AFBR-S50-MULTISEN

Integration of Multiple AFBR-S50 Sensors

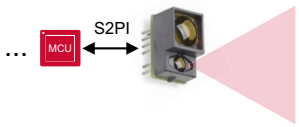
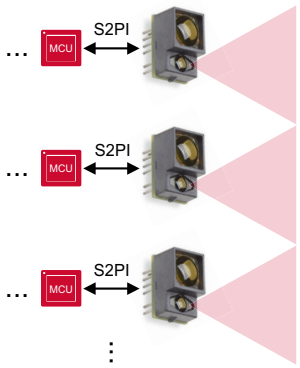
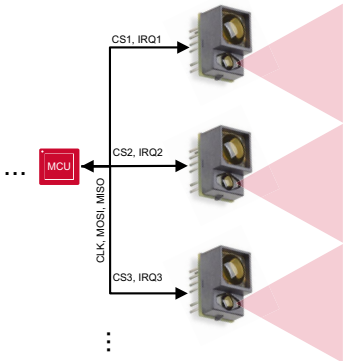
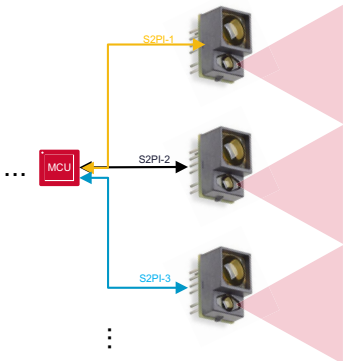
Application Note

Chapter 1: Overview

The purpose of this application note is to facilitate the implementation of multiple AFBR-S50 sensors in a system. Beside a standard single-sensor configuration, Figure 1 shows the several multi-sensor connection options:

- Option 1 – Dedicated MCU and dedicated SPI bus
- Option 2 – Shared MCU and shared SPI bus
- Option 3 – Shared MCU and dedicated SPI bus

Figure 1: Single vs Multiple Sensor Options

Single-Sensor Configuration	**Multi-Sensor Configurations**		
	Dedicated MCU, Dedicated SPI	Shared MCU, Shared SPI	Shared MCU, Dedicated SPI
	Option 1	*Option 2*	*Option 3*
			

S2PI := CS, CLK, MOSI, MISO, & IRQ (see https://broadcom.github.io/AFBR-S50-API/porting_guide.html#pg_s2pi)

Although options 1 to 3 aim to increase the field of view (FoV), all have their pros and cons, which are indicated in Table 1.

