AFBR-S50 Time-of-Flight (ToF) Sensor Family
Reference Design

Application Note
Version 1.0

Table of Contents

AFBR-S50-RD: Reference Design Application Note

1 Introduction

1.1 Overview

This application note describes a reference design example for controlling a sensor from the Broadcom® AFBR-S50 Time-of-Flight (ToF) sensor family. It helps the designer realize an easy-to-implement subsystem, which consists of a ToF Sensor and an MCU, featuring a minimalistic interface footprint.

NOTE: For more information about Broadcom, see https://www.broadcom.com/. For more information about the AFBR-S50 Time-of-Flight sensor family, see https://www.broadcom.com/products/optical-sensors/time-of-flight-3d-sensors.

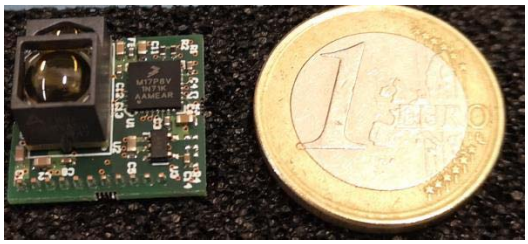
The reference design (referred to as AFBF-S50-RD in the following text) includes a 32-bit MCU and a VCSEL-based ToF sensor (Laser Class 1 eye safety), mounted on a compact-sized PCB, measuring only 16.5 mm × 18 mm in size.

The AFBR-S50-RD is composed of the main components shown in the following table.

Hardware Design – PCB	Software Design – API[a]
■ AFBR-S50 ToF sensor module (15-pin footprint) ■ Microcontroller MKL17Z256VFM4 from NXP (QFN-32) ■ AFBF-S50-RD connector (11-pin header)	■ Firmware Explorer App for MKL17Z256VFM4: For connecting with the AFBR-S50 Explorer GUI application ■ Firmware Example App for MKL17Z256VFM4: Sending measurement results over the UART interface. Printing the measurement results in the serial terminal.

a. UART interface implemented in the current firmware version (I2C, SPI available but not implemented).

Figure 1: AFBR-S50-RD Reference Design Size Compared to a 1 Euro Coin



This document provides the following information about the AFBR-S50-RD:

- All information that is required to rebuild the reference design based on the MKL17Z256VFM4 MCU.
- Instructions on how to integrate the AFBR-S50 Time-of-Flight and an Arm Cortex-M0+ MCU into a closed system that provides the full interface to external devices (Explorer App SW implementation).
  - The complete software solution is included, using the AFBR-S50 API and a reference Serial Communications Interface (SCI), provided as a source (the Explorer App [API Demo] section in the *AFBR-S50 SDK: Argus API Reference Manual*). The latter can be directly used or altered to the needs of the designer.
  - Connection to the PC using the GUI application.
  - Working in an embedded environment.

2 Hardware Documentation

This section describes all of the hardware aspects of the AFBR-S50-RD, including the schematics and PCB layout, and it explains the interconnection between the internal components. It also describes the minimum hardware requirements for the proper functioning of the AFBR-S50 sensor, including the required speed, memory amount, power supply requirements, and other parameters that may be of interest to a system integrator.

The following table shows the hardware requirements for using the S50 API and one AFBR-S50 sensor with respect to the specifications of the MKL17Z256VFM4 (Kinetis KL17Z) MCU from NXP. As the recommended MCU that meets all hardware requirements, the Kinetis KL17Z is the perfect choice for the AFBR-S50-RD.

NOTE: For more information about the Kinetis KL17Z, see https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/kl-series-cortex-m0-plus/kinetis-kl1x-48-mhz-mainstream-small-ultra-low-power-microcontrollers-mcus-based-on-arm-cortex-m0-plus-core:KL1x.

Note that the memory requirements between the Explorer App and the Example App may vary.

Requirements		Kinetis KL17Z Specification	Additional Information
CPU core	Arm core	Arm Cortex-M0+ core	Cortex-M0+ or higher Cortex-Mx processor
CPU frequency	48 MHz	48 MHz	
Peripherals	SPI Interface with GPIO access	Two 16-bit SPI modules supporting up to 24 Mb/s	API SPI interface is called S2PI
	Additional single GPIO IRQ line	28 GPIO pins	
	2 × GPIO for EEPROM communication		
	Lifetime counter	Real-time clock One 6-channel Timer/PWM module Two 2-channel Timer/PWM modules One low-power timer	Keeps track of timing in the magnitude of microseconds
	Optional – Periodic interrupt timer	Periodic interrupt timer	Triggers measurements on a time-based schedule by using interrupts
	Optional – Nonvolatile memory interface (for example, Flash)	256-KB program flash memory	Saves user calibration data upon a power or reset cycle
Memory	■ RAM: 8 KB (4 KB Heap + 4 KB Stack) ■ ROM/Flash: 128 KB	■ RAM: 32 KB ■ ROM/Flash: 256KB	

Memory used by Kinetis KL17Z in the project build phase (note that the memory requirements may vary).

Memory Explorer App Firmware	**Memory Region**	**Used Size**	**Region Size**	**Percentage Used**
	PROGRAM_FLASH:	112472 B	256 KB	42.90%
	SRAM:	26344 B	32 KB	80.40%
Memory Example App Firmware	**Memory Region**	**Used Size**	**Region Size**	**Percentage Used**
	PROGRAM_FLASH:	80780 B	256 KB	30.82%
	SRAM:	17988 B	32 KB	54.90%

The MKL17Z256VFM4 MCU has the following mechanical characteristics:

- 32 QFN
- 5 × 5 × 0.65 mm (W × L × H)
- 0.5-mm pitch
- Link to the data sheet: https://www.nxp.com/docs/en/data-sheet/KL17P64M48SF6.pdf

CAUTION! Laser Class 1 operation depends on the correct system integration and configuration of the software. Without the correct configuration or before the integration has been completed, the module can emit at higher levels and is rated as the **Laser Class 3B device**!

2.1 Schematic

Figure 2 shows the schematic of the AFBR-S50-RD.

The input voltage for the AFBR-S50-RD is 5V, and it is distributed to the circuit via the pin 1 of the J1 connector (FTS-111-02-F-S, a single-row 11-pin header with 1.27-mm/50-mil pitch).

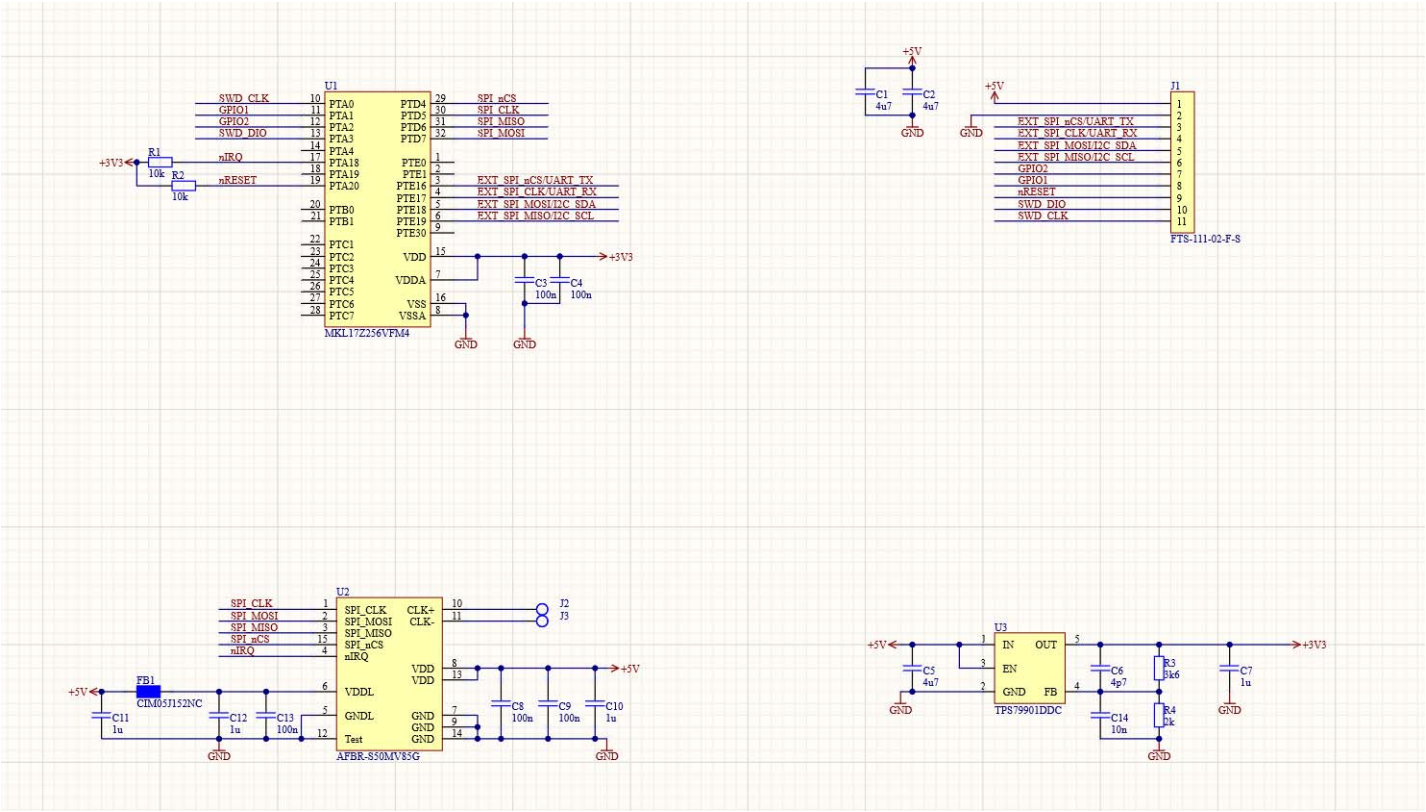
However, the operating voltage of the Kinetis KL17Z MCU is within the range from 1.7V to 3.6V. To facilitate the power requirements of the Kinetis KL17Z MCU, an additional low-dropout voltage regulator (LDO) must be used. For this purpose, the AFBR-S50-RD uses the TPS79901-EP, an ultra-low noise LDO from Texas Instruments, configured to reduce the voltage from 5V to 3.3V. It can deliver up to 200 mA with a negligible voltage drop of 100 mV, which is more than enough for the proper operation of the Kinetis KL17Z MCU. The 100-nF and 10-nF bypass capacitors on both VDD (pin 15) and VDDA (pin 7) power supply pins positioned next to the MCU itself provide the additional MCU voltage stabilization.

