The FSP200 is a 6-axis IMU processor, integrating CEVA's Hillcrest Labs business unit's high-performance sensor hub software stack, to provide heading and orientation outputs. When connected to one of several supported sensors, it performs all the accelerometer and gyroscope sensor fusion processing necessary to produce stable and accurate heading and orientation outputs. This document is intended to provide information about the FSP200 Development Kit, the pin connections and software provided by Hillcrest to facilitate customer integration.

1. Hardware

1.1. Introduction

The FSP200 Development Kit includes a shield known as the FSP200 ("target device") Development Board that is designed for quick and easy development and prototyping. The shield is designed to connect to the STM32F411RE Nucleo ("bridge board") which is developed and sold by STMicroelectronics[1] (ST) and is included in the FSP200 Development Kit package. The Nucleo platform includes a STM32F411 MCU that serves as the master to the FSP200. This document includes details for connecting and using the FSP200 Development Kit with the ST Nucleo prototyping platform.

[1] https://www.st.com/en/evaluation-tools/nucleo-f411re.html

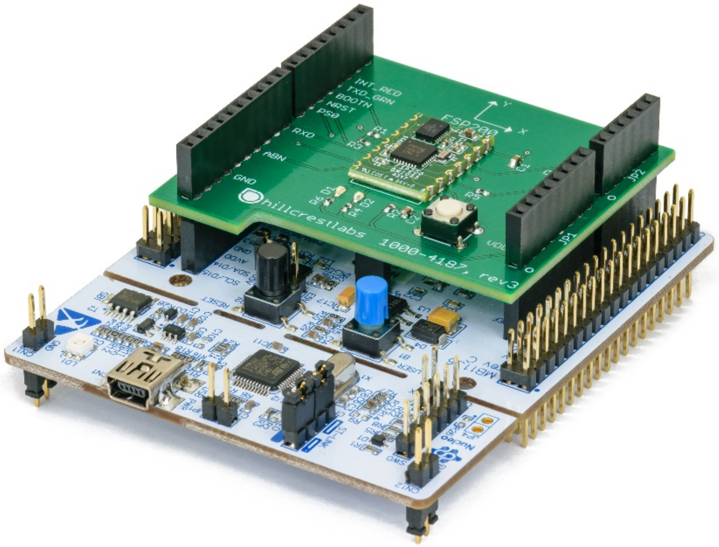


Figure 1: FSP200 Development Kit with ST Nucleo

1.2. Connections

The FSP200 Development Board simply plugs into the Nucleo board and is ready to use. The reader is encouraged to review the FSP200 Datasheet [1] for more information on the pinout configurations. The FSP200 communicates with a host system over a UART interface, either in UART-RVC mode, UART-RVC-LOG mode or in UART-SHTP mode.

Dev Board	Signal	Nucleo
JP1.1	NC	CN6.1
JP1.2	VDD	CN6.2
JP1.3	NC	CN6.3
JP1.4	NC	CN6.4
JP1.5	NC	CN6.5
JP1.6	GND	CN6.6
JP1.7	GND	CN6.7
JP1.8	NC	CN6.8

Dev Board	Signal	Nucleo
JP2.1	CAL_PB	CN8.1
JP2.2	NC	CN8.2
JP2.3	NC	CN8.3
JP2.4	PS1	CN8.4
JP2.5	NC	CN8.5
JP2.6	NC	CN8.6

Dev Board	Signal	Nucleo
JP3.10	NC	CN5.10
JP3.9	NC	CN5.9
JP3.8	NC	CN5.8
JP3.7	GND	CN5.7
JP3.6	NC	CN5.6
JP3.5	NC	CN5.5
JP3.4	H_MOSI_ABN	CN5.4
JP3.3	NC	CN5.3
JP3.2	NC	CN5.2
JP3.1	H_RX	CN5.1

Dev Board	Signal	Nucleo
JP4.8	NC	CN9.8
JP4.7	H_WAKEN_PS0	CN9.7
JP4.6	RESETN	CN9.6
JP4.5	BOOTN	CN9.5
JP4.4	H_TX_LEDGRN	CN9.4
JP4.3	H_INT_LEDRED	CN9.3
JP4.2	NC	CN9.2
JP4.1	NC	CN9.1

Figure 2: FSP200 Development Board Interface to Nucleo Host device

In UART-SHTP mode, the FSP200 uses the Sensor Hub Transport Protocol (SHTP) to communicate with a system or application processor (host that connects to the FSP200). The SHTP protocol is documented in the Sensor Hub Transport Protocol [4], allowing a customer to potentially develop their own host software if they choose to do so. In order to ease customer integration, Hillcrest has developed software that runs on a host platform such as the STM32F411RE Nucleo series. The software driver fully implements the communication protocol used by the FSP200. Hillcrest provides this software driver package as source code. The FSP200 Development Kit has programmed the ST Nucleo to work with the Hillcrest PC application to demonstrate functions. Customers who intend to use the FSP200 Development Kit for their own software development should use the driver package to download new firmware.

In UART-RVC mode, the FSP200 transmits heading and sensor information at 100Hz over the H_TX_LEDGRN pin. UART-RVC mode has its own dedicated packet format. It’s not configurable, but convenient to get a limited sensor data set for applications.

UART-RVC-LOG mode is similar to the UART-RVC mode but with the addition of raw accelerometer and gyroscope outputs for debug purposes.

1.3. Host Interface Configurations

The communication interface is configured by setting the protocol selection PS0 and PS1 pins appropriately.

PS1	PS0	FSP200 Transport Protocol
0	0	UART-RVC-LOG
0	1	UART-RVC
1	0	UART-SHTP
1	1	Reserved

Figure 3: FSP200 Available Selection of Host Communication Interface

1.4. Calibration Button

There is a calibration button on the board to perform simple calibration. See section 3 for details on performing calibration.

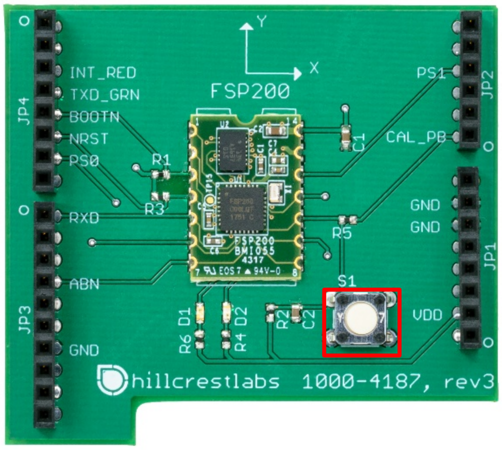


Figure 4: Simple Calibration Button

2. Software

2.1. FSP200 Development Kit

FSP200 Development Kit contains a pre-programmed STM32 Nucleo board with Hillcrest software that allows communication between the FSP200 and Freespace™ MotionStudio 2.

Freespace™ MotionStudio 2 is a Windows application to allow users to control and configure the FSP200 through a USB interface. FSP200 Development Kit can be used for a quick evaluation of the FSP200. A generalized system diagram is shown in Figure 5.

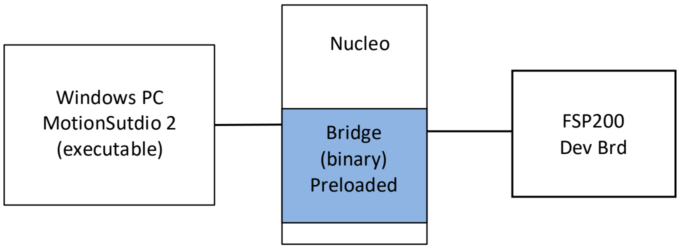


Figure 5: Simplified System Diagram with Freespace™ MotionStudio 2

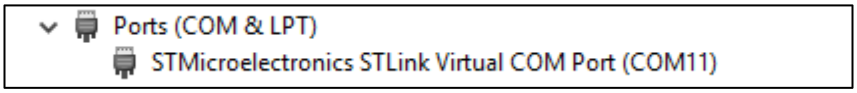
2.2. Running a PC Demo Application with FSP200 Development Kit

2.2.1. Requirement

Running Freespace™ MotionStudio 2 with FSP200 development kit requires the following items.

- ST-LINK/V2 USB driver available from the ST website (http://www.st.com/en/embedded-software/stsw-link009.html).
- ST32 Virtual COM Port Driver from ST website (http://www.st.com/en/development-tools/stsw-stm32102.html). The FSP200 software package is tested with STSW version 1.4.0. Once you downloaded and extracted the driver, follow the readme.txt file for the instruction to complete the installation.
- Freespace™ MotionStudio 2 application from https://www.hillcrestlabs.com/downloads/freespace-motionstudio-2

Connect USB Type A to Mini-B cable to Nucleo board and your PC. The virtual COM port should appear in your Device Manager.



Ports (COM & LPT)
STMicroelectronics STLink Virtual COM Port (COM11)

Figure 6: Device Manager to Check Installed Driver for ST Virtual COM Port

Start Freespace™ MotionStudio 2 (MotionStudio2.exe) after FSP200 development kit virtual COM port is successfully detected in your PC.

2.2.2. Running PC Application

Start Freespace™ MotionStudio 2

After you unzip the PC Application package, launch MotionStudio2.exe under MotionStudio2 folder. This will open MotionStudio2 window.

