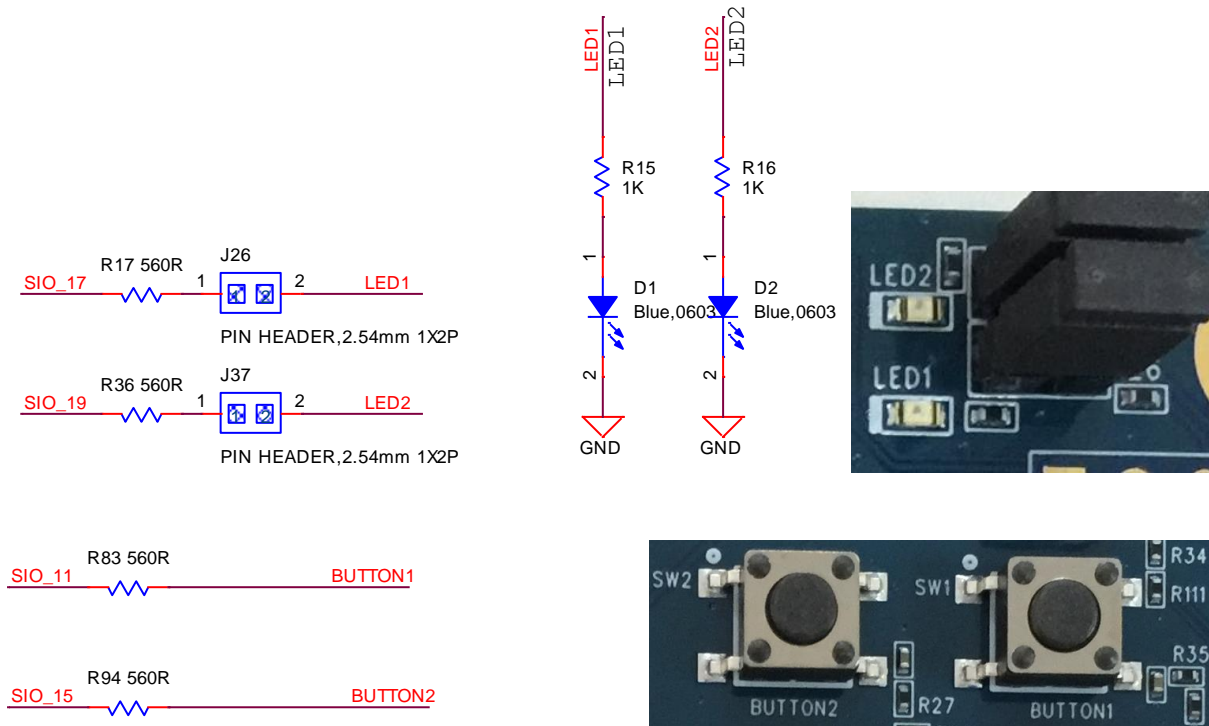


Figure 15: LEDs and Buttons schematic and PCB

The buttons (BUTTON1 and BUTTON2) have no external pull-up resistor, so to use the buttons, the SIO_11 and SIO_15 pins must be configured as inputs with internal pull-up resistors (which is the default), the following *smartBASIC* lines configure the pull-ups:

```
rc = GPIOSETFUNC(17,1,4)           '//sets SIO_11 (Button1) as a digital in,
                                   strong pull up
rc = GPIOSETFUNC(19,1,4)           '//sets SIO_15 (Button5) as a digital in,
                                   strong pull up
```

Refer to the *smartBASIC* application script example “*btn.button.led.test.sb*” in the GitHub *smartBASIC* sample application repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications>

The LEDs are active high, meaning that writing a logical one (“1”) to the output pin illuminates the LED.

One example of when push buttons can be used is when a *smartBASIC* application is written to simulate a generic data profile. Push buttons can then be pressed to increment and decrement, such as a heart rate.