Two open-drain, active LOW nIRQ and nRESET signal pins (pin 17 and pin 19, respectively) are pulled up to the 3.3V supply rail using 10-kΩ resistors. Other pin definitions and functions of the Kinetis KL17Z MCU are listed in the system connection overview table in Section 2.3, System Connections Overview.

The ARGUS-S50 (AFBR-S50MV85G) ToF sensor uses 5V for its operation, so it does not require additional voltage reduction. However, due to the dynamic power consumption nature of its VCSEL driver, the sensor may introduce significant voltage ripple and noise while operating. If not properly filtered, this might cause the power supply noise to be coupled back to the application circuit and the sensor module itself. The filtering network consists of several bypass capacitors and a ferrite bead, forming a pi filter. Minimum electrical xtalk values can be achieved by shorting GND and GNDL. The filtering network is designed according to the circuit and layout recommendations from the AFBR-S50MV85G data sheet (the Application Circuit and Layout Recommendations chapter).

The AFBR-S50MV85G data sheet can be downloaded from the following location:
https://docs.broadcom.com/docs/AFBR-S50MV85G-DS

Figure 2: Schematic Reference Design



NOTE: All Gerber files of the AFBR-S50-RD will be available for download on the AFBR-S50 product pages.

2.2 PCB Layout

The PCB layout of the AFBR-S50-RD PCB follows the layout recommendations from the AFBR-S50MV85G data sheet. All the bypass capacitors are next to the ICs. Only the top layer is populated with components, optimizing both heat dissipation and manufacturing costs. The AFBR-S50-RD uses four-layer PCB topology, with the two inner layers acting as ground and power planes (GND and +5V). Most of the signal traces are routed on the bottom layer, allowing high-density PCB design. A large ground plane ensures low impedance for the signal return paths, therefore eliminating any signal crosstalk. By shielding the signal traces from the power plane, the ground plane further prevents power supply noise coupling. The compact size of the entire PCB allows the AFBR-S50-RD to be used as a complete ToF module in an out-of-the-box manner, cutting the time to market.

The following are the key specifications of the PCB layout:

- The PCB uses a four-layer configuration with dedicated ground and power planes.
- Only the top layer is populated to optimize heat dissipation and PCB costs.
- Dimensions of the PCB are 16.5 mm × 18 mm.
- AFBR-S50-RD J1 connector pitch is 1.27 mm (50 mil).
- AFBR-S50 ToF sensor pitch is 1.27 mm (50 mil).

Figure 3 and Figure 4 show the layout and the 3D model of the AFBR-S50-RD PCB design.

Figure 3: AFBR-S50-RD PCB Layout – Top View (Left), and AFBR-S50-RD PCB Layout – Bottom View (Right)

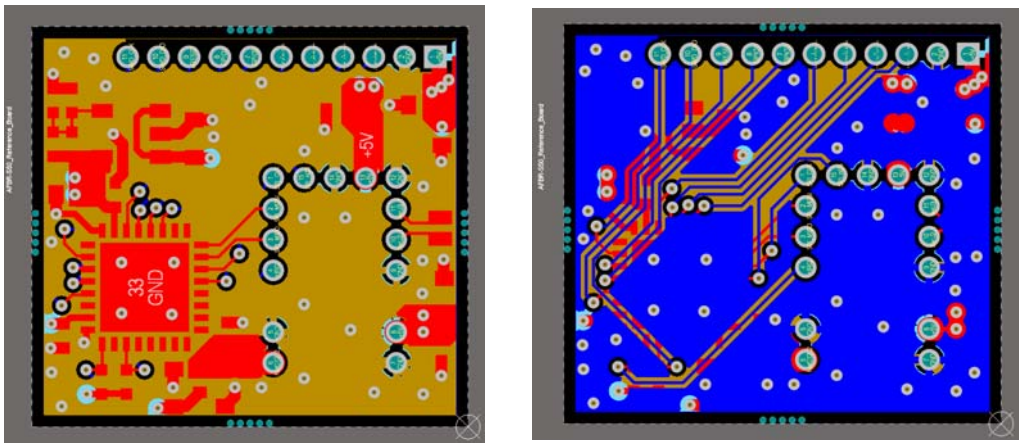
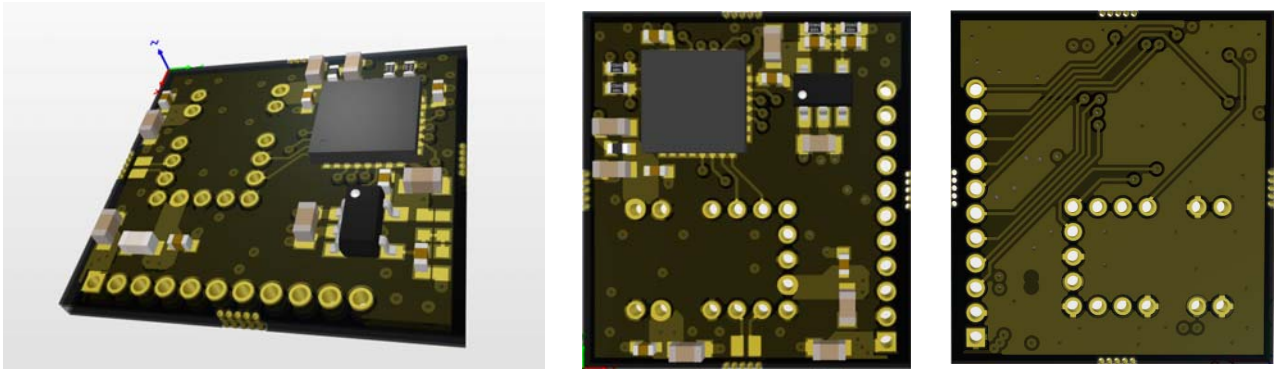


Figure 4: Three-Dimensional Models of the AFBR-S50-RD PCB



NOTE: All Gerber files of the AFBR-S50-RD will be available from the AFBR-S50 product pages.

2.3 System Connections Overview

The following table shows an overview of the signal connections from the 11-pin AFBR-S50-RD header (J1), through the Kinetis KL17Z MCU, to the AFBR-S50 ToF sensor.