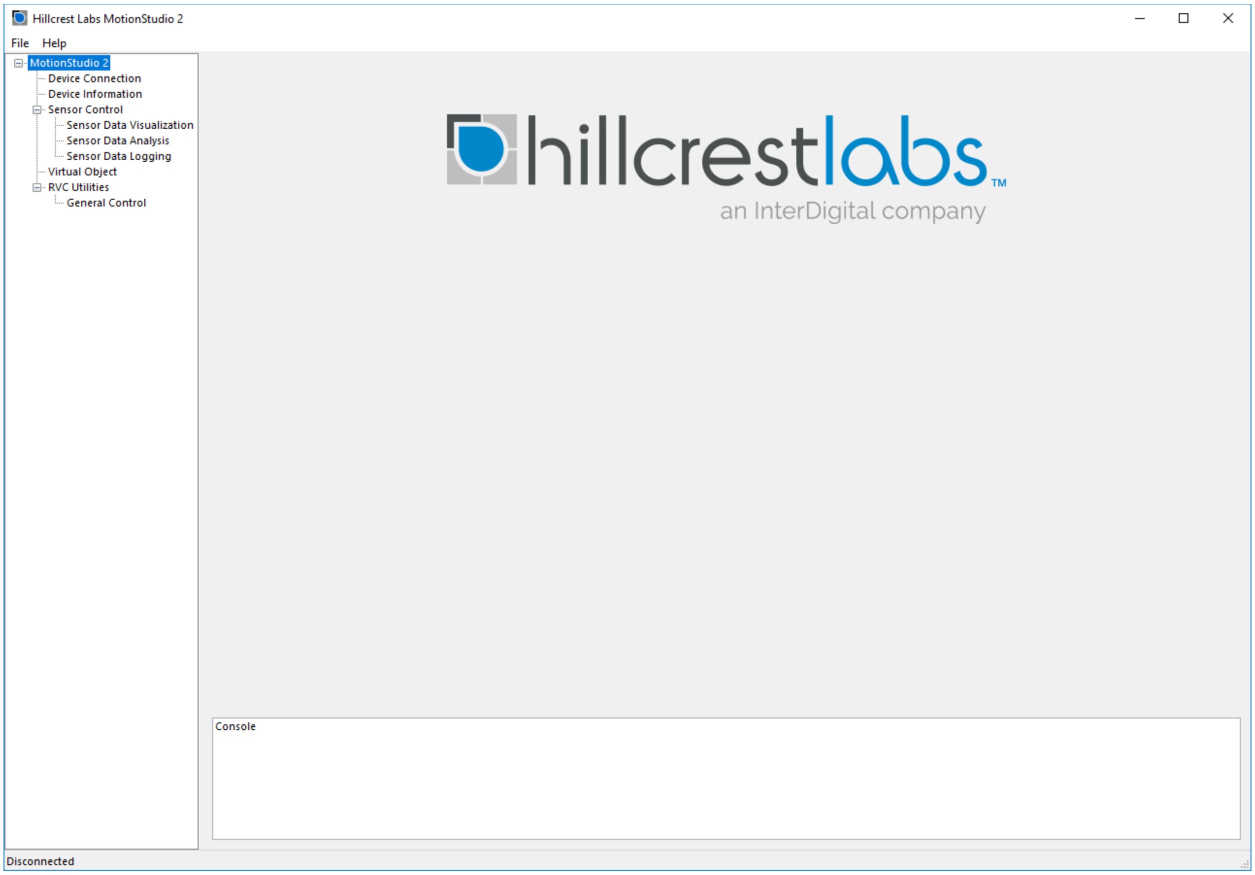
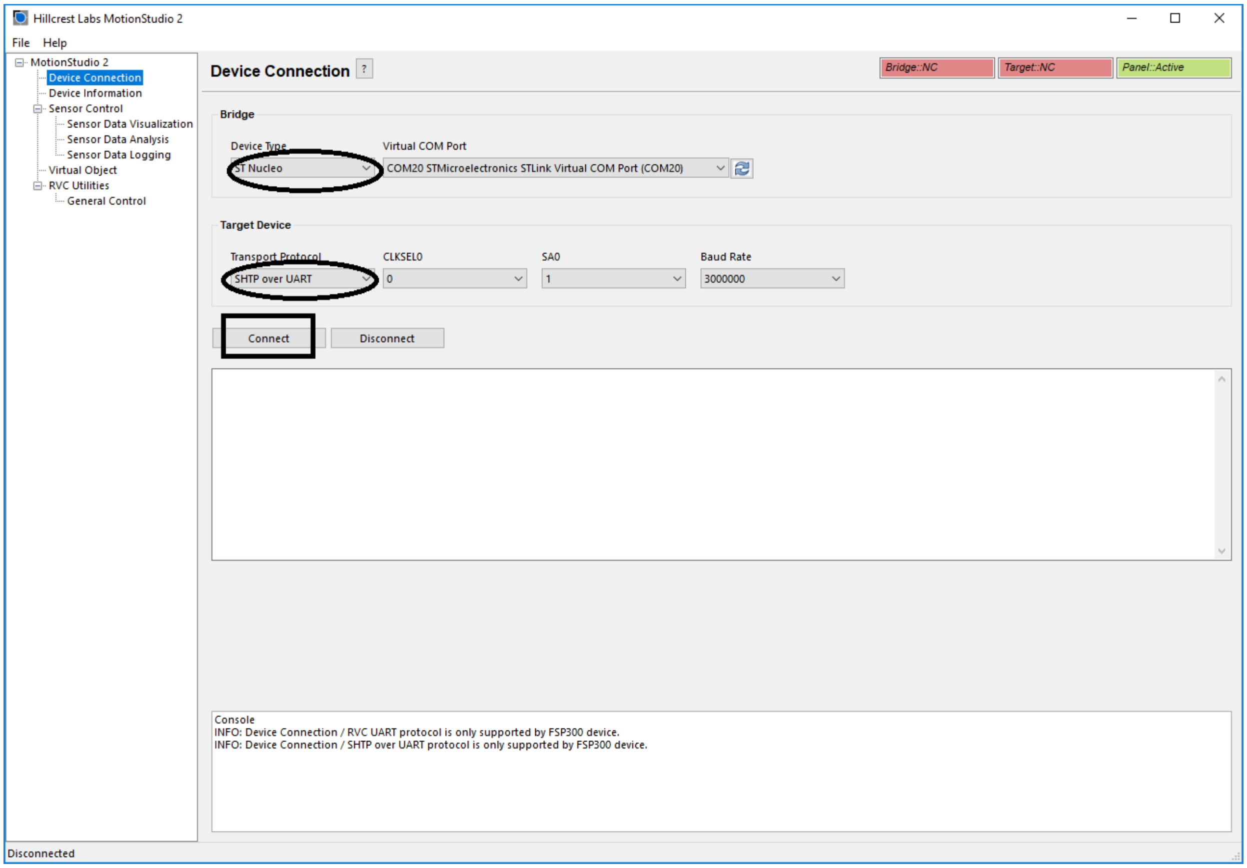


Figure 7: Startup Window of Freespace™ MotionStudio 2

Establish Connection to the Nucleo Board

From the menu panel on the left, select Device Connection. This panel allows users to select device type, transport protocol and more.

- “Device Type” of the bridge is set to ST Nucleo.
- ST Link Virtual COM Port available in your PC appears in “Virtual COM Port” box.
- “Target Device” is set to SHTP over I2C by default, but only SHTP over UART and RVC UART are available for FSP200. Refer to Section 1.3 for transport protocol selection. The ST Nucleo will set the PS0 and PS1 pins to the proper state based on the protocol selection.
- Use “Connect” button to start.



Bridge

Device Type: ST Nucleo

Virtual COM Port: COM20 STMicroelectronics STLink Virtual COM Port (COM20)