8.3.5 NFC External Antenna Connector and NFC Antenna RF Matching Circuit

The NFC antenna input connector (CON2) allows the Laird supplied flex-PCB NFC antenna to be plugged in. The BL652 module NFC circuit uses two pins, pin 15 (NFC1/SIO_9) and pin 16 (NFC2/SIO_10) to connect the antenna. These pins are shared with GPIOs (SIO.09 and SIO.10). BL652 NFC pins are enabled by default. NFC can be disabled via *smartBASIC* application. Pin 15 (NFC1/SIO_9) and pin 16 (NFC2/SIO_10) are configured by default on the development board schematic to use NFC antenna, but if pin 15 (NFC1/SIO_9) and pin 16 (NFC2/SIO_10) are needed as normal GPIO’s, R98 and R99 must be removed and R100 and R101 must be shorted by 0R.

C53 (300pF) and C54 (300pF) are RF tuning of the flexi-PCB NFC antenna.

Table 14: NFC input BL652 SIO signal mappings

BL652 (U5) SIO	Bring out SIO_9 and SIO_10 to NFC antenna connector (CON2)	Bring out SIO_9 and SIO_10 to Header connector (J36)
pin 15 (NFC1/SIO_9)	Fit R98 0R (default) Remove R100 0R (default)	Remove R98 0R Fit R100 0R
pin 16 (NFC2/SIO_10)	Fit R99 0R (default) Remove R101 0R (default)	Remove R99 0R Fit R101 with 0R

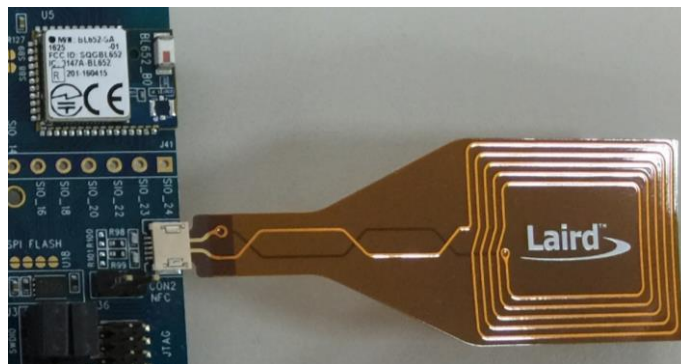
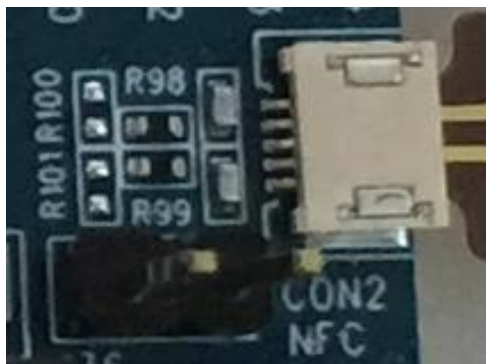
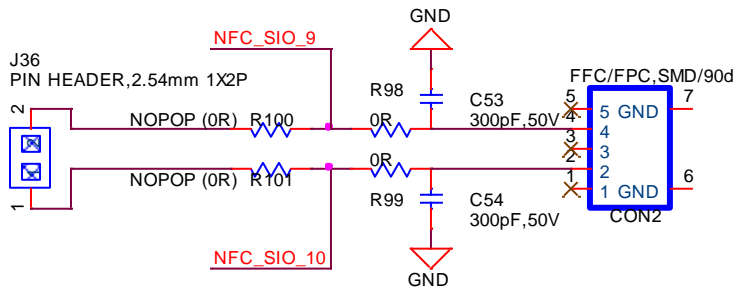


Figure 16: NFC antenna RF matching circuit, NFC antenna connector schematic and NFC plugged in to connector CON2

The *smartBASIC* application *nfc.all.launch.sb* in the GitHub *smartBASIC* sample application repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications> exercises the following over the BL652 NFC:

- On Android NFC enabled devices – Opens the Laird toolkit application or shows it in the Google Playstore if it's not installed
- On Windows NFC enabled devices – Opens the calculator
- On other NFC enabled devices – Shows the Laird website or text saying **this is a BL652**

8.3.6 Optional External Serial SPI Flash IC

There is an optional external serial SPI flash IC (U18) that may be used, for example, for data logging purposes. This optional external serial (SPI) flash (U18) must connect to BL652 module pins SIO_12 (SFLASH_CS), SIO_14 (SFLASH_MISO), SIO_16 (SFLASH_CLK), and SIO_20 (SFLASH_MOSI); in that case, a high level API in *smartBASIC* can be used for fast access using open/close/read/write API functions.

Solder bridges SB4, SB5, SB6 and SB7 must individually be shorted to connect this optional external serial (SPI) flash (U18 to the BL652 module).

By default, these are GPIO pins. Only when FlashOpen() command appears in the *smartBASIC* application script are these lines dedicated to SPI and for talking to the off-board flash.

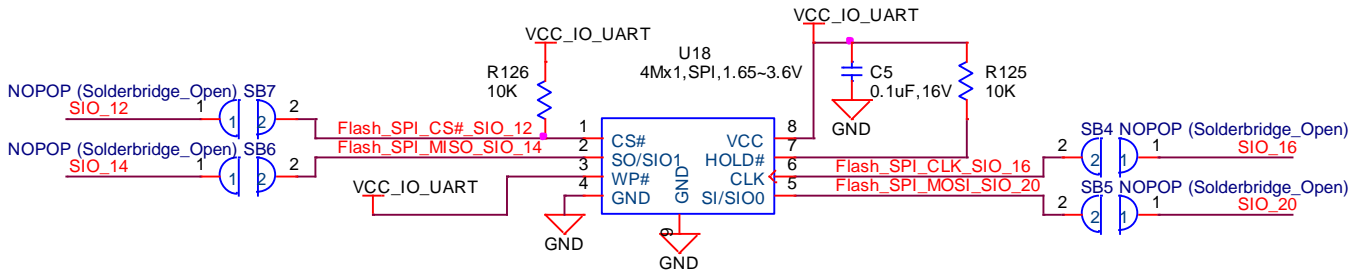


Figure 17: Optional external serial SPI flash IC (U18) schematic and PCB

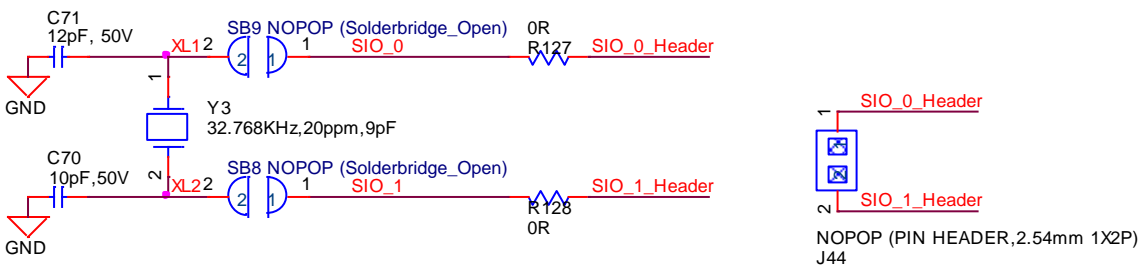
The *smartBASIC* application for this external optional serial SPI flash IC will be available in the future in the GitHub *smartBASIC* sample application repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications>

8.3.7 Optional 32.76 kHz Crystal

The BL652 on-chip 32.768kHz RC oscillator provides the standard accuracy of ± 250 ppm, with calibration required every 8 seconds (default) to stay within ± 250 ppm.