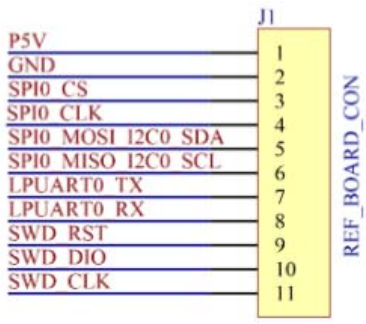
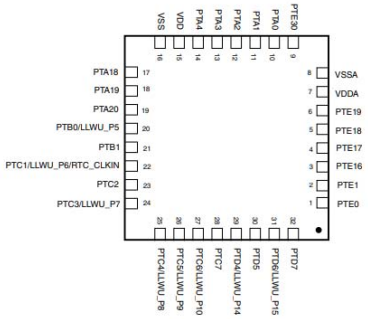
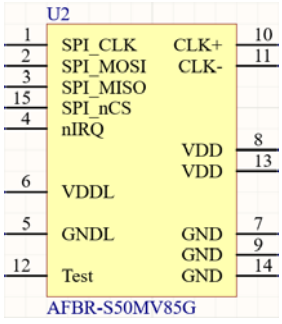
AFBR-S50-RD J1 Connector	Kinetis KL17Z Pin Function	AFBR-S50 ToF Sensor	Additional Information
Pin 1	VDD/VDDA	Pin6/8/13[a]	Input Voltage VDD 5V for ToF sensor, regulated to 3.3V for the MCU
Pin 2	VSS/VSSA	Pin5/7/9/14[a]	GND
Pin 3	PTE16	Pin 15	SPI nCS
Pin 4	PTE17	Pin 1	SPI CLK
Pin 5	PTE18	Pin 2	SPI MOSI
Pin 6	PTE19	Pin 3	SPI MISO
Pin 7	PTA2	Pin 4	rIRQ
Pin 8	PTA1		
Pin 9	PTA20		SWD interface[b]
Pin 10	PTA3		
Pin 11	PTA0		

a. All pins must be connected.

b. Debugging interface.

The following simplified scheme illustrates the communication between interfaces within the AFBR-S50-RD. Although it is possible to use the SPI, I2C, or UART interface to communicate with the MCU Kinetis KL17Z itself (interface #1), communication between the MCU and the AFBR-S50 ToF sensor is made exclusively through the SPI interface (interface #2), because the ToF sensor natively uses the SPI interface.

NOTE: More precisely, the AFBR-S50 API uses the so-called S2PI interface for the communication between the Kinetis KL17Z MCU and the ToF sensor. The S2PI module consists of a standard SPI interface plus a single GPIO interrupt line. Furthermore, the SPI pins are accessible using GPIO control to allow a software emulation of additional protocols using the same pins. For more information, refer to the *AFBR-S50 SDK: Argus API Reference Manual* More information on the S2PI interface can be found in the API reference manual, which is part of the AFBR-S50 SDK.

AFBR-S50-RD J1 Connector	→ #1	Kinetis KL17Z	→ #2	AFBR-S50 ToF Sensor
	SPI, I2C UART		SPI	

The following table lists the six multiplexed pins that allow the use of the three available communication interfaces (UART, I2C, and SPI).[1]

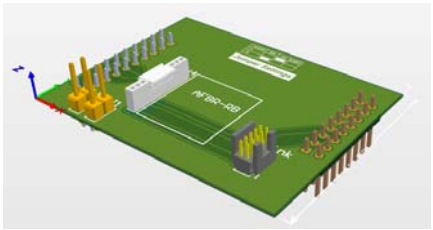
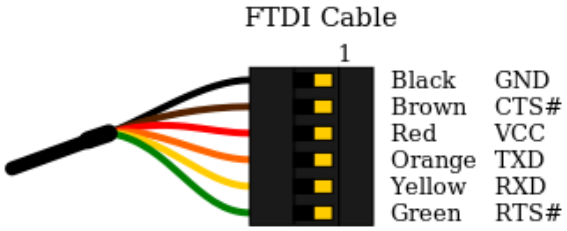
AFBR-S50-RD J1 Connector	Kinetis KL17Z Pin Function	UART Mode	I2C Mode	SPI Mode
Pin 3	PTE16	Not Used	Interface Detect	CS
Pin 4	PTE17	Not Used	Not Used	SCK
Pin 5	PTE18	Not Used	SDA	MOSI
Pin 6	PTE19	Not Used	SCL	MISO
Pin 7	PTA2	UART TX	Not Used	Interface Detect
Pin 8	PTA1	UART RX	IRQ	IRQ

2.4 AFBR-S50-RD Hardware Interfaces

This section describes two methods of accessing the AFBR-S50-RD interface. The first method represents connecting with an additional adapter board, and the second one is connecting by using an FTDI cable.

NOTE: The adapter board is primarily intended for communication with the FRDM-KL46Z evaluation kit. Additional firmware needs to be uploaded to the KL46Z MCU. The data is transferred using the USB connector. Testing was not performed for the Arduino and STM development boards.

The following table provides a brief overview of the available options.

	Option #1	Option #2
	Adapter Board	FTDI cable
		
Short description	Routing board with two connectors that are used to provide Arduino/FRDM-KL46Z compatible pinout for various development boards	Offers a serial connection between the PC and the AFBR-S50-RD reference design
Pros and Cons	Pros: Multiple interfaces, programming and debugging is possible. More robust solution. Cons: Additional firmware required.	Pros: Well suited for situations when additional features are not needed (for example, programming, debugging, I2C, and so on). Cons: Only the UART connection is available.
Debugging	Available with JTAG/SWD	Not available
Interfaces	UART and I2C	UART

1. UART interface implemented in current firmware version (I2C, SPI available but not implemented).

2.4.1 Adapter Board for the AFBR-S50-RD

The AFBR-S50-RD has a single-row 11-pin connector and, therefore, has limited connectivity capabilities. To cope with this issue and expand the connectivity, Broadcom offers an additional adapter board. The idea behind the AFBR-S50-RD adapter board is to cover as many different interfaces as possible, offering a unified connectivity solution for the AFBR-S50-RD. Hardware revisions of the adapter board are planned for the future, bringing even more connectivity features.

The AFBR-S50-RD adapter board incorporates several header connectors. Two Arduino/FRDM-KL46Z-compatible connectors offer I2C, SPI, and UART interfaces for communication with connected development boards, while the standard 10-pin JTAG/SWD connector allows for easy programming and debugging. The AFBR-S50-RD adapter board simplifies development by offering simplified connectivity to many different development boards and tools, which can be used to evaluate the AFBR-S50-RD and the ToF sensor quickly and easily.

2.4.1.1 PCB Layout and Schematics

Figure 5 to Figure 7 show the schematic and the PCB layout of the AFBR-S50-RD adapter board. The following are some of the key features of the AFBR-S50-RD adapter board:

- Dimensions of the AFBR-S50-RD adapter board PCB are 54.1 mm × 41 mm.
- The AFBR-S50-RD adapter board PCB uses a double-layer configuration.
- Altium Designer project files for the AFBR-S50-RD adapter board will be available for download.