Table 1: Comparison of Options

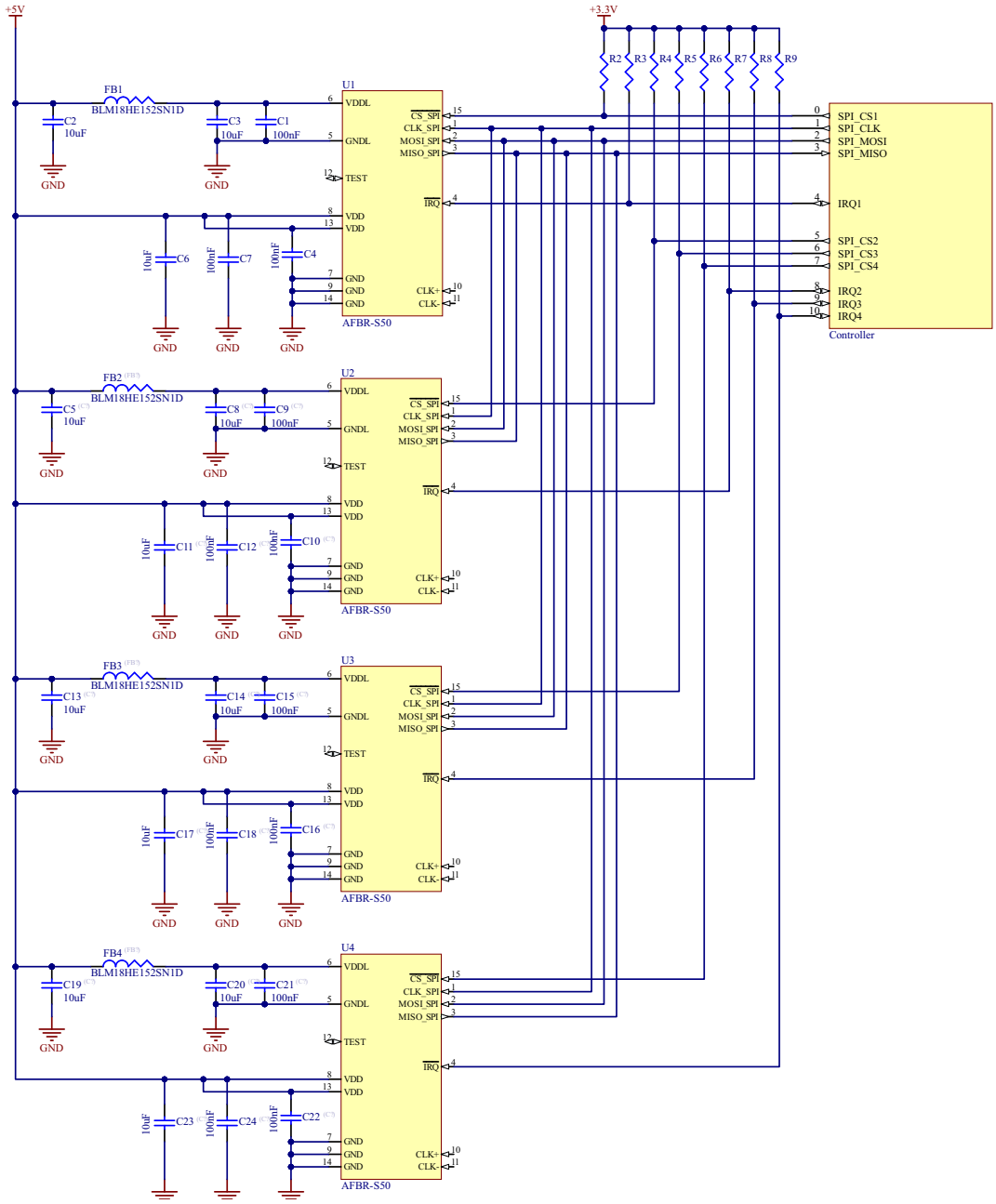
Scenario	Extended FoV	MCU Occupancy	PCB Area Occupancy	Modularity	BOM Costs
Option 1	✓	✓	☒	✓	☒
Option 2	✓	☒	✓	☒	✓
Option 3	✓	☒	☒	☒	✓

While all options yield an extended FoV, option 1 has a clear advantage in terms of MCU occupancy due to a dedicated MCU per sensor. On the other hand, the least PCB occupancy is given with option 2, which is using only a single SPI bus and a single MCU for all system communication. The modularity strongly depends on the implementation but can be more convenient with separated couples of MCU+sensor, which gives option 1 an advantage. Eventually, the BOM costs benefit from the shared MCU possibility of options 2 and 3.

Chapter 2: Hardware

The PCB design of multiple sensors and a single MCU requires a dedicated chip select (CS) and an IRQ (interrupt) line for each sensor.

Figure 2: Schematic – Multi-sensor Wiring to a Controller Unit (Option 2)



NOTE: The pull-up resistors R2 to R9 in Figure 2 only indicate the native low-active implementation of CS and IRQ and are usually configured from within the MCU.

Chapter 3: Software

The AFBR-S50 GitHub repository or software development kit (SDK) offers an example of a multi-sensor implementation in C. The following C-code configurations must be carried out before you can run the example. Keep in mind that up to four sensors are natively supported as of revision v1.4.4. If you need to connect more sensors, see Which code changes are required when using more than four sensors on a single MCU?

NOTE: Make sure to always use the latest example code from our GitHub repository under https://github.com/Broadcom/AFBR-S50-API.

Steps:

1. Open the `Example.h` header file, and change the `API_EXAMPLE` definition to 4.

```
/*! Selector for example:
 *   - 1: 01_simple_example.c: Runs measurements in simplest blocking manner.
 *   - 2: 02_advanced_example.c: Starts measurements automatically from timer interrupt.
 *   - 3: 03_high_speed_example.c: Runs measurements with up to 3000 frames per second.
 *   - 4: 04_multi_device_example.c: Demonstrates the usage of multiple devices per MCU. */
#ifndef API_EXAMPLE
#define API_EXAMPLE 4
#endif
```

2. Change number of devices according to your target count in the `04_multi_device_example.c` source file.

```
#include "examples.h"
#if API_EXAMPLE == 4

#include "argus.h"
#include "board/board_config.h"    // declaration of S2PI slaves
#include "platform/argus_print.h"  // declaration of print()
#include "driver/irq.h" // declaration of IRQ_LOCK/UNLOCK()

#if RUN_HAL_TESTS
#include "argus_hal_test.h"
#endif

/*! The number of devices connected/used in the example. Range: 1 to 4. */
#ifndef DEVICE_COUNT
#define DEVICE_COUNT 4
#endif
```

3. Open a terminal application to view the streamed data.

```
#1:    22.173765 s; Range:   420 mm;  Amplitude:  687 LSB;  Quality: 100;  Status: 0
#2:    22.220142 s; Range:   458 mm;  Amplitude:  518 LSB;  Quality: 100;  Status: 0
#3:    22.265742 s; Range:   508 mm;  Amplitude:  519 LSB;  Quality: 100;  Status: 0
#4:    22.311035 s; Range:   575 mm;  Amplitude:  566 LSB;  Quality: 100;  Status: 0
#1:    22.357510 s; Range:   425 mm;  Amplitude:  601 LSB;  Quality: 100;  Status: 0
#2:    22.393178 s; Range:   465 mm;  Amplitude:  552 LSB;  Quality: 100;  Status: 0
#3:    22.429316 s; Range:   516 mm;  Amplitude:  539 LSB;  Quality: 100;  Status: 0
#4:    22.465157 s; Range:   581 mm;  Amplitude:  519 LSB;  Quality: 100;  Status: 0
```

The example provides distance plus monitoring data for each sensor indicated with the sensor number on the left side.

NOTE: See https://broadcom.github.io/AFBR-S50-API/stm32cubeide.html#autotoc_md24 for more information on how to establish a connection via UART and a PC.

Chapter 4: FAQs

This section provides an overview of the most frequently asked questions about implementing multiple sensors in a system.

How much MCU memory is required when controlling multiple sensors?

Single Sensor:

- RAM: 4-KB Heap + 4-KB Stack
- ROM/Flash: 128 KB

n Sensors on a Single MCU:

- RAM: n × 4-KB Heap + 4-KB Stack
- ROM/Flash: 128 KB

What is the frame-rate trade-off when using more sensors on a single MCU and a single SPI bus?

The reference to achieve the maximum frame rate of 3 kHz is tested with MCU clock = 100 MHz and SPI clock = 21 MHz with a single sensor. The maximum frame rate downscales in accordance with the number of connected sensors.

Number of Sensors	1	2	3	4	...
Maximum Frame Rate	3 kHz	1.5 kHz	1 kHz	750 Hz	...

Why do I receive memory faults when using more sensors?

Check for the correct heap memory size in the linker script (for instance, `STM32F401RETX_FLASH.ld` in the case of an STM MCU). Make sure to allocate a minimum heap size of 4 KB per sensor.

```
/* Highest address of the user mode stack */
_estack = ORIGIN(RAM) + LENGTH(RAM);    /* end of "RAM" Ram type memory */

_Min_Heap_Size = 0x4000;    /* required amount of heap  */
_Min_Stack_Size = 0x4000;   /* required amount of stack */
```

NOTE: The provided project via GitHub already uses 16 KB (0x4000), which is sufficient for four sensors. If you merge the AFBR-S50 API into an existing project, you might need to double-check the memory allocation.

Which code changes are required when using more than four sensors on a single MCU?

Several API implementations must be added or changed:

1. Amend GPIO definitions including assignments to additional IRQ and CS signals.
   See the `board_config.h` header file.
2. Extend all functions with SPI slave usage accordingly.
   See the `s2pi.c source` file.

NOTE: As of AFBR-S50 API release v4.4, the precompiled library can support up to 16 sensors connected to a single MCU. If your implementation requires a higher number, contact support.tof@broadcom.com.


BROADCOM®