The BL652 also allows, as an option, to connect an external higher accuracy (± 20 ppm) 32.768 kHz crystal to the BL652-SX-xx pins SIO_01/XL2 (pin 24) and SIO_00/XL1 (pin 25). This provides improved protocol timing and helps with radio power consumption in the system standby doze/deep sleep modes by reducing the time that the Rx window must be open.

To connect the optional external 32.76kHz crystal oscillator circuit to the BL652 module, remove R127 and R128 and short SB8 and short SB9.



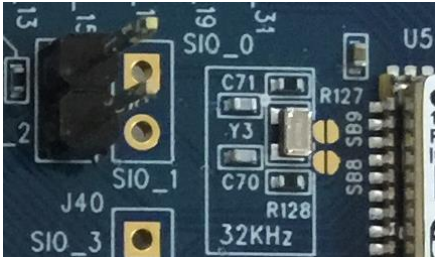


Figure 18: Optional external 32.768kHz crystal circuit schematic and PCB

A smartBASIC application will be available in the future in the GitHub smartBASIC sample application repository on the BL652 product page at <http://www.lairdtech.com/products/bl652-ble-module>

9 OTHER FEATURES

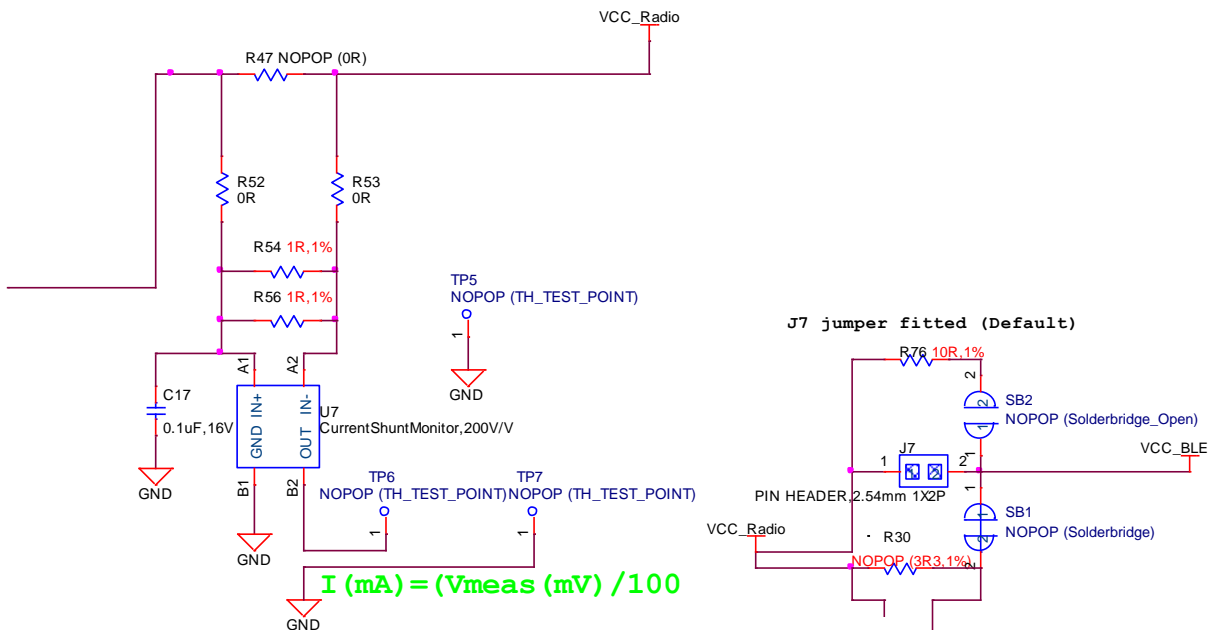
9.1 Current Consumption Measurement

A removable jumper (on J7) is provided to break the power supply line directly to the module, allowing you to measure current consumption. For normal operation, the jumper on J7 between pin1 and pin2 must be fitted (and is fitted by default).