Figure 5: Adapter Board Schematic

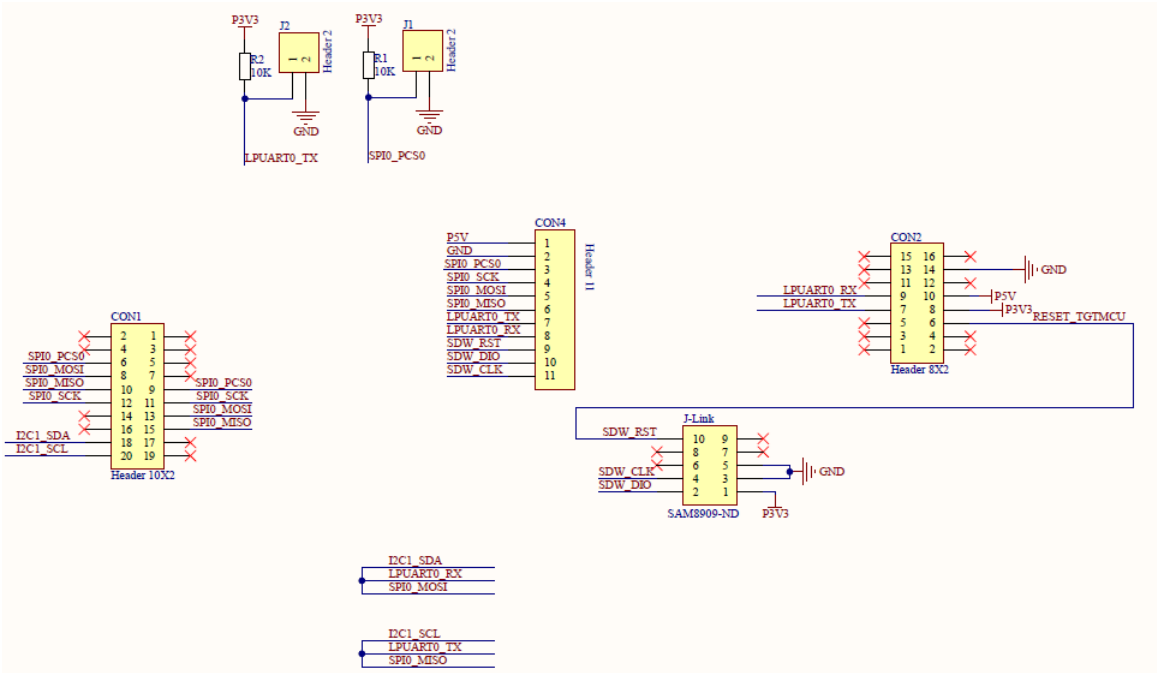


Figure 6: PCB Layout: Top View, Bottom View, and 3D Model

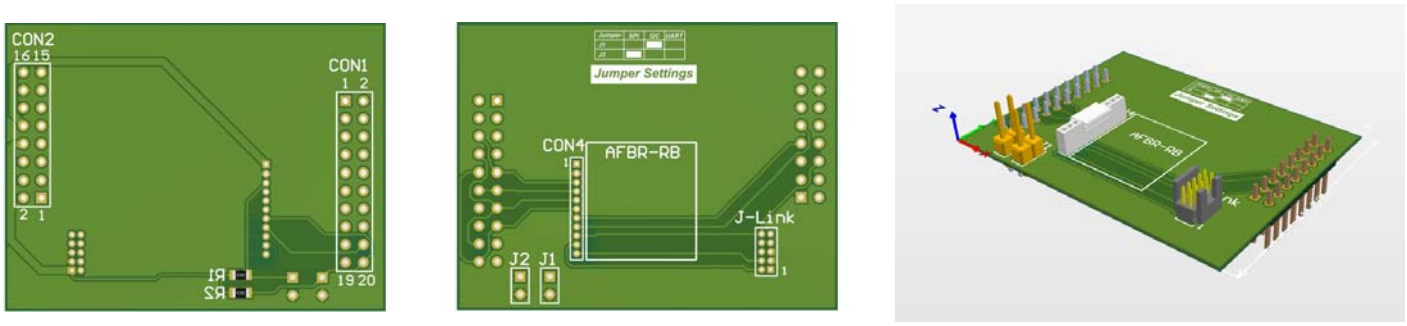
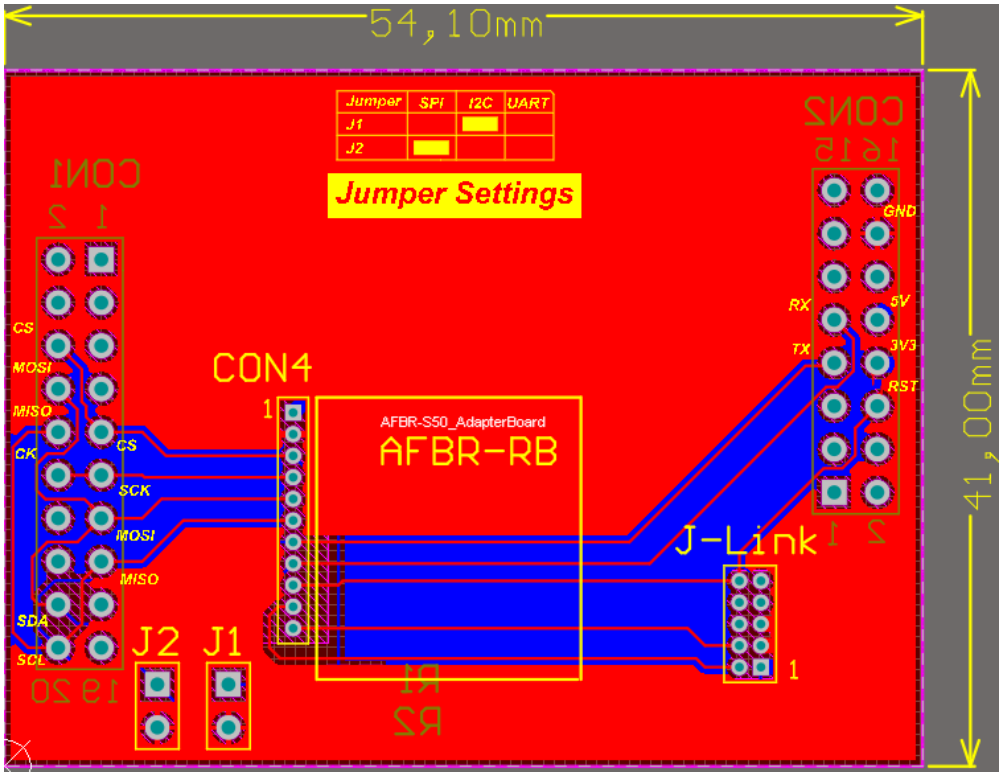


Figure 7: Adapter Board Pinout Layout



NOTE: All Gerber files of the AFBR-S50-RD will be available for download from the AFBR-S50 product pages.

Interface options are listed in the following table for UART, I2C, and SPI.

AFBR-S50-RD J1 Connector	Adapter Board CON1 Connector	Adapter Board CON2 Connector	Kinetis KL17Z Pin Function	UART Mode	I2C Mode	SPI Mode
Pin 3	Pin 6 and Pin 9	/	PTE16	Not used	Interface detection (low level)	CS
Pin 4	Pin 11 and Pin 12	/	PTE17	Not used	Not used	SCK
Pin 5	Pin 8 and Pin 13 and Pin 18	/	PTE18	Not used	SDA	MOSI
Pin 6	Pin 10 and Pin 15 and Pin 20	/	PTE19	Not used	SCL	MISO
Pin 7	Pin 10 and Pin 15 and Pin 20	Pin 7	PTA2	UART TX	Not Used	Interface detection (low level)
Pin 8	Pin 8 and Pin 13 and Pin 18	Pin 9	PTA1	UART RX	IRQ	IRQ

2.4.1.2 Available Connectors and Jumpers

The following set of header connectors is available on the adapter board for the AFBR-S50-RD:

- One 1×11-pin header (female) with 1.27-mm (50-mil) pitch, for interfacing with the AFBR-S50-RD PCB (labeled as CON4 on the schematic)
- Two 2×10-pin headers (male) with 2.54-mm (100-mil) pitch, Arduino/FRDM-KL46Z-compatible pinout (labeled as CON1 and CON2 on the schematic)
- One 2×5-pin header (male, shrouded, keyed) with 1.27-mm (50-mil) pitch, a standard 10-pin JTAG/SWD debugging header with SWD interface (labeled as J-Link on the schematic)
- Two 1×2-pin headers (male) with 2.54-mm (100-mil) pitch, used as jumpers for the communication interface selection (labeled as J1 and J2 on the schematic)

2.4.1.2.1 CON1 and CON2 Connectors

As described previously, these two connectors provide Arduino/FRDM-KL46Z-compatible pinout to various development boards, development systems, and MCU evaluation kits. The communication interface type (UART, SPI, I2C) can be selected as described in Section 2.4.1.2.2, Communication Interface Selection Jumpers. Development boards and systems compatible with the Arduino R3 pinout can use the outer row of pins:

- I2C interface (available on CON1):
  - Pin 18 (SDA)
  - Pin 20 (SCL)
  - Pin 6 (IRQ)
- SPI interface (available on CON1):
  - Pin 6 (CS)
  - Pin 8 (MOSI)
  - Pin 10 (MISO)
  - Pin 12 (SCK)
  - Pin 18 (IRQ)
- UART interface (available on CON1):
  - Pin 18 (UART RX)
  - Pin 20 (UART TX)

Development boards and systems, such as FRDM-KL46Z from NXP, can use the inner row of pins:

- SPI interface (available on CON1, except IRQ):
  - Pin 9 (CS)
  - Pin 11 (SCK)
  - Pin 13 (MOSI)
  - Pin 15 (MISO)
  - Pin 9 (IRQ, available on CON2)
- UART interface (available on CON2):
  - Pin 9 (RX)
  - Pin 7 (TX)

Power supply and Reset pins are available on the CON2 connector (outer pins):

- Pin 6 (nRESET)
- Pin 8 (3V3 INPUT)
- Pin 10 (5V INPUT)
- Pin 14 (GND)

NOTE: Both 3V3 INPUT pin and 5V INPUT must be connected. 5V is for powering AFBR-S50-RD and 3V3 for for the interface selection circuit.

2.4.1.2.2 Communication Interface Selection Jumpers

Communication interface selection jumpers (J1 and J2) are used to select the interface between the host development board and the AFBR-S50-RD. The following table shows how to select the particular interface using these jumpers. Please note that the same table exists on the bottom side of the AFBR-S50-RD adapter board.

Jumper	SPI	I2C	UART
J1	OPEN	CLOSED	OPEN
J2	CLOSED	OPEN	OPEN

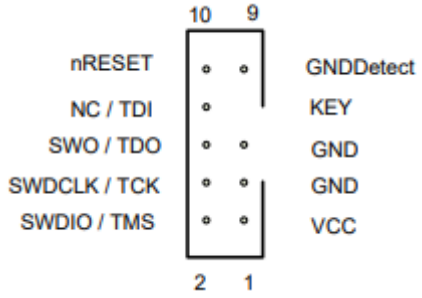
2.4.1.3 Debugging Interface and Programming Options

The AFBR-S50-RD uses the Serial Wire Debug (SWD) interface for programming and debugging. This interface consists of the VCC, GND, SWD_DIO, SWD_CLK, and nRESET signals routed to the AFBR-S50-RD J1 connector. However, it is not physically possible to connect the standard JTAG/SWD cable to the AFBR-S50-RD J1 connector directly. Therefore, an adapter board must be used.