Target Device

Transport Protocol: SHTP over UART

CLKSEL0: 0

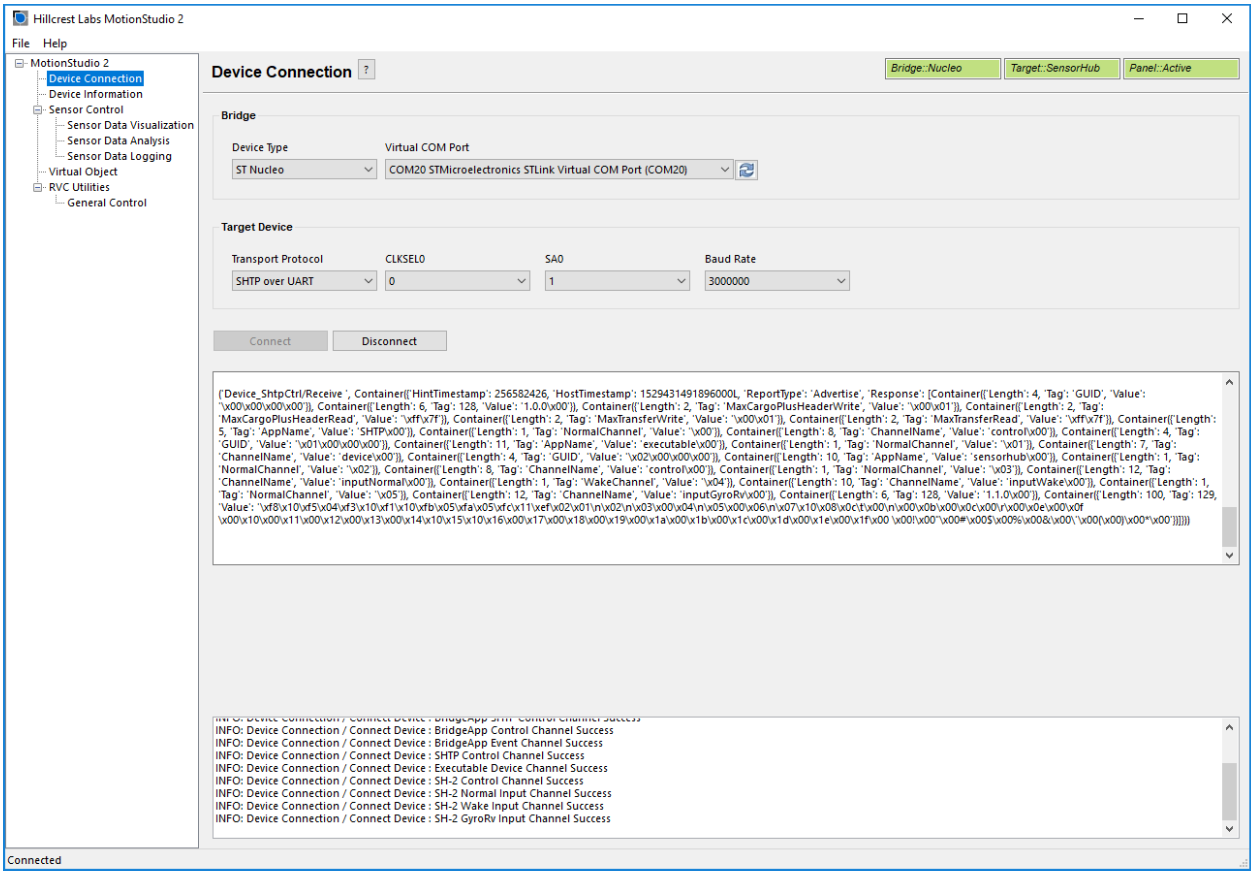
SA0: 1

Baud Rate: 3000000

```
Console
INFO: Device Connection / RVC UART protocol is only supported by FSP300 device.
INFO: Device Connection / SHTP over UART protocol is only supported by FSP300 device.
```

Disconnected

Figure 8 Device Connection Panel in Freespace™ MotionStudio 2



Device Type: ST Nucleo

Virtual COM Port: COM20 STMicroelectronics STLink Virtual COM Port (COM20)

Transport Protocol: SHTP over UART

CLKSEL0: 0

SA0: 1

Baud Rate: 3000000

```
('Device_ShtpCtrl/Receive ', Container({'HintTimestamp': 256582426, 'HostTimestamp': 1529431491896000L, 'ReportType': 'Advertise', 'Response': [Container({'Length': 4, 'Tag': 'GUID', 'Value':
'\x00\x00\x00\x00'}), Container({'Length': 6, 'Tag': 128, 'Value': '1.0.0\x00'}), Container({'Length': 2, 'Tag': 'MaxCargoPlusHeaderWrite', 'Value': '\x00\x01'}), Container({'Length': 2, 'Tag':
'MaxCargoPlusHeaderRead', 'Value': '\xff\x7f'}), Container({'Length': 2, 'Tag': 'MaxTransferWrite', 'Value': '\x00\x01'}), Container({'Length': 2, 'Tag': 'MaxTransferRead', 'Value': '\xff\x7f'}), Container({'Length':
5, 'Tag': 'AppName', 'Value': 'SHTP\x00'}), Container({'Length': 1, 'Tag': 'NormalChannel', 'Value': '\x00'}), Container({'Length': 8, 'Tag': 'ChannelName', 'Value': 'control\x00'}), Container({'Length': 4, 'Tag':
'GUID', 'Value': '\x01\x00\x00\x00'}), Container({'Length': 11, 'Tag': 'AppName', 'Value': 'executable\x00'}), Container({'Length': 1, 'Tag': 'NormalChannel', 'Value': '\x01'}), Container({'Length': 7, 'Tag':
'ChannelName', 'Value': 'device\x00'}), Container({'Length': 4, 'Tag': 'GUID', 'Value': '\x02\x00\x00\x00'}), Container({'Length': 10, 'Tag': 'AppName', 'Value': 'sensorhub\x00'}), Container({'Length': 1, 'Tag':
'NormalChannel', 'Value': '\x02'}), Container({'Length': 8, 'Tag': 'ChannelName', 'Value': 'control\x00'}), Container({'Length': 1, 'Tag': 'NormalChannel', 'Value': '\x03'}), Container({'Length': 12, 'Tag':
'ChannelName', 'Value': 'inputNormal\x00'}), Container({'Length': 1, 'Tag': 'WakeChannel', 'Value': '\x04'}), Container({'Length': 10, 'Tag': 'ChannelName', 'Value': 'inputWake\x00'}), Container({'Length': 1,
'Tag': 'NormalChannel', 'Value': '\x05'}), Container({'Length': 12, 'Tag': 'ChannelName', 'Value': 'inputGyroRv\x00'}), Container({'Length': 6, 'Tag': 128, 'Value': '1.1.0\x00'}), Container({'Length': 100, 'Tag': 129,
'Value': '\xf8\x10\xf5\x04\xf3\x10\xf1\x10\xfb\x05\xfa\x05\xfc\x11\xef\x02\x01\n\x02\n\x03\x00\x04\n\x05\x00\x06\n\x07\x10\x08\x0c\t\x00\n\x00\x0b\x00\x0c\x00\r\x00\x0e\x00\x0f
\x00\x10\x00\x11\x00\x12\x00\x13\x00\x14\x10\x15\x10\x16\x00\x17\x00\x18\x00\x19\x00\x1a\x00\x1b\x00\x1c\x00\x1d\x00\x1e\x00\x1f\x00 \x00!\x00"\x00#\x00$\x00%\x00&\x00\'\x00(\x00)\x00*\x00'})]}))
```

```
INFO: Device Connection / Connect Device : BridgeApp SHTP Control Channel Success
INFO: Device Connection / Connect Device : BridgeApp Control Channel Success
INFO: Device Connection / Connect Device : BridgeApp Event Channel Success
INFO: Device Connection / Connect Device : SHTP Control Channel Success
INFO: Device Connection / Connect Device : Executable Device Channel Success
INFO: Device Connection / Connect Device : SH-2 Control Channel Success
INFO: Device Connection / Connect Device : SH-2 Normal Input Channel Success
INFO: Device Connection / Connect Device : SH-2 Wake Input Channel Success
INFO: Device Connection / Connect Device : SH-2 GyroRv Input Channel Success
```

Figure 9: Device Connection Window after Successful Communication in Freespace™ MotionStudio 2

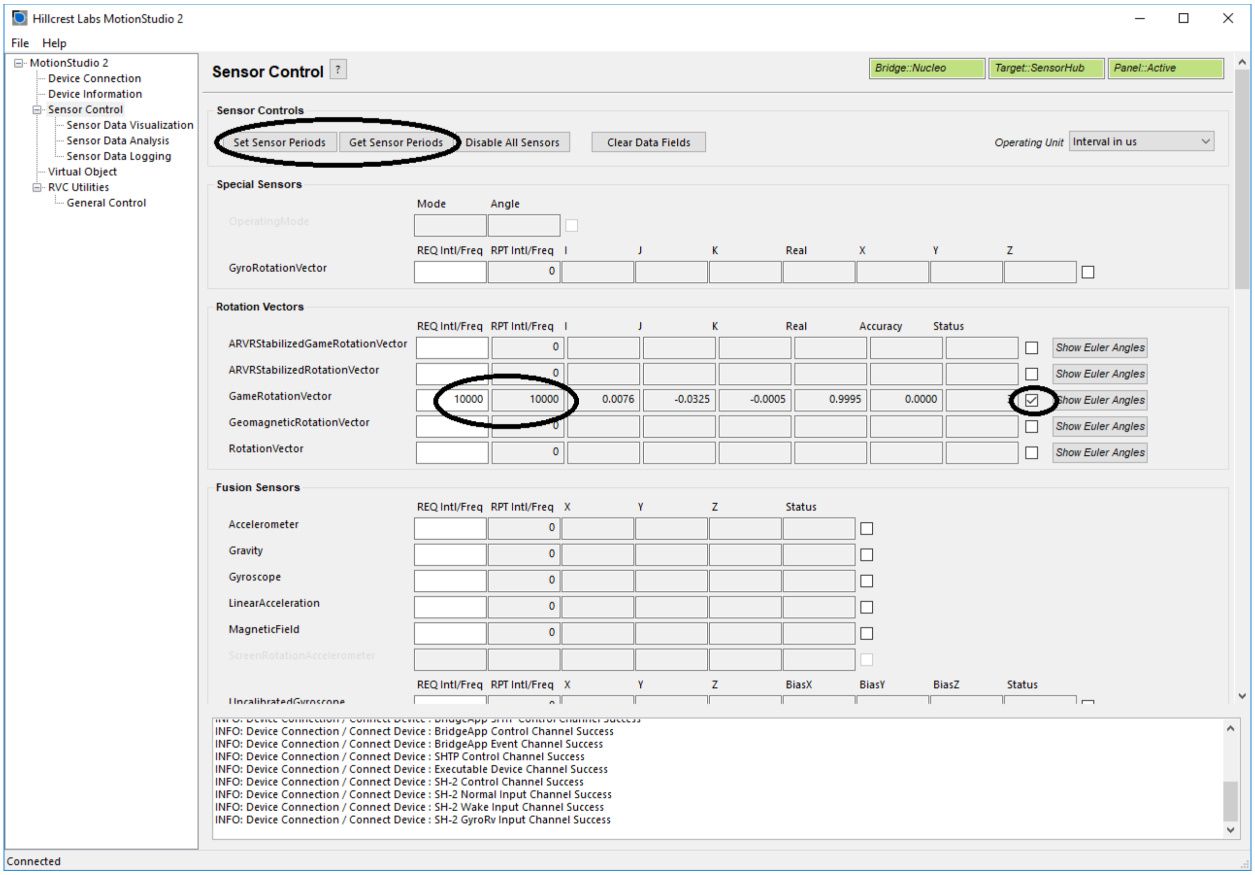
When connection process is completed, the three status indicator text boxes on the upper right corner of the panel and the console window on the bottom provide the result of connection process. The three status indicators show the status of the connected system and the status of the associated panel. If the specific panel supports the protocol used by the connected device, the panel becomes active and shows in green color.