IMPORTANT: To achieve the optimal power consumption of the BL652 series module on the development board, see the “LowPower.sb” file in the GitHub smartBASIC sample application repository on the BL652 product page at <https://github.com/LairdCP/BL652-Applications>.

Note: This measures the current consumption of the **BL652** series module ONLY.

The current drawn by the BL652 series module can be monitored on the development board. Figure 19 shows the schematic and location of measuring points on the PCB related to current measurements.



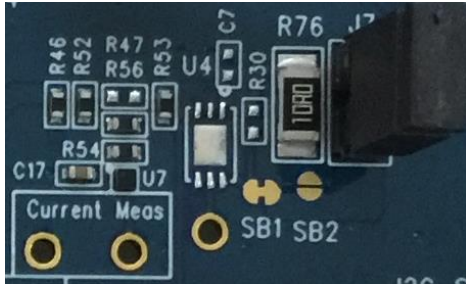


Figure 19: Current measurement schematic and PCB

There are two primary ways to measure the current consumption:

- **Using Ammeter** – Connect an ampere meter between the two pins of J7 pins 1-2. This monitors the current directly.
- **Using Oscilloscope** – The open solder bridge SB2 first needs to be shorted with solder, then the on-board 10 Ohm resistor R76 which is mounted across J7 pins 1-2 can be used as current sense resistor. Connect an oscilloscope or similar with two probes on the pins on the J7 connector and measure the voltage drop. The voltage drop is proportional with current consumption. The 10 Ohm resistor is chosen, 10 mV equals 10mA.

There is also a third way to measure current:

- **Using Current Shunt Monitor** – The current drawn by the BL652 module can be monitored using the Current Shunt Monitor (CSM), INA216 (U7). The gain of INA216 is 200 V/V for the lowest possible drop voltage.

Note: Using the current shunt monitor method allows the dynamic current consumption waveforms to be shown on an oscilloscope as the BL652 radio operates. This can provide insight into power optimization.

Current consumed by the BL652 series module is measured as a voltage (that is proportional to the current) using the current shunt monitor (U7). This is performed by connecting a voltmeter or oscilloscope to TP6 and the ground to TP7. Current in milliamps can be determined from the following equation:

$$I(\text{mA}) = V_{\text{meas_TP6}}(\text{mV}) / 100$$

CAUTION: Take care not to short TP6 (the Current Shunt Monitor IC (U7)) output to GND, as that will permanently damage the IC U7.

10 APPENDIX

10.1 Coin Cell Insertion

To insert the coin cell, follow these steps:

1. Push the coin cell against positive contact spring of holder J34.

Note: The coin cell sits below the positive contact spring (as shown with arrow).

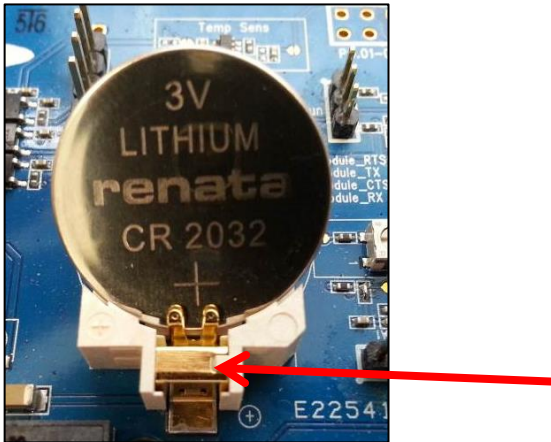


Figure 20: Inserting the coin cell (step 1)

2. Push the coin cell down into the holder (J34).



Figure 21: Inserting the coin cell (step 2)

10.2 Coin Cell Removal

To remove the coin cell, follow these steps:

1. Hold down the coin cell holder (J34) at the corners.
2. Use a screwdriver in the position shown in picture below, to gently remove the coin cell from the coin cell holder (J34), This is the correct method to remove coin-cell from holder (J34).