The AFBR-S50-RD adapter board offers both the standard 10-pin JTAG/SWD program/debug connector and the 11-pin female connector, providing the necessary interconnection between the J-Link programmer and the AFBR-S50-RD itself. With the AFBR-S50-RD adapter board, the AFBR-S50-RD can be programmed and debugged directly, without requiring any hardware modifications.

- Equipment used for the programming/debugging:
  - J-Link debug probe (J-Link PRO, LPC-Link2)
  - The AFBR-S50-RD adapter board, as described previously.

Figure 8: Top View of the Standard 10-pin JTAG/SWD Connector



Besides the original J-Link PRO from SEGGER, It is also possible to use the LPC-Link2 debug probe from NXP. The LPC-Link2 debug probe is configurable using the firmware images, allowing support for various development tools and IDEs. The video in the following link explains how to flash the J-Link firmware image to the LPC-Link2 debug probe.

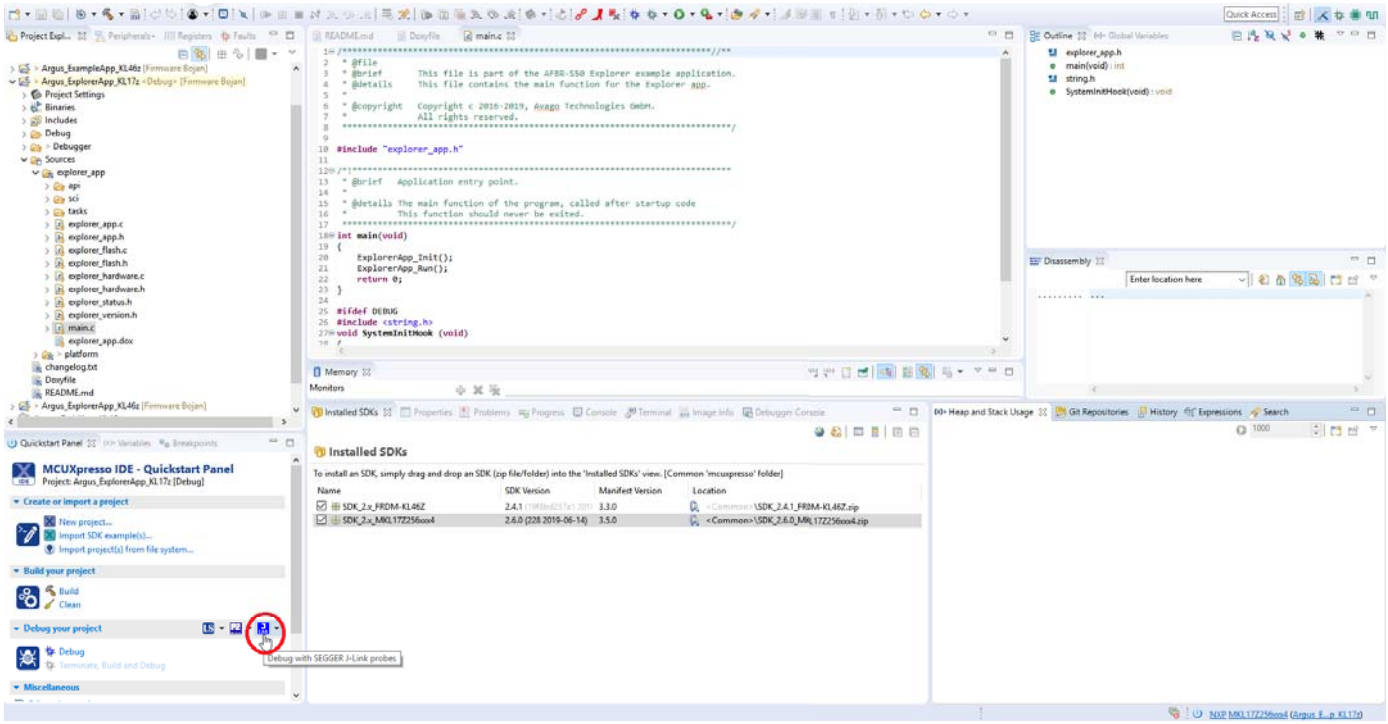
- LPC-Link2 debug probe from NXP:
  https://www.nxp.com/design/microcontrollers-developer-resources/lpc-microcontroller-utilities/lpc-link2:OM13054
- A tutorial video on YouTube, explaining how to flash the J-Link firmware:
  https://www.youtube.com/watch?v=YhHGCyLqPpI

SWD Interface pinout:

AFBR-S50-RD J1 Connector	Kinetis KL17Z Pin Function	JTAG/SWD Connector	SWD Interface Signal
Pin 9	PTA20	Pin 10	SWD_RST
Pin 10	PTA3	Pin 2	SWD_DIO
Pin 11	PTA0	Pin 4	SWD_CLK

Figure 9 shows how to debug a project by clicking on the J-Link icon. The J-Link debug probe must be installed before debugging.

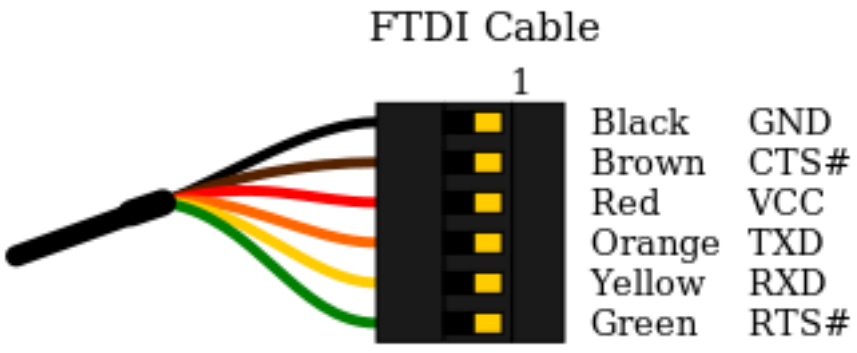
Figure 9: J-link Debug Project



2.4.2 FTDI Cable

The FTDI cable is a USB to Serial (TTL level) converter, which allows for a simple way to connect TTL interface devices, such as MCUs, to a USB. The cable is based on the FTDI FT232R IC, integrated within the cable USB type A connector. The other end of the cable is terminated with the standard 2.54-mm (100-mil) 6-pin female header connector, exposing the UART interface.

Figure 10: FTDI Cable Pinout



The FTDI cable can be connected to the UART interface of the AFBR-S50-RD adapter board using the pins on the CON2 connector, as described in the following table. Wire jumpers are used to establish the connection.

FTDI Cable UART Header	AFBR-S50-RD CON2 Header
TXD	Pin 9
RXD	Pin 7
RTS#	Pin 8
VCC	Pin 10
GND	Pin14

NOTE: Before using the FTDI cable, make sure that the FTDI drivers are installed. The FTDI drivers can be found at the FTDI Chip official site, at this address: https://www.ftdichip.com/FTDrivers.htm

3 Software Documentation

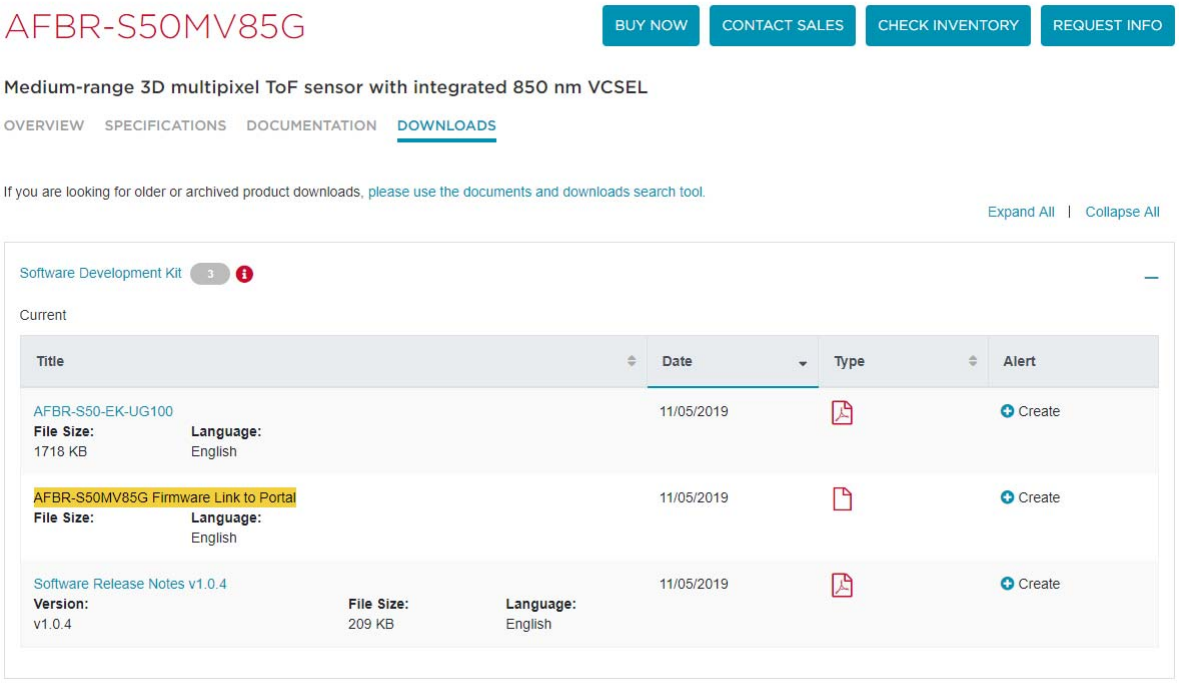
The API from the AFBR-S50 SDK contains two project files for the Kinetis KL17Z MCU: `AFBR_S50_Example_KL17Z.zip`, and `AFBR_S50_ExplorerApp_KL17Z.zip`. The Example App project provides a minimal implementation to obtain a range of values from the sensor over the UART interface. The Explorer App project contains the complete serial interface to access the full API functionality of the ToF sensor.