Sensor Control

The Sensor Control panel allows the user to enable and disable the various sensors individually. There are two ways to control sensors:

- To enable an individual sensor at a default operation rate, use the check box on the right end of the row for each sensor.
- To enable sensors at specific rates, input the requested operating period, in microseconds, in the 'Requested Period (us)" fields. Then click the "Set Sensor Periods" button on the top of the panel. All sensors will be updated with the specified operating period. The "Requested Period (us)" fields which are left blank or have invalid values are assumed to be "zero".

In many cases, the sensors do not operate at the exact rate as requested. The actual operating period is shown in the "Reported Period (us)" field. Users can also use the "Get Sensor Periods" button on top of the panel to refresh the actual operating period for all sensors.



Operating Unit: Interval in us

Special Sensors

	Mode	Angle							
OperatingMode			□						

	REQ Intl/Freq	RPT Intl/Freq	I	J	K	Real	X	Y	Z	
GyroRotationVector		0								□

Rotation Vectors

	REQ Intl/Freq	RPT Intl/Freq	I	J	K	Real	Accuracy	Status		
ARVRStabilizedGameRotationVector		0							□	Show Euler Angles
ARVRStabilizedRotationVector		0							□	Show Euler Angles
GameRotationVector	10000	10000	0.0076	-0.0325	-0.0005	0.9995	0.0000		☑	Show Euler Angles
GeomagneticRotationVector		0							□	Show Euler Angles
RotationVector		0							□	Show Euler Angles

Fusion Sensors

	REQ Intl/Freq	RPT Intl/Freq	X	Y	Z	Status	
Accelerometer		0					□
Gravity		0					□
Gyroscope		0					□
LinearAcceleration		0					□
MagneticField		0					□
ScreenRotationAccelerometer							□

	REQ Intl/Freq	RPT Intl/Freq	X	Y	Z	BiasX	BiasY	BiasZ	Status
UncalibratedGyroscope		0							

```
INFO: Device Connection / Connect Device : BridgeApp SHTP Control Channel Success
INFO: Device Connection / Connect Device : BridgeApp Control Channel Success
INFO: Device Connection / Connect Device : BridgeApp Event Channel Success
INFO: Device Connection / Connect Device : SHTP Control Channel Success
INFO: Device Connection / Connect Device : Executable Device Channel Success
INFO: Device Connection / Connect Device : SH-2 Control Channel Success
INFO: Device Connection / Connect Device : SH-2 Normal Input Channel Success
INFO: Device Connection / Connect Device : SH-2 Wake Input Channel Success
INFO: Device Connection / Connect Device : SH-2 GyroRv Input Channel Success
```

Connected

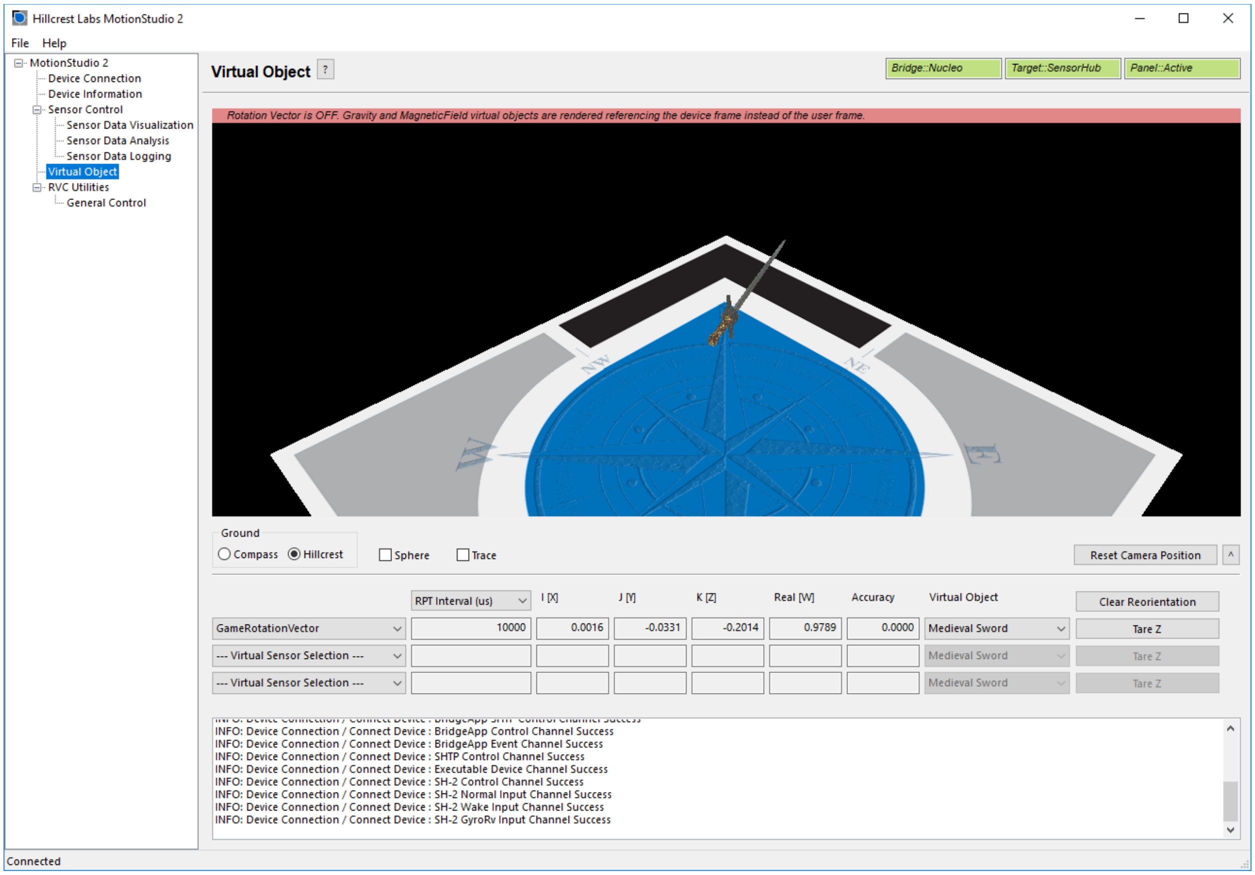
Figure 10: Sensor Control Panel in Freespace™ MotionStudio 2

Virtual Object

Virtual Object panel shows the orientation of the device. Please note that you need to enable sensors in **Sensor Control** panel, then select the sensor from the drop down menu in **Virtual Object** panel. The sword in the Virtual Object will move according to the device orientation.

To adjust the camera position, move the cursor to the Virtual object Panel, then press the LEFT mouse button. Hold the button down and move the mouse to change the view position. To reset the camera position, use the "Reset Camera Position" button.

To display the game rotation vectors, select the game rotation vectors from the drop down menu, the data fields should start updating with the received sensor data. The virtual object will move according to the orientation of the hardware. Use the Sensor Control Panel to enable or disable the specific sensor. This panel does not control the sensor but displays the output data.



Rotation Vector is OFF. Gravity and MagneticField virtual objects are rendered referencing the device frame instead of the user frame.

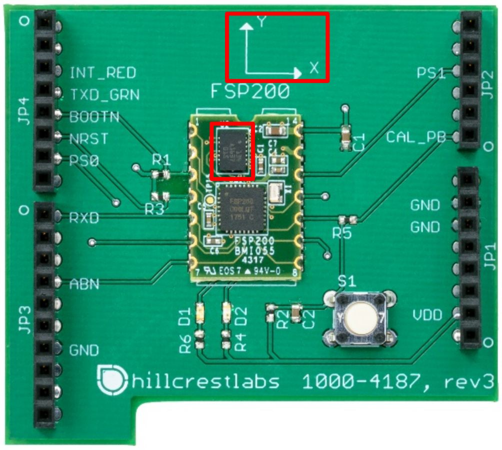
Ground: ○ Compass ● Hillcrest

□ Sphere □ Trace

	RPT Interval (us)	I [X]	J [Y]	K [Z]	Real [W]	Accuracy	Virtual Object	Clear Reorientation
GameRotationVector	10000	0.0016	-0.0331	-0.2014	0.9789	0.0000	Medieval Sword	Tare Z
--- Virtual Sensor Selection ---							Medieval Sword	Tare Z
--- Virtual Sensor Selection ---							Medieval Sword	Tare Z

```
INFO: Device Connection / Connect Device : BridgeApp SHTP Control Channel Success
INFO: Device Connection / Connect Device : BridgeApp Control Channel Success
INFO: Device Connection / Connect Device : BridgeApp Event Channel Success
INFO: Device Connection / Connect Device : SHTP Control Channel Success
INFO: Device Connection / Connect Device : Executable Device Channel Success
INFO: Device Connection / Connect Device : SH-2 Control Channel Success
INFO: Device Connection / Connect Device : SH-2 Normal Input Channel Success
INFO: Device Connection / Connect Device : SH-2 Wake Input Channel Success
INFO: Device Connection / Connect Device : SH-2 GyroRv Input Channel Success
```