Note: Due to tight fit of coin cell in the coin-cell holder (J34), care should be taken prevent damage to the J34 land pads.

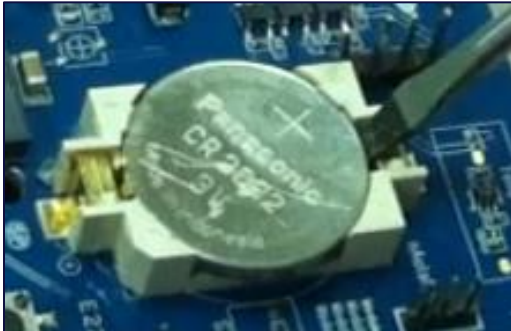


Figure 22: Removing the coin cell (step 2)

11 ADDITIONAL DOCUMENTATION

Laird offers a variety of documentation and ancillary information to support our customers through the initial evaluation process and ultimately into mass production. Additional documentation can be accessed from the Documentation tab of the [Laird BL652 Product Page](#).

For any additional questions or queries, or to receive technical support for this Development Kit or for the BL652 module series, please contact the Embedded Wireless Solutions Support Center: <http://ews-support.lairdtech.com>.

© Copyright 2016 Laird. All Rights Reserved. Patent pending. Any information furnished by Laird and its agents is believed to be accurate and reliable. All specifications are subject to change without notice. Responsibility for the use and application of Laird materials or products rests with the end user since Laird and its agents cannot be aware of all potential uses. Laird makes no warranties as to non-infringement nor as to the fitness, merchantability, or sustainability of any Laird materials or products for any specific or general uses. Laird, Laird Technologies, Inc., or any of its affiliates or agents shall not be liable for incidental or consequential damages of any kind. All Laird products are sold pursuant to the Laird Terms and Conditions of Sale in effect from time to time, a copy of which will be furnished upon request. When used as a tradename herein, *Laird* means Laird PLC or one or more subsidiaries of Laird PLC. Laird™, Laird Technologies™, corresponding logos, and other marks are trademarks or registered trademarks of Laird. Other marks may be the property of third parties. Nothing herein provides a license under any Laird or any third party intellectual property right.

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for [Bluetooth Development Tools - 802.15.1 category](#):

Click to view products by [Laird Connectivity manufacturer](#):

Other Similar products are found below :

[DA14580PRODTLKT](#) [1628](#) [MBH7BLZ02-EF-KIT](#) [CYBLE-014008-PROG](#) [FWM7BLZ20-EB-KIT](#) [ATSAMB11ZR-XPRO](#) [SKY66111-21EK1](#) [SECO-RSL10-TAG-GEVB](#) [3026](#) [MIKROE-2471](#) [MOD-NRF8001](#) [BLE-IOT-GEVB](#) [450-0184](#) [MIKROE-2399](#) [EKSHCNZXZ](#) [EVAL_PAN1026](#) [EVAL_PAN1720](#) [EVAL_PAN1740](#) [2267](#) [2479](#) [2487](#) [2633](#) [STEVAL-IDB005V1D](#) [STEVAL-IDB001V1](#) [MIKROE-2545](#) [SIPKITSLF001](#) [2995](#) [STEVAL-IDB007V1M](#) [2829](#) [DFR0267](#) [DFR0296](#) [DFR0492](#) [TEL0073](#) [BM-70-CDB](#) [WSM-BL241-ADA-008DK](#) [STEVAL-BTDP1](#) [ACD52832](#) [TEL0095](#) [ISP1507-AX-TB](#) [RN-4871-PICTAIL](#) [DA14695-00HQDEVKT-P](#) [DA14695-00HQDEVKT-U](#) [EVK-NINA-B112](#) [EBSHJNZXZ](#) [EKSHJNZXZ](#) [BMD-200-EVAL-S](#) [ACN BREAKOUT BOARD](#) [ACN SKETCH](#) [2269](#) [2746](#)