More information on both SW projects can be found in the API reference manual on the following location.

Before the *API Reference Manual* can be accessed, the latest AFBR-S50-SDK application must be installed from the following web address:

https://www.broadcom.com/products/optical-sensors/time-of-flight-3d-sensors/afbr-s50mv85g

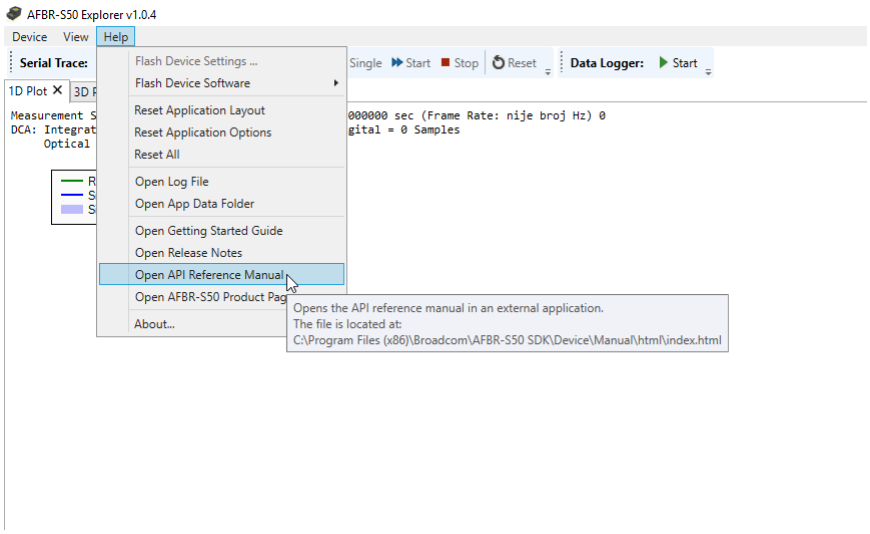
Figure 11: Location for Installation AFBR-S50-SDK Application



Direct link to the SDK: https://docs.broadcom.com/docs/12395357

After the installation, the *API Reference Manual* can be opened from the Help menu, as illustrated in Figure 12.

Figure 12: Location to *AFBR-S50 SDK: Argus API Reference Manual*



Location of Zip Projects files:

The project files (in .zip format) are located in the AFBR-S50-Explorer application installation directory (default folder: `C:\Program Files (x86)\Broadcom\AFBR-S50 SDK\Device\Projects`).

The available project files are intended for use in the MCUXpresso IDE.

An overview of the AFBR-S50 API example projects follows:

- AFBR_S50_ExampleApp_KL17Z: Basic examples for the Kineti KL17Z MCU, demonstrating the use of the AFBR-S50 SDK.
- AFBR_S50_ExplorerApp_KL17Z: The AFBR-S50 SDK for the Kinetis KL17Z MCU, embedded into the evaluation board software, containing a serial communication interface (UART). The Explorer App contains a three-layer Interface which is already used for the Evaluation Kit.

Before using either of the two available examples, it is required first to upload the appropriate compiled binary files to the AFBR-S50-RD (more precisely, to the flash memory of the Kinetis KL17Z MCU). Note that programming is only possible using the J-Link or compatible programmer (for example, LPC-Link2), as described in Section 2.4.1.3, Debugging Interface and Programming Options. This section also explains how to use FTDI cable to enable data transfer between the personal computer (PC) and the AFBR-S50-RD.

Refer to the *API Reference Manual* on how to compile and run the example projects.

3.1 Example App

As mentioned previously, the Example App only sends measurement results over the UART interface. The data output can be either printed on the PC terminal application (by using a USB-to-UART converter, such as the FTDI cable) or otherwise processed by an external development board (by using the AFBR-S50-RD adapter board).

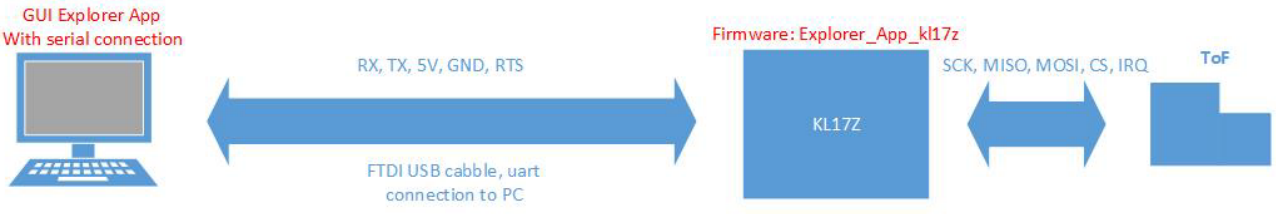
3.2 Explorer App

After the compiled binary was uploaded to the AFBR-S50-RD, the Explorer App is ready to be used.

The Explorer App allows for more functionality, using UART to send data to the PC GUI application. The application graphically displays the results of each sensor cell in the matrix, providing comprehensive visual feedback on the sensor performance.

The Explorer App requires a USB-to-UART converter to be used (for example, FTDI cable) to enable data transfer between the AFBR-S50-RD and the GUI application on the PC.

Figure 13: Diagram PC through UART to KL17Z



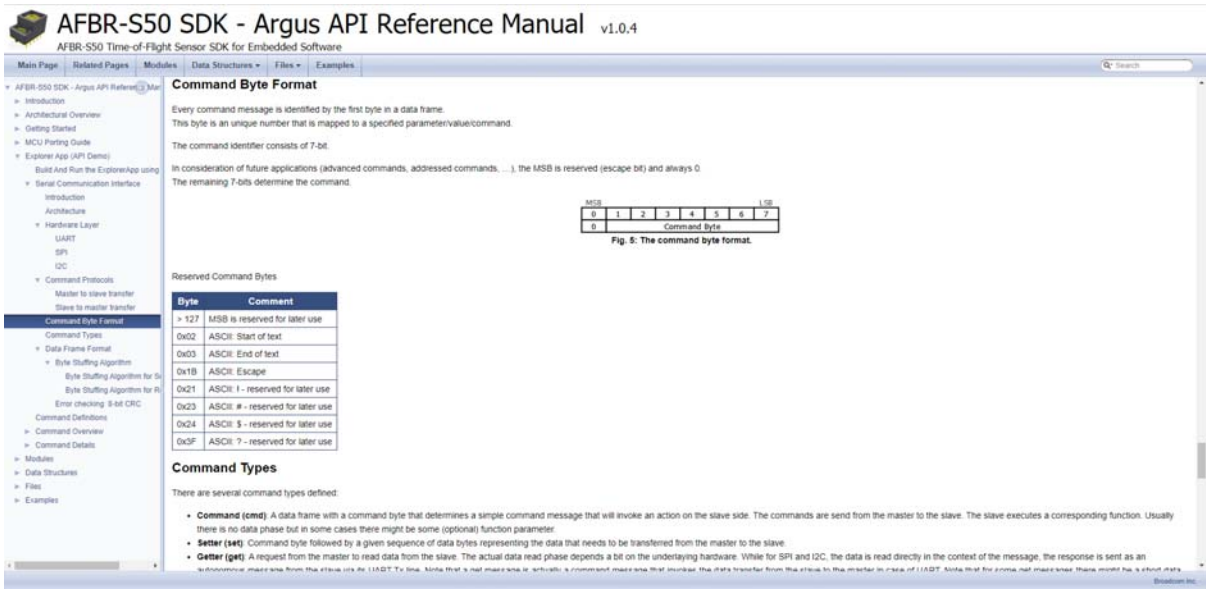
3.2.1 Terminal Commands

The advantage of using terminal commands is to have a hardware-independent library that should contain DataLink and Message layers, but not the actual hardware layers (UART, I2C, SPI) because they depend on the hardware platform. An example that implements the SCI master is provided, demonstrating connection from an external host MCU/PC to the AFBR-S50-RD using the UART interfaces.