Figure 11: Virtual Object Panel in Freespace™ MotionStudio 2

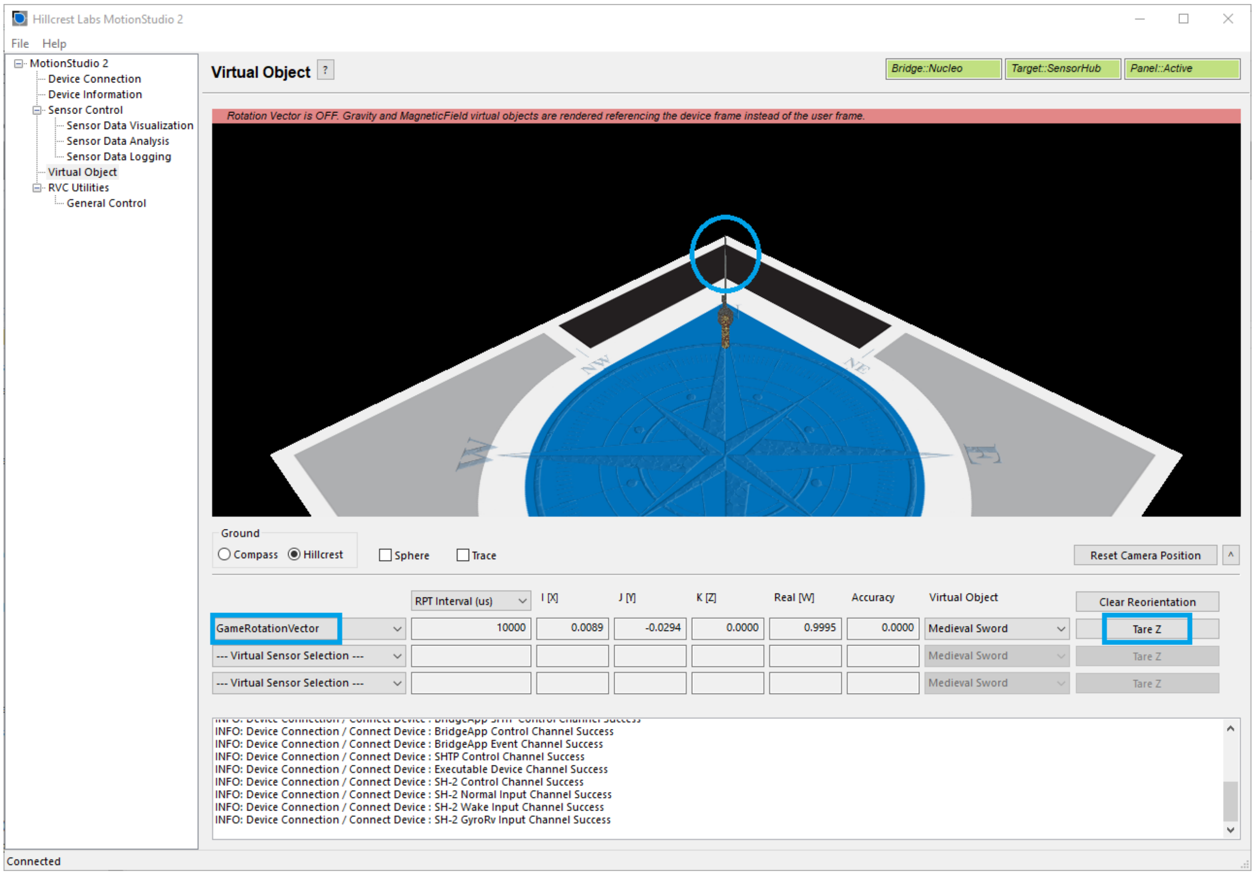


X/Y as marked on the board, using right handed rule

Figure 12: FSP200 Device Orientation

Please follow the instructions below to align your device.

- Enable Game Rotation Vector and ARVR Stabilized Game Rotation Vector in “Sensor Control” panel.
- Switch to Virtual Object panel and move the background so the black corner of the Hillcrest logo on the ground plane points to your forward direction (heading).
- Hold the FSP200 development kit Y+ axis points to your forward direction as well.
- Select “Game Rotation Vector” in drop down menu and click “Tare Z”. Now, the sword will point to the edge of the Hillcrest logo and is aligned with your device Y+.



Rotation Vector is OFF. Gravity and MagneticField virtual objects are rendered referencing the device frame instead of the user frame.

	RPT Interval (us)	I [X]	J [Y]	K [Z]	Real [W]	Accuracy	Virtual Object	
GameRotationVector	10000	0.0089	-0.0294	0.0000	0.9995	0.0000	Medieval Sword	Tare Z
--- Virtual Sensor Selection ---							Medieval Sword	Tare Z
--- Virtual Sensor Selection ---							Medieval Sword	Tare Z

```
INFO: Device Connection / Connect Device : BridgeApp Control Channel Success
INFO: Device Connection / Connect Device : BridgeApp Event Channel Success
INFO: Device Connection / Connect Device : SHTP Control Channel Success
INFO: Device Connection / Connect Device : Executable Device Channel Success
INFO: Device Connection / Connect Device : SH-2 Control Channel Success
INFO: Device Connection / Connect Device : SH-2 Normal Input Channel Success
INFO: Device Connection / Connect Device : SH-2 Wake Input Channel Success
INFO: Device Connection / Connect Device : SH-2 GyroRv Input Channel Success
```

Figure 13: Sensor Orientation in Virtual Object Panel in Freespace™ MotionStudio 2

2.2.3. Backup Bridge Image

ⓘ Please keep in mind that FSP200 Development Kit is programmed to work with a Windows PC Application. Development example code explained in the Section 2.4 is different and will overwrite the binary image shipped with the FSP200 Development Kit. If you want to switch back to the bridge code to run the Windows PC Application, please either download the latest bridge firmware from Hillcrest website(https://www.hillcrestlabs.com/downloads/bridge-firmware-for-nucleo), or read the Nucleo device memory using STM32 ST-LINK Utility before downloading a new image into the Nucleo board. Please follow the instructions below to back up the prebuilt bridge image.

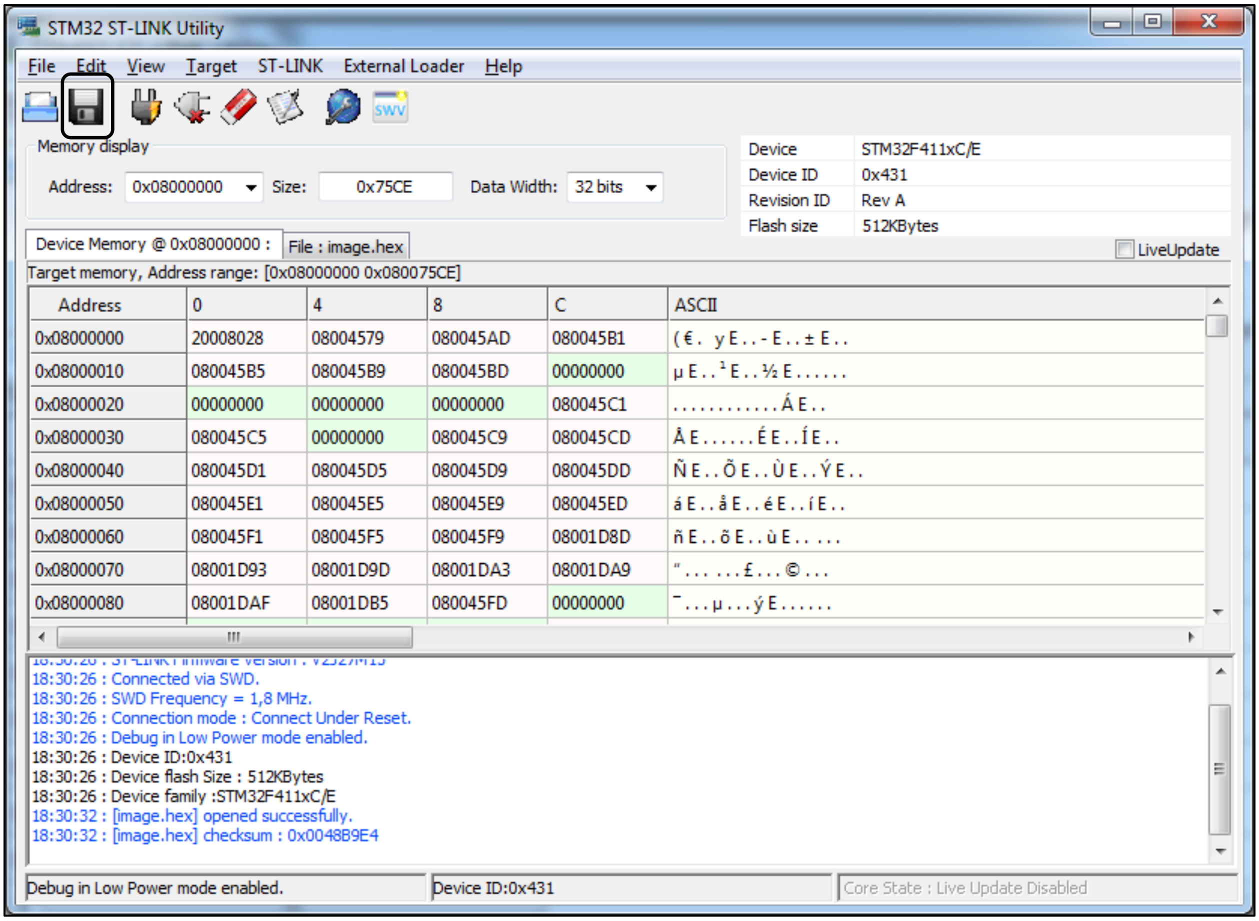
To save the contents of Nucleo bridge board:

- Open STM32 ST-LINK Utility
- Target->Connect
- Use “Save the displayed content in a Binary File” on the top left corner to save the image into a file for future use.