The UART interface supports only point-to-point communication. It has independent lines for sending and receiving data. The slave device can transmit data at any time, without requiring any specific actions from the master. Therefore, in this mode, the interface does not require an interrupt line to inform the master about new data readiness or error conditions; the data is transmitted immediately, which means that the master must always listen to its RX line.

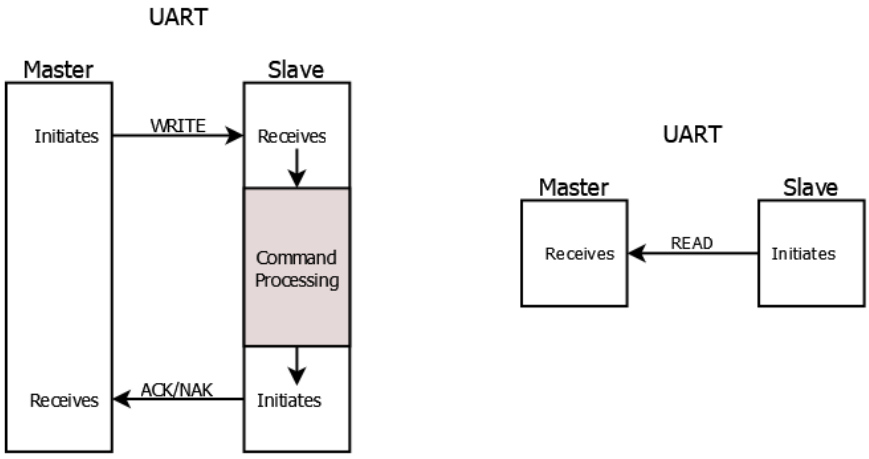
The data framing is implemented with byte stuffing. There are three special bytes: start, stop, and escape bytes. They are used to determine the boundaries of a data frame. The corresponding data bytes are inverted and terminated with the escape byte, to make start and stop bytes unique, and keep the full data range per byte. In python script, these are labeled as start_byte, stop_byte, and esc_byte. The values assigned to those variables can be found in the Command Byte Format section in the *AFBR-S50 SDK: Argus API Reference Manual*, as shown in Figure 14.

Figure 14: API Reference Manual, Command Byte Format



Each command to a slave is acknowledged after successful execution. If any error occurs, a not-acknowledge is invoked by the slave. Only a single command can be sent to the slave at once, that is, the slave has to (not-)acknowledge the command before the master can send another one. A timeout can be implemented in the master to check if the slave responds within a given time and is still alive or if it is stuck in some invalid state. Figure 15 shows these concepts.

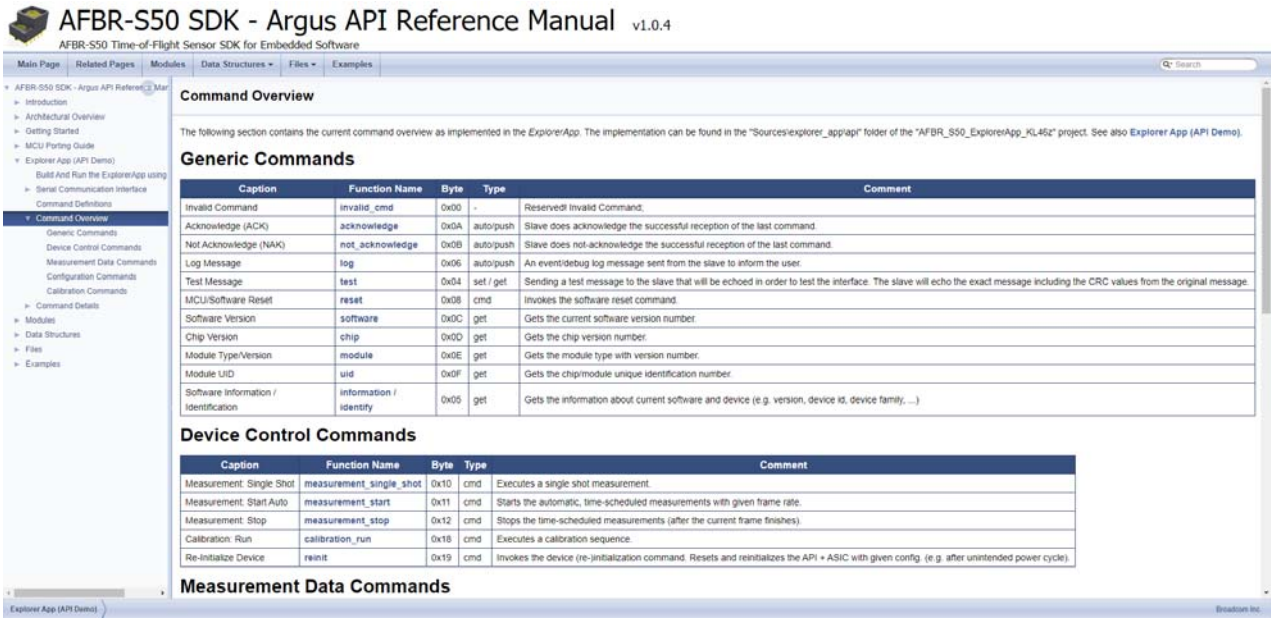
Figure 15: The Left Side Shows Master to Slave Communication. The Right Side Shows Slave to Master Communication.



Besides of the independent TX line, the UART protocol supports the special feature of log and error messages.

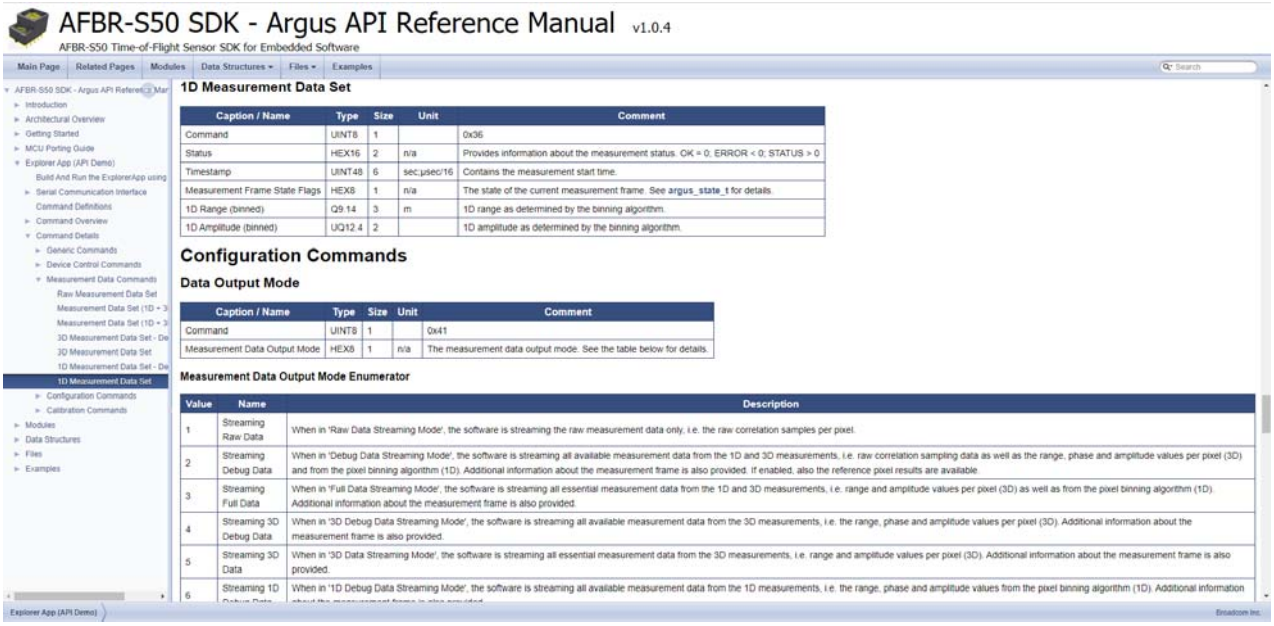
For interpreting specific commands, refer to the Command Overview section in the *AFBR-S50 SDK: Argus API Reference Manual*, as shown in Figure 16.

Figure 16: API Reference Manual, Command Overview



In Command Overview under Measurement Data Commands, you can follow the measurement_data_1d function, and find out what the format of the received data will be. The function is located in **Command Details > Measurement Data Commands > 1D Measurement Data Set**, as shown in Figure 17.