To program Nucleo bridge board with the saved image:

- Open STM32 ST-LINK Utility
- Target->Connect

- Target->Program
- In new window select “File path” to locate previously saved file.
- Click “Start”



Address: 0x08000000 Size: 0x75CE Data Width: 32 bits

Device	STM32F411xC/E
Device ID	0x431
Revision ID	Rev A
Flash size	512KBytes

Device Memory @ 0x08000000 : File : image.hex

□ LiveUpdate

Target memory, Address range: [0x08000000 0x080075CE]

Address	0	4	8	C	ASCII
0x08000000	20008028	08004579	080045AD	080045B1	(€. yE..-E..±E..
0x08000010	080045B5	080045B9	080045BD	00000000	µE..¹E..½E......
0x08000020	00000000	00000000	00000000	080045C1	............ÁE..
0x08000030	080045C5	00000000	080045C9	080045CD	ÅE......ÉE..ÍE..
0x08000040	080045D1	080045D5	080045D9	080045DD	ÑE..ÕE..ÙE..ÝE..
0x08000050	080045E1	080045E5	080045E9	080045ED	áE..åE..éE..íE..
0x08000060	080045F1	080045F5	080045F9	08001D8D	ñE..õE..ùE.....
0x08000070	08001D93	08001D9D	08001DA3	08001DA9	"......£...©...
0x08000080	08001DAF	08001DB5	080045FD	00000000	¯...µ...ýE......

```
18:30:26 : ST-LINK Firmware version : V2J27M15
18:30:26 : Connected via SWD.
18:30:26 : SWD Frequency = 1,8 MHz.
18:30:26 : Connection mode : Connect Under Reset.
18:30:26 : Debug in Low Power mode enabled.
18:30:26 : Device ID:0x431
18:30:26 : Device flash Size : 512KBytes
18:30:26 : Device family :STM32F411xC/E
18:30:32 : [image.hex] opened successfully.
18:30:32 : [image.hex] checksum : 0x0048B9E4
```

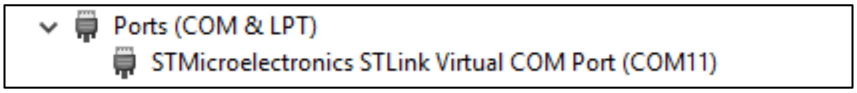
Debug in Low Power mode enabled. | Device ID:0x431 | Core State : Live Update Disabled

Figure 14: STM32 ST-LINK Utility Window

2.3. Development Environment

The example software requires the following items to execute.

- IAR Embedded Workbench® for ARM (EWARM) by IAR Systems.
- ST-LINK/V2 USB driver. This driver is available from the ST website and is supported by the IAR Embedded Workbench for ARM (EWARM). After installing EWARM, check IAR_INSTALL_DIRECTORY\arm\drivers\ST-Link\ Please skip this step if you have installed already from Section 2.2.1.
- ST32 Virtual COM Port Driver from the ST website. Please skip this if you have installed already from Section 2.2.1.


Ports (COM & LPT)

STMicroelectronics STLink Virtual COM Port (COM11)

Figure 15: Installed driver for ST Virtual COM port

- Terminal emulator software like Tera Term or PuTTy. Set up the terminal emulator at 115200 – 8bit – no parity – 1bit stop bit – no flow control.

2.4. Example Software

Hillcrest provides a complete software package for the STM32F411RE Nucleo boards.

The example application for FSP200 development kit source code is available in public github.

https://github.com/hcrest/sh2-demo-nucleo

Clone this repository using the --recursive flag with git. Alternatively, you can download a ZIP file from the link. An example git clone command is shown below.

git clone --recursive https://github.com/hcrest/sh2-demo-nucleo.git

Everything required to obtain outputs from the FSP200 is included in this package. The software package incorporates the FSP200 driver, enabling SH2 functionality for the development system.

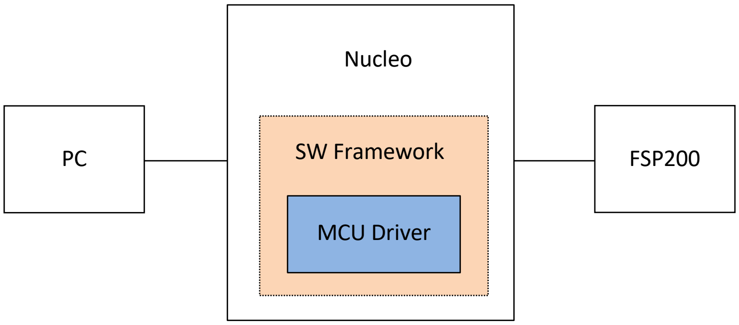


Figure 16: Simplified System Diagram (blue indicates driver developed by Hillcrest)

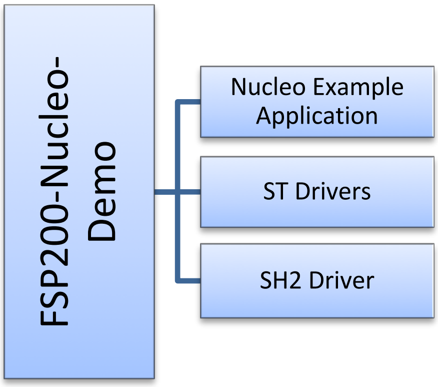


Figure 17: Source Code Structure

The software is organized as an IAR EWARM project that can be dropped into the IAR IDE on a Windows PC. Follow this procedure to compile the project and download the software to the Nucleo board.

- Open IAR Embedded Workbench for ARM (EWARM)
- In the File menu, select Open and choose Workspace. Browse to where the example package is extracted and select “sh2-demo-nucleo/EWARM/Project.eww”. This should open an IAR workspace with all the files within the project.
- Select ‘demo-uart’ or ‘fsp200-cal’ in the project configuration. ‘demo-i2c’ and ‘demo-spi’ are reserved for BNO08X or FSM30X utilizing the same SH2 protocol, but with different interfaces. And fsp200-dfu is used for upgrading the FSP200, please contact Hillcrest Labs for details.
- In the “Project” menu, select “Rebuild All” to compile the project.
- After the project is successfully compiled, go to the Project menu and select Download and Debug.

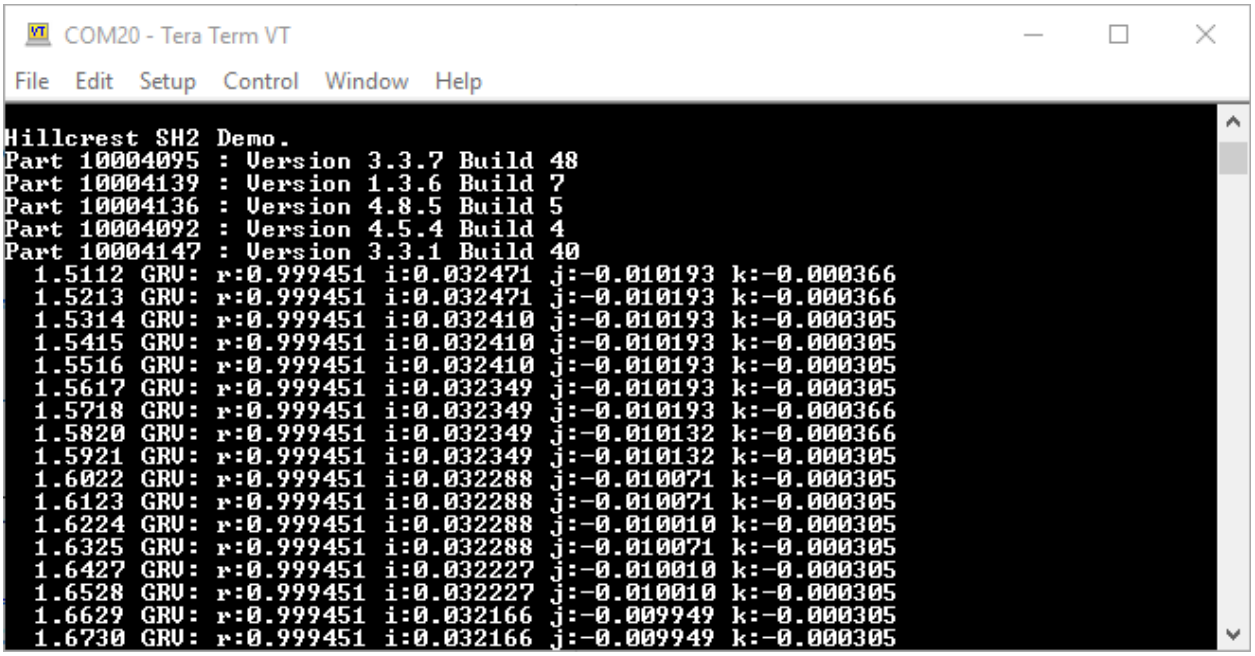
The “sh2” directory contains a full implementation of the Hillcrest communications protocol for the FSP200.

The reader is encouraged to review the FSP200 datasheet [1] and the SH-2 Reference Manual [2] for details on how to construct messages.

The output from the FSP200 is printed through the serial port. The first few lines indicate that the host has established proper communication (“Product ID Request”) with the FSP200 and the FSP200 has responded with version information (“Product ID Response”).

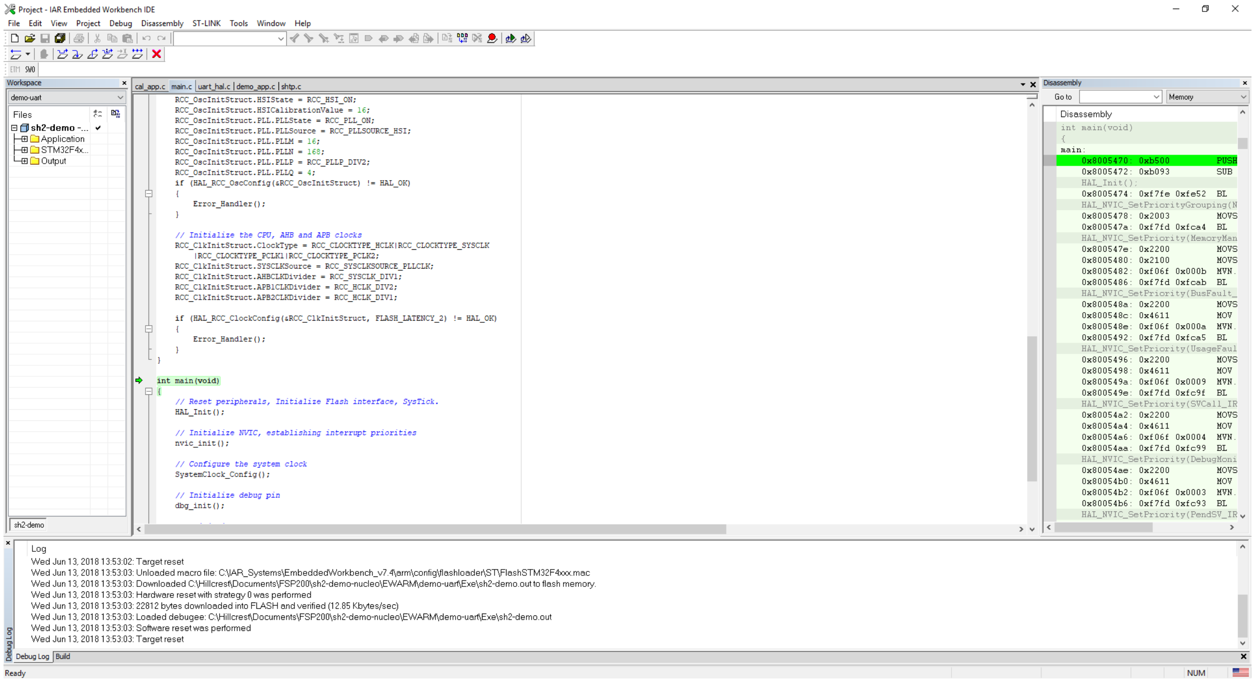
- demo-uart project:

Game Rotation vector is enabled at 100Hz by default and reports are printed through the serial port.



```
Hillcrest SH2 Demo.
Part 10004095 : Version 3.3.7 Build 48
Part 10004139 : Version 1.3.6 Build 7
Part 10004136 : Version 4.8.5 Build 5
Part 10004092 : Version 4.5.4 Build 4
Part 10004147 : Version 3.3.1 Build 40
  1.5112 GRV: r:0.999451 i:0.032471 j:-0.010193 k:-0.000366
  1.5213 GRV: r:0.999451 i:0.032471 j:-0.010193 k:-0.000366
  1.5314 GRV: r:0.999451 i:0.032410 j:-0.010193 k:-0.000305
  1.5415 GRV: r:0.999451 i:0.032410 j:-0.010193 k:-0.000305
  1.5516 GRV: r:0.999451 i:0.032410 j:-0.010193 k:-0.000305
  1.5617 GRV: r:0.999451 i:0.032349 j:-0.010193 k:-0.000305
  1.5718 GRV: r:0.999451 i:0.032349 j:-0.010193 k:-0.000366
  1.5820 GRV: r:0.999451 i:0.032349 j:-0.010132 k:-0.000366
  1.5921 GRV: r:0.999451 i:0.032349 j:-0.010132 k:-0.000305
  1.6022 GRV: r:0.999451 i:0.032288 j:-0.010071 k:-0.000305
  1.6123 GRV: r:0.999451 i:0.032288 j:-0.010071 k:-0.000305
  1.6224 GRV: r:0.999451 i:0.032288 j:-0.010010 k:-0.000305
  1.6325 GRV: r:0.999451 i:0.032288 j:-0.010071 k:-0.000305
  1.6427 GRV: r:0.999451 i:0.032227 j:-0.010010 k:-0.000305
  1.6528 GRV: r:0.999451 i:0.032227 j:-0.010010 k:-0.000305
  1.6629 GRV: r:0.999451 i:0.032166 j:-0.009949 k:-0.000305
  1.6730 GRV: r:0.999451 i:0.032166 j:-0.009949 k:-0.000305
```

Figure 18: Terminal Emulator Screenshot


Project - IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly ST-LINK Tools Window Help
Workspace
demo-uart
Files
sh2-demo -...
Application
STM32F4x...
Output
sh2-demo
cal_app.c main.c uart_hal.c demo_app.c shtp.c
Disassembly
Go to
Memory
Debug Log Build
Ready
NUM

```
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = 16;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
    RCC_OscInitStruct.PLL.PLLM = 16;
    RCC_OscInitStruct.PLL.PLLN = 168;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 4;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }

    // Initialize the CPU, AHB and APB clocks
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
        |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
    {
        Error_Handler();
    }
}

int main(void)
{
    // Reset peripherals, Initialize Flash interface, SysTick.
    HAL_Init();

    // Initialize NVIC, establishing interrupt priorities
    nvic_init();

    // Configure the system clock
    SystemClock_Config();

    // Initialize debug pin
    dbg_init();
```

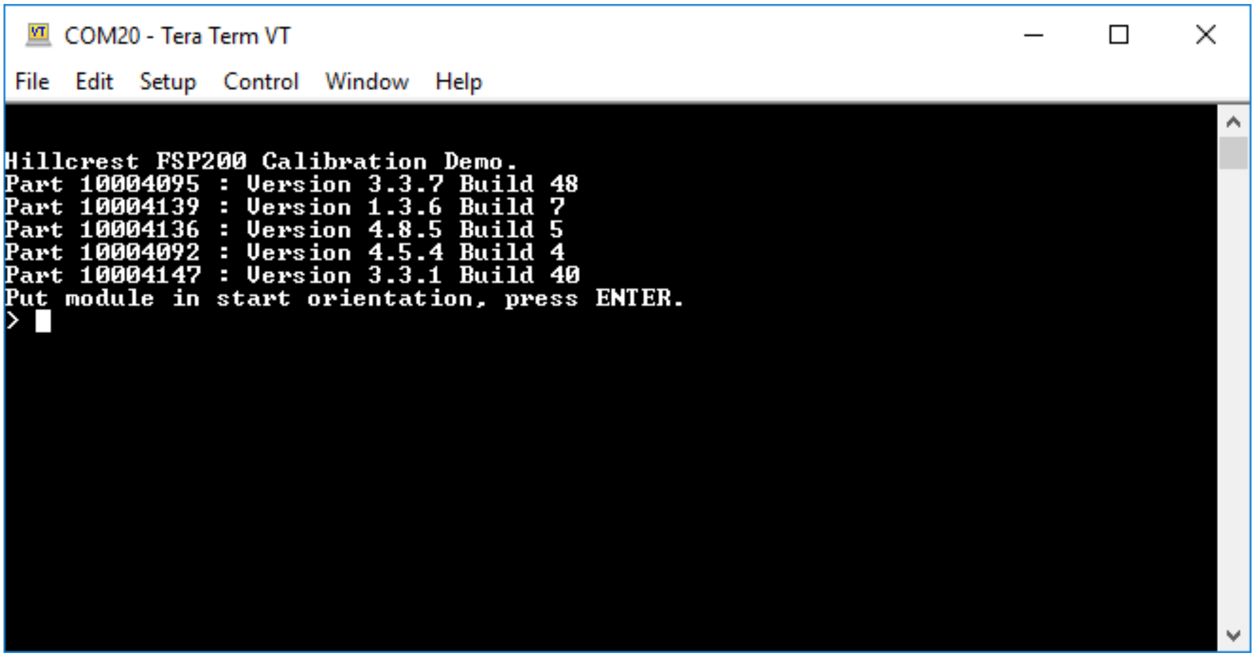
```
Disassembly
int main(void)
{
main:
    0x8005470: 0xb500         PUSH
    0x8005472: 0xb093         SUB
    HAL_Init();
    0x8005474: 0xf7fe 0xfe52  BL
    HAL_NVIC_SetPriorityGrouping(N
    0x8005478: 0x2003         MOVS
    0x800547a: 0xf7fd 0xfca4  BL
    HAL_NVIC_SetPriority(MemoryMan
    0x800547e: 0x2200         MOVS
    0x8005480: 0x2100         MOVS
    0x8005482: 0xf06f 0x000b  MVN.
    0x8005486: 0xf7fd 0xfcab  BL
    HAL_NVIC_SetPriority(BusFault_
    0x800548a: 0x2200         MOVS
    0x800548c: 0x4611         MOV
    0x800548e: 0xf06f 0x000a  MVN.
    0x8005492: 0xf7fd 0xfca5  BL
    HAL_NVIC_SetPriority(UsageFaul
    0x8005496: 0x2200         MOVS
    0x8005498: 0x4611         MOV
    0x800549a: 0xf06f 0x0009  MVN.
    0x800549e: 0xf7fd 0xfc9f  BL
    HAL_NVIC_SetPriority(SVCall_IR
    0x80054a2: 0x2200         MOVS
    0x80054a4: 0x4611         MOV
    0x80054a6: 0xf06f 0x0004  MVN.
    0x80054aa: 0xf7fd 0xfc99  BL
    HAL_NVIC_SetPriority(DebugMoni
    0x80054ae: 0x2200         MOVS
    0x80054b0: 0x4611         MOV
    0x80054b2: 0xf06f 0x0003  MVN.
    0x80054b6: 0xf7fd 0xfc93  BL
    HAL_NVIC_SetPriority(PendSV_IR
```

```
Log
Wed Jun 13, 2018 13:53:02: Target reset
Wed Jun 13, 2018 13:53:03: Unloaded macro file: C:\IAR_Systems\EmbeddedWorkbench_v7.4\arm\config\flashloader\ST\FlashSTM32F4xxx.mac
Wed Jun 13, 2018 13:53:03: Downloaded C:\Hillcrest\Documents\FSP200\sh2-demo-nucleo\EWARM\demo-uart\Exe\sh2-demo.out to flash memory.
Wed Jun 13, 2018 13:53:03: Hardware reset with strategy 0 was performed
Wed Jun 13, 2018 13:53:03: 22812 bytes downloaded into FLASH and verified (12.85 Kbytes/sec)
Wed Jun 13, 2018 13:53:03: Loaded debugee: C:\Hillcrest\Documents\FSP200\sh2-demo-nucleo\EWARM\demo-uart\Exe\sh2-demo.out
Wed Jun 13, 2018 13:53:03: Software reset was performed
Wed Jun 13, 2018 13:53:03: Target reset
```