Figure 17: API Reference Manual, 1D Measurement Data Set



3.2.2 Python Commands

Example of Python (3.6) demo on how to use the SCI with UART interfaces:

Procedure for FRDM-KL46Z evaluation kit:

1. Prepare your evaluation kit (with NXP MKL46z MCU) by flashing the UART binary to the device.
2. Connect the OpenSDA USB port (not the one labeled with KL46Z) to your computer.
3. Go to the Device/Binary folder install directory of your SDK (default: `C:\Program Files (x86)\Broadcom\AFBR-S50 SDK\Device\Binary`) and copy the `AFBR.S50.ExplorerApp.vX.X.X_KL46z_UART.srec` file to the OpenSDA USB drive.

Procedure for the adapter board with AFBR-S50-RD and FRDM-KL46Z evaluation kit:

1. Prepare your evaluation kit (with NXP MKL46z MCU) by flashing the UART code to the device.
2. Connect the OpenSDA USB port (not the one labeled with KL46Z) to your computer.
3. Go to the Device/Projects folder install directory of your SDK (default: `C:\Program Files (x86)\Broadcom\AFBR-S50 SDK\Device\Projects`) and open project `AFBR.S50.BridgeApp.vX.X.X_KL46z_UART`, which has UART as default option enabled (preprocessor key: USE_USB=0).
4. Upload code as described in Section 2.4.1.3, Debugging Interface and Programming Options.
5. Program AFBR-S50-RD as described in Section 3, Software Documentation, with code AFBR_S50_ExplorerApp.vX.X.X_KL17Z.

Now that you have finished one of the preceding procedures, after flashing, the device is ready to receive commands using the OpenSDA serial port. Go to your device manager to find out which COM port is assigned to the device. Type it to the port variable below, before starting the script.

The script requires the pySerial module, which may need to be installed additionally.

See: https://pyserial.readthedocs.io/en/latest/index.html

The script sends configuration commands to set the data output mode to 1D data only, and the frame rate to 5 Hz. After setting up the configuration parameters, the measurement is started. The range is extracted from the received data frames, and it is printed to the console.

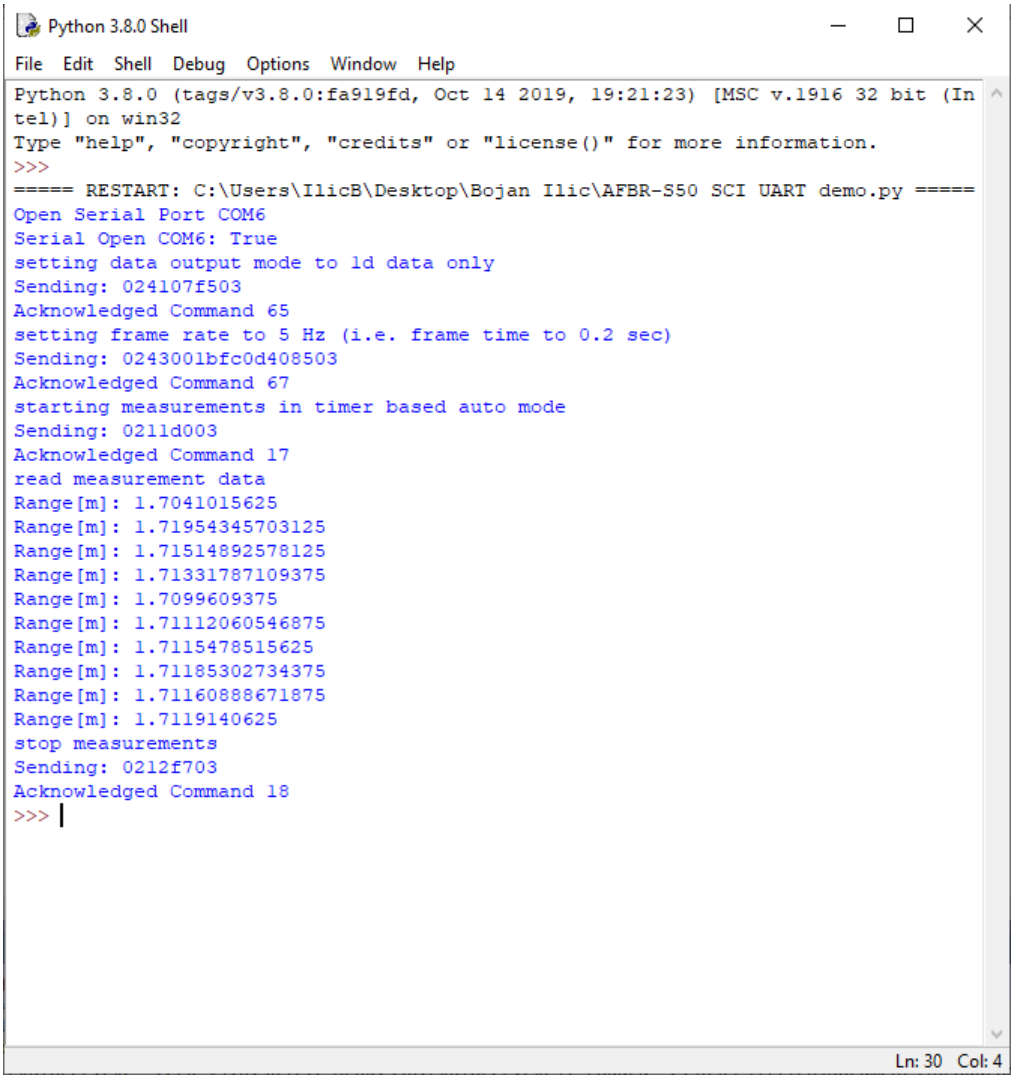
NOTE: The CRC values are calculated manually and added before the frames are sent. You can use the online calculator from the following link with CRC8_SAE_J1850_ZERO to obtain the CRC values for a frame: http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

The results of executing the script can be seen in Figure 18, the variable sample_count is changed to 10.

```
import serial

# input parameters
port = "COM6"
sample_count = 100


# byte stuffing definitions
start_byte = b'\x02'
stop_byte = b'\x03'
esc_byte = b'\x1B'

def write(tx: bytes):

    print("Sending: " + tx.hex())
    ser.write(tx)

    return

def read():

    # read until next stop byte
    rx = bytearray(ser.read_until(stop_byte))

    # remove escape bytes if any
    rxi = rx.split(esc_byte)
    rx = b''
    for i in range(len(rxi)):
       rxi[i][0] ^= 0xFF # invert byte after escape byte (also inverts start byte, but we don't care..)
    rx = rx.join(rxi)

    # extract command byte (first after start byte)
    cmd = rx[1]

    # interpret commands
    if cmd == 0x0A: # Acknowledge
        print ("Acknowledged Command " + str(rx[2]))

    elif cmd == 0x0B: # Not-Acknowledge
        print ("Not-Acknowledged Command " + str(rx[2]) + " - Error: " + str((rx[3] << 8) + rx[4]))

    elif cmd == 0x06: # Log Message
        print("Device Log: " + str(rx[8:-2]))
```

```
    elif cmd == 0x36: # 1D Data Set
        # Extract Range:
        r = (rx[12] << 16) + (rx[13] << 8) + rx[14]
        r = r / 16384.0 # convert from Q9.14
        print ("Range[m]: " + str(r))

    else: # Unknown or not handled here
        print ("Received Unknown: " + rx.hex())

    return rx

# open serial port w/ "11500,8,N,1", no timeout
print("Open Serial Port " + port)
with serial.Serial(port, 115200) as ser:
    print("Serial Open " + port + ": " + str(ser.is_open))

    # discard old data
    ser.timeout = 0.1
    while len(ser.read(100)) > 0: pass
    ser.timeout = None

    # setting data output mode to 1D data only
    print("setting data output mode to 1d data only")
    write(bytes.fromhex('02 41 07 F5 03'))
    read()

    # setting frame time to 200000 µsec = 0x00030D40 µsec
    # NOTE: the 0x03 must be escaped and inverted (i.e. use 0x1BFC instead of 0x03)
    print("setting frame rate to 5 Hz (i.e. frame time to 0.2 sec)")
    write(bytes.fromhex('02 43 00 1B FC 0D 40 85 03'))
    read()

    # starting measurements
    print("starting measurements in timer based auto mode")
    write(bytes.fromhex('02 11 D0 03'))
    read()

    #read measurement data
    print("read measurement data")
    for i in range(sample_count):
        read()

    # starting measurements
    print("stop measurements")
    write(bytes.fromhex('02 12 F7 03'))
    read()

    ser.close()             # close port
```

Figure 18: Output from the Python Script



4 Troubleshooting and FAQs

4.1 Related Documents

For more information, refer to the *AFBR-S50 SDK: Argus API Reference Manual*.

Revision History

AFBR-S50-RD-AN100; April 23, 2020

- Initial release of the document.


BROADCOM®