Figure 19: IAR EWARM Screenshot

- fsp200-cal project:

Simple Calibration mode (software controlled method) is entered. The user should follow the instructions on the serial window to perform the calibration. A precise 180 degree rotation is required between the start orientation and the final orientation. After 'Calibration completed successfully', it will run into the next cycle of calibration.



```
Hillcrest FSP200 Calibration Demo.
Part 10004095 : Version 3.3.7 Build 48
Part 10004139 : Version 1.3.6 Build 7
Part 10004136 : Version 4.8.5 Build 5
Part 10004092 : Version 4.5.4 Build 4
Part 10004147 : Version 3.3.1 Build 40
Put module in start orientation, press ENTER.
> 
```

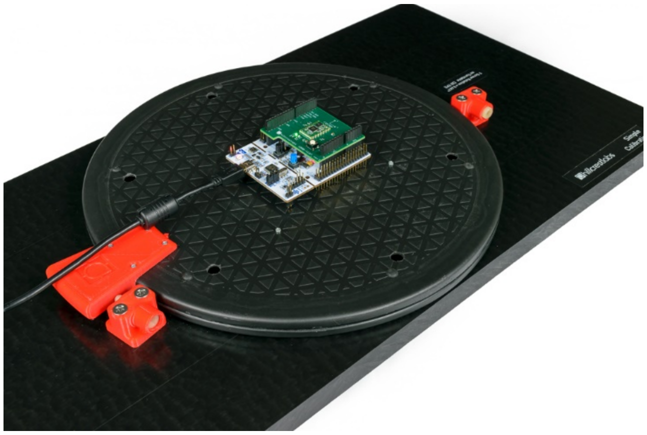
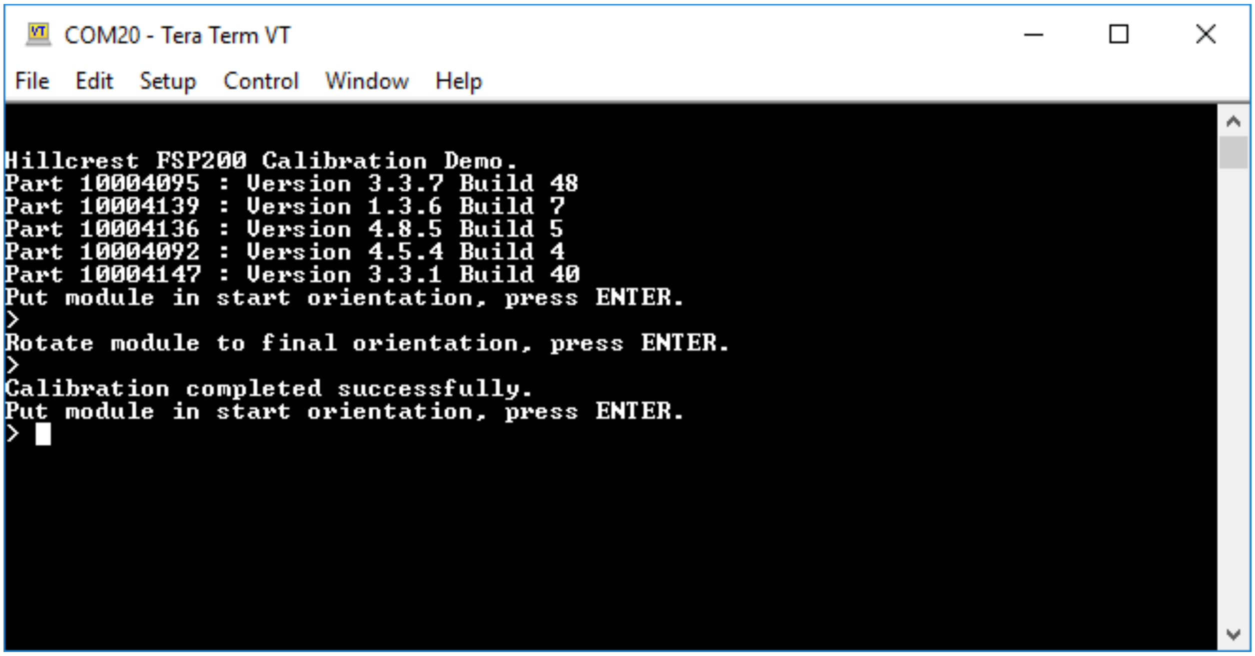


Figure 20: Simple Calibration Procedure - Start



```
Hillcrest FSP200 Calibration Demo.
Part 10004095 : Version 3.3.7 Build 48
Part 10004139 : Version 1.3.6 Build 7
Part 10004136 : Version 4.8.5 Build 5
Part 10004092 : Version 4.5.4 Build 4
Part 10004147 : Version 3.3.1 Build 40
Put module in start orientation, press ENTER.
>
Rotate module to final orientation, press ENTER.
>
Calibration completed successfully.
Put module in start orientation, press ENTER.
>
```

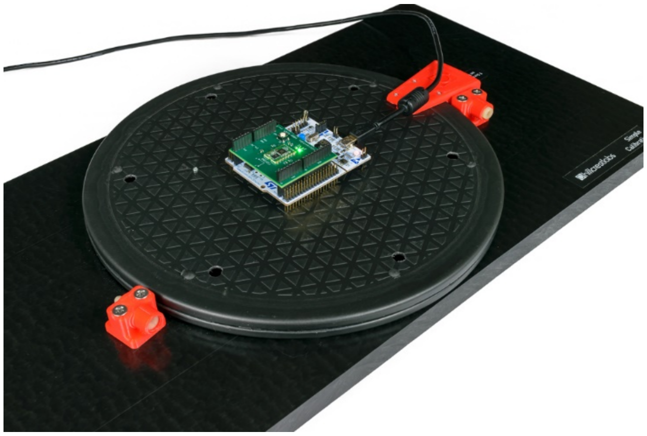


Figure 21: Simple Calibration Procedure - Complete

3. Simple Calibration

3.1. Hardware Controlled Simple Calibration Method

Hardware Controlled Simple Calibration only works in RVC mode.

This procedure follows the example design of calibration signals by pressing the Simple Calibration Button in Figure 4:

- Enter Calibration Mode: Push CAL_PB button to start calibration. Wait for the green LED (TX_LEDGRN) to turn on and flash.
- Rotate: When the green LED is flashing, rotate the device 180° clockwise and stop, then push CAL_PB button.
- Wait for the Calibration Results:

- The Green LED (TX_LEDGRN) will be turned on to indicate a successful calibration. Reset the device to use newly calibrated sensor records.
- The Red LED (INT_LEDRED) will be turned on to indicate a failed calibration. Rerun the calibration a couple of more times. If the calibration still returns a failure, the device is out of specification.

Refer to FSP200 Simple Calibration Usage Guide [3] for more details.

3.2. Software Controlled Simple Calibration Method

Software Controlled Simple Calibration only works in SHTP mode.

Configuration from the IAR Embedded Workbench *"fsp200-cal"* shows how to execute the FSP200 Simple Calibration. SH-2 Reference Manual [2] Section 6.4.11 also shows the details of the commands used in the example application.

- Enter Calibration Mode: *startCalStart(*) shows command to initiate the Simple Calibration. Wait for the command response, *startCalRx()*.
- Rotate: Once a valid response is received, rotate the device 180° clockwise, then send finish calibration command, *finishCalStart()*.
- Wait for the Calibration Results: wait for the finish calibration response, *finishCalRx()* and check the calibration status.

When Finish Calibration Response returns any error code, rerun the Simple Calibration for a couple of more times. If the calibration still returns fail, the device is out of specification.

References

1. 1000-4121 FSP200 Datasheet, Hillcrest Labs
2. 1000-3625 SH-2 Reference Manual, Hillcrest Labs
3. 1000-4155 FSP200 Simple Calibration Usage Guide, Hillcrest Labs
4. 1000-3535 Sensor Hub Transport Protocol, Hillcrest Labs

Notices

Freespace is a registered trademark of CEVA. Other company and product names mentioned in this document may be the trademark or registered trademark of their respective owners.

Disclaimer: The information furnished herein is believed to be accurate and reliable. However, the information is provided “AS IS”, without any express or implied warranty of any kind including warranties of merchantability, non-infringement of intellectual property, or fitness for any particular purpose.

In no event shall CEVA or its suppliers be liable for any claims and/or damages whatsoever arising out of the use of or inability to use the materials. CEVA and its suppliers further do not warrant the accuracy or completeness of the information, text, graphics or other items contained within these materials. CEVA may make changes to these materials, or to the products described within.

www.ceva-dsp.com

FOR MORE
INFORMATION: