BNO085 SPI quick start guide –

Contents

Introduction

Warning!! There are probably plenty of mistakes in this document.

I am a BNO085 amateur.

The aim is to give someone just enough information to get SPI communications working on the BNO085. I have only touched on the basics, but this should be enough to get you started.

We will cover the following –

1. Reading the unsolicited start up data
2. Requesting a product ID report
3. Setting the sensor up for a continuous rotation vector report

I have tried to make this a simple as possible so make sure that you read all sections.

Pins

I am using the Adafruit BNO085 breakout.

VIN – 3.3V

GND – common ground

SCL – microcontroller clock pin

SDA – microcontroller MISO pin

DI – microcontroller MOSI pin

CS – microcontroller digital pin

INT – microcontroller digital pin (interrupt capable pin required)

RST – microcontroller digital pin

P1 – 3.3V

P0 - microcontroller digital pin

Interrupt pin

You can't run SPI comms without being able to read the interrupt (INT) pin. When the BNO asserts this pin (the INT pin is active low) you have to read the awaiting data immediately. Don't read data until this pin is asserted. The only exception is when SHTP data on startup.

Wake Up

I am using Adafruites BNO085 breakout board. It's great and the Adafruit team does an amazing job. In their SPI setup they have the P0 and P1 pins pulled to 3.3V. I was unable to get my SPI comms working with this configuration. Instead I pulled only P1 to 3.3V. P0 is wired to a digital output pin on my micro and used for waking the device -

```
pinMode(BNO08X_WAKE, OUTPUT); // set the wake pin as an output
```

I have to wake the chip before sending the product ID request.
After waking the device and sending the product ID request we get 2 x product ID responses (explained in this document). The BNO goes to sleep after these responses and I again have to wake it.

So in total I have to wake the BN0 twice. Once before the product ID request and once after the product ID responses (before setting the sensor up for continuous reads)

Here is an example of how I wake the sensor up before requesting the product ID –

1. Write the wake pin low
2. Wait of the interrupt pin to assert (shows the device has woken up)
3. Write the chip select low
4. Write the wake pin high
5. Write or read from the device

```
//Write the wake pin low
digitalWrite(BNO08X_WAKE, LOW);
//Wait of the interrupt pin to assert
while(digitalRead(BNO08X_INT));
//Write the chip select low
digitalWrite(BNO08X_CS, LOW);
//Write the wake pin high
digitalWrite(BNO08X_WAKE, HIGH);
//Write or read from the device
      for (INT i=0; i<6; i++){
      cargo[i] =SPI.transfer(Product_ID[i]);
      Serial.print(cargo[i],HEX);
      Serial.print(" , ");
}
Serial.println("....");
digitalWrite(BNO08X_CS, HIGH);
```

SHTP Header

All reads and writes are preceed by the SHTP header. Even the unsolicited messages have the header preceeding –

When you read from the BNO, the header can be used to determine the number of bytes you have to read. The Length LSB and Length MSB include the header plus the rest of the payload. So for example, read the first 4 bytes from the BNO and determine the total number of bytes remaining.

Total number of bytes remaining = Payload length (derived from SHTP header length bytes) – 4
Don't forget that bit 15 is not included in the length field. It is used to indicate if payload data is part of a new payload or part of an old one. See the Hillcrest documentation for more info.

When writing to the BNO, you specify the totally bytes (including the header bytes) you are going to send.

2.2.1 SHTP Header

The SHTP header is shown in Figure 2.

Byte	Field
0	Length LSB
1	Length MSB
2	Channel
3	SeqNum

Figure 2: SHTP Header

Length	The MSB of the length field is used to indicate if a transfer is a continuation of a previous transfer. Bits 14:0 are used to indicate the total number of bytes in the cargo plus header, which may be spread over multiple messages. The bytes in the header field are counted as part of the length. A length of 65535 (0xFFFF) is reserved because a failed peripheral can too easily produce 0xFFFF. Therefore, the largest cargo that can be transported is 32766 minus the header bytes.
Channel	The channel number of the cargo. Channel 0 is the command channel and is used by the SHTP.
SeqNum	The sequence number of the cargo. The sequence number is a monotonically incrementing number that increments once for each cargo sent or cargo continuation sent. Each channel has its own sequence number. The sequence number is used to detect duplicate or missing cargoes and to associate segmented cargoes with each other.

Channels

The BNO085 supports 6 channels.

I guess a good way to think about channels is that each channel is used to send a different family of data.
When writing to the device, you place the required channel in byte 2 of the header. When reading from the device you will see the channel number in byte 2 of the header.

- Channel 0: the SHTP control channel
- Channel 1: executable
- Channel 2: sensor hub control channel
- Channel 3: input sensor reports (non-wake, not gyroRV)
- Channel 4: wake input sensor reports (for sensors configured as wake up sensors)
- Channel 5: gyro rotation vector

Explanations of the first 4 channels is shown below –

The SHTP **control channel** provides information about the applications built into the BNO08X firmware image. The BNO08X uses advertisements to publish the channel maps and the names of the built-in applications.

The **executable channel** allows the host to reset the BNO08X and provide details of its operating mode

SHTP Channel	Use	Direction
1 (executable)	0 – reserved 1 – reset 2 – on 3 – sleep 4-255 – reserved	Write
	0 – reserved 1 – reset complete 2-255 – reserved	Read

The **sensor hub control channel** is used to configure the BNO08X. Responses from the BNO08X in reaction to configuration requests are also sent over this channel.

The **input sensor report channel** is unidirectional, passing data from the BNO08X to the host. All input reports pass over this channel except for sensors that are configured as wake sensors and the gyro rotation vector.

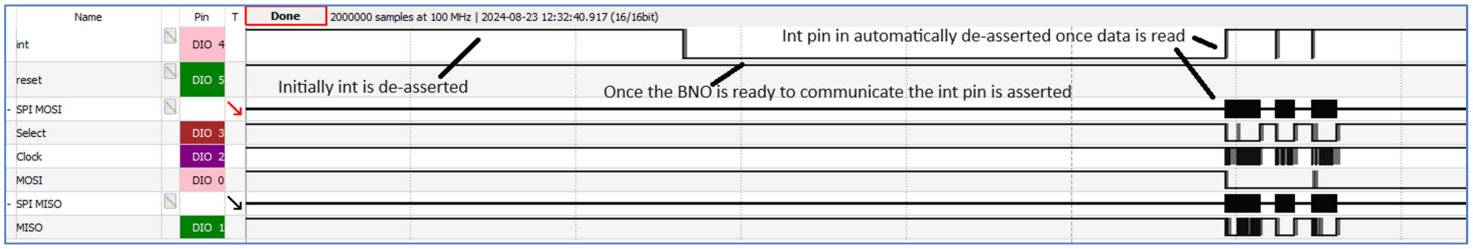
Startup

After power up or reset the BNO will assert INT once the device is ready to communicate.

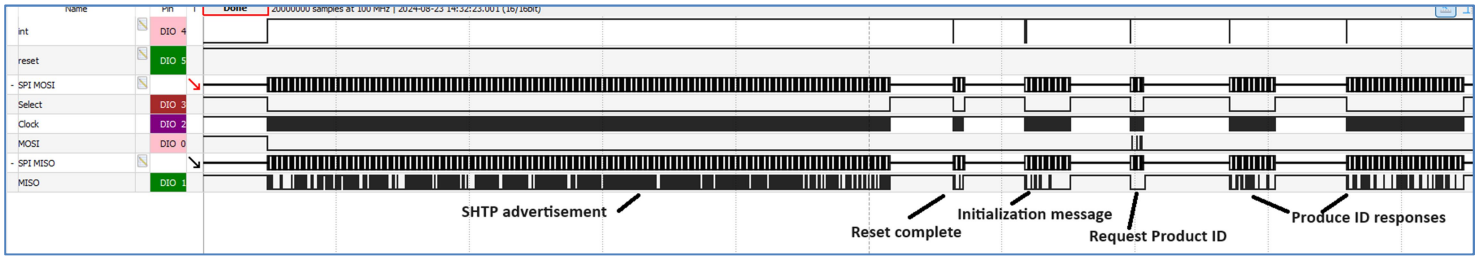
The BNO then sends 3 x unsolicited messages –

1. SHTP advertisement packet on channel 0
2. Reset complete message on channel 1
3. Initialization message on channel 2

Below you can see what happens after the BNO is powered up. Initially the INT pin is de-asserted (INT pin is active low), after some time the INT pin is asserted, SPI reads can now begin. Int pin is automatically de-asserted once the SHTP advertisement packet read starts.



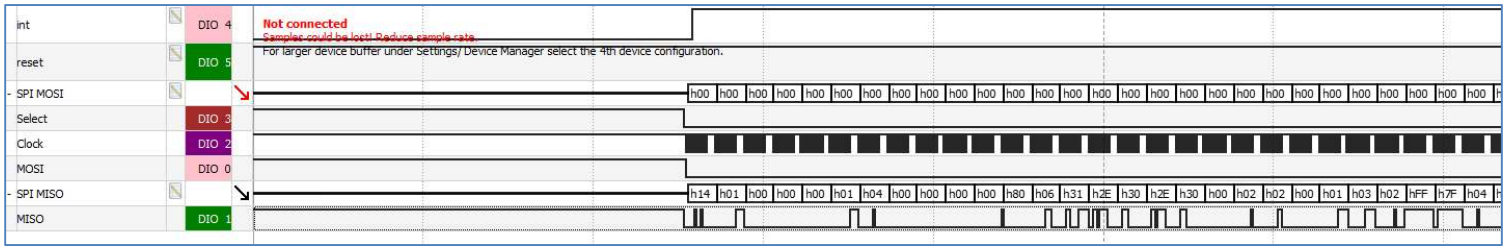
Below we can see the initial unsolicited payloads. Notice that each payload except for the SHTP advertisement is preceded by the INT pin being asserted by the BNO. Don't try reading any payload after the SHTP advertisement packet until you see the INT pin being pulled low.



SHTP advertisement

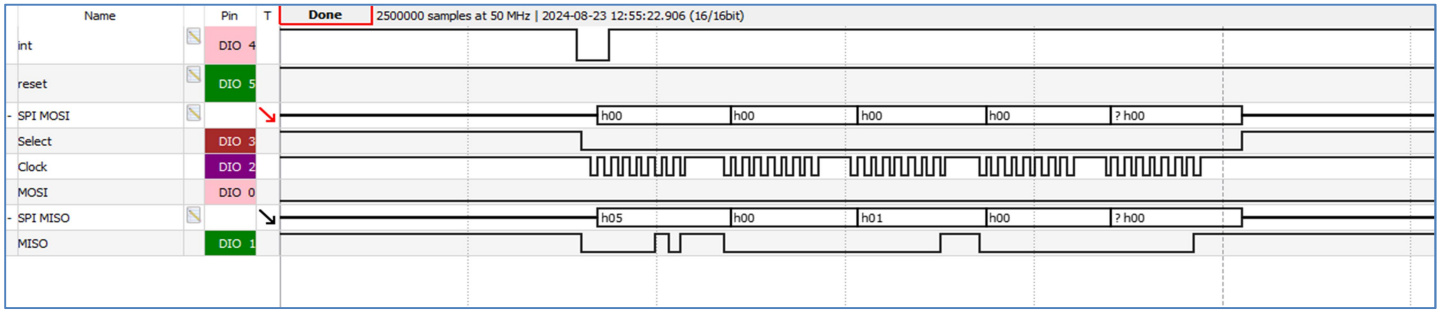
The first read from the BNO will then return the initial SHTP advertisement packet. This will be on channel zero (control channel). INT is de-asserted when the chip is selected for this read.
This read is long (LSB = 0x14 and MSB = 0x1) = 00000001 00001110 = 270 bytes including header.
The most significant bit is used to indicate if the data is part of a previous payload (the previous payload was not fully read. We can avoid this by making sure we look at the header and evaluate the total number of bytes that need to be read). In this case the MSB is a zero so we don't have to worry.

The parameters of the advertisement response use the tag-length-value format. You can find more info on this in the SHTP reference manual section 5.2.



Reset complete message

The 2nd read (started once the INT pin is again re-asserted) is a 5 byte reply on channel 1 (executable channel). The 5th byte has the value of 1 (showing a zero here for some reason but microcontroller is saying 1). From executable table this is reset complete.

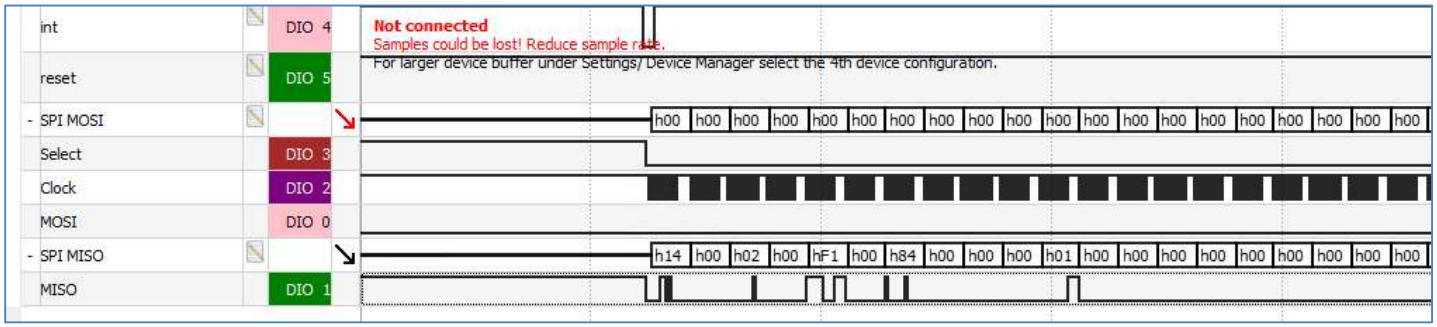


SHTP Channel	Use	Direction
1 (executable)	0 – reserved 1 – reset 2 – on 3 – sleep 4-255 – reserved	Write
	0 – reserved 1 – reset complete 2-255 – reserved	Read

Initialization message

The 3rd read (started once the INT pin is again re-asserted) is a 20 byte reply on channel 2 (sensor hub control channel). 0xF1 is the command response and is shown in the table below.
You can see that the unsolicited initialize command is being reported.

Status is successful on subsystem 1 (not sure what subsystem 1 is exactly).



6.4.5.2 Initialize Response

The sensor hub responds to the Initialize command with an Initialize Response. In the case where the sensor hub reinitializes itself, this response is unsolicited. An unsolicited response is also generated after startup.

Byte	Name	Description
0	Report ID	0xF1- Command Response
1	Sequence Number	See Section 6.3.9
2	Command	0x04 – Initialize, 0x84 – Initialize (unsolicited)
3	Command Sequence Number	See 6.3.9
4	Response Sequence Number	See 6.3.9
5	R0	Status (0 – successful. 1 – Operation failed)
6	R1	Subsystem
7	R2	Reserved
8	R3	Reserved
9	R4	Reserved
10	R5	Reserved
11	R6	Reserved
12	R7	Reserved
13	R8	Reserved
14	R9	Reserved
15	R10	Reserved

Figure 55: Initialize Response

Requesting and receiving a report

All the unsolicited start up data has now been read. We will now request the product ID. Before we can do this we need to wake the sensor up (as it goes to sleep after reading all the unsolicited messages). Check out the wake up section or the code for how to wake the sensor.

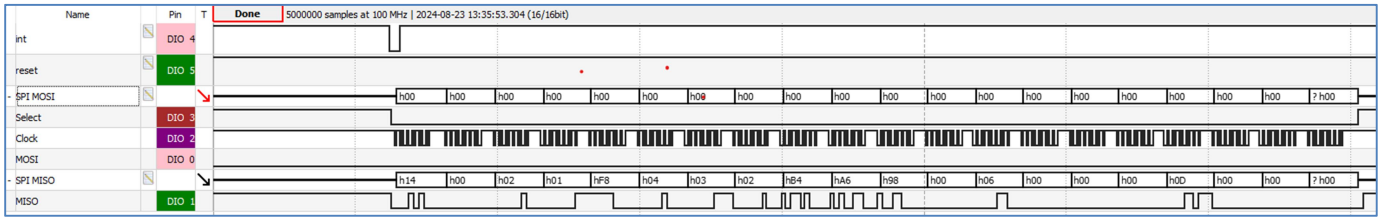
To request a report you have to write the header then the report number you want.
As per the previously mentioned header values, we have -

```
Product_ID[] = {6, 0, 2, 0, 0xF9, 0};
```

- 6 bytes of data to be transferred
- Channel 2
- Report 0xF9 (Product ID request)

1st product ID report

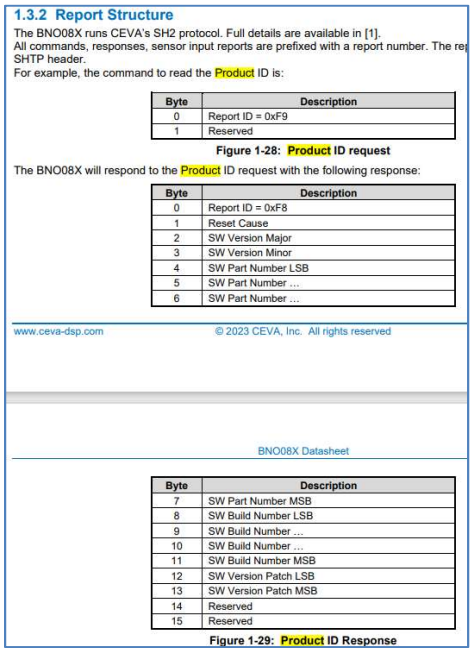
Once the data is written you again wait for the INT to de-assert then read the cargo. Here is the 20 byte response –



As can be seen we get a 0x14 byte long response from the sensor hub control channel. The report ID is 0xF8 (product ID response). The last reset was 0x04.

From the SH-2 ref manual -

0 – Not Applicable
1 – Power On Reset
2 – Internal System Reset
3 – Watchdog Timeout
4 – External Reset
5 – Other



1.3.2 Report Structure

The BNO08X runs CEVA's SH2 protocol. Full details are available in [1].
All commands, responses, sensor input reports are prefixed with a report number. The re
SHTP header.
For example, the command to read the Product ID is:

Byte	Description
0	Report ID = 0xF9
1	Reserved

Figure 1-28: Product ID request

The BNO08X will respond to the Product ID request with the following response:

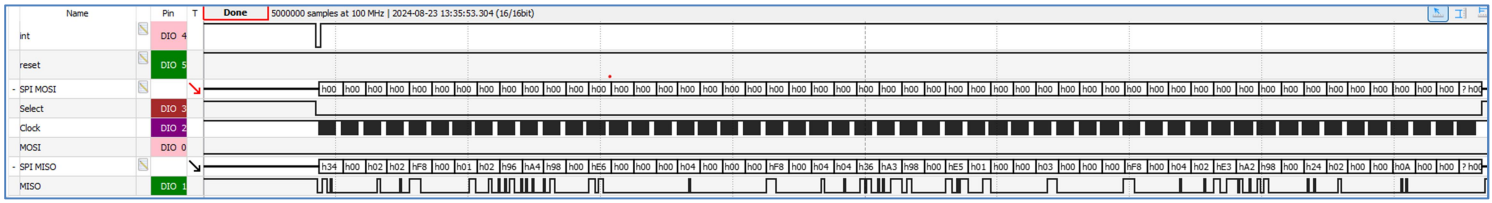
Byte	Description
0	Report ID = 0xF8
1	Reset Cause
2	SW Version Major
3	SW Version Minor
4	SW Part Number LSB
5	SW Part Number …
6	SW Part Number …

Byte	Description
7	SW Part Number MSB
8	SW Build Number LSB
9	SW Build Number …
10	SW Build Number …
11	SW Build Number MSB
12	SW Version Patch LSB
13	SW Version Patch MSB
14	Reserved
15	Reserved

Figure 1-29: Product ID Response

2nd product ID report

The BNO then sends another set of product ID's. I can't find what this is in the documentation but I am assuming it's the product ID responses for each device on the BNO board. There are 3 0xF8 data sets in this response. There are no more product ID responses after this 2nd product report.



Setting up a continuous sensor report

Here we tell the sensor to report the rotation vector at a set frequency. Make sure you read the Wake up section before proceeding as the sensor will be asleep at this stage. We need to wake it up.

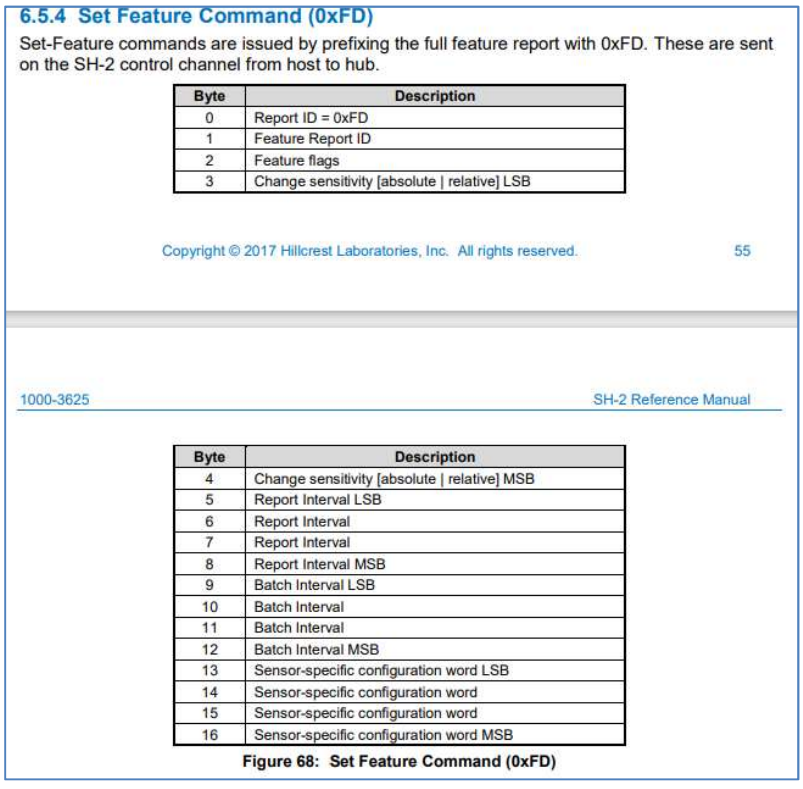
The rotation vector has a feature report ID of 0x05.
To set the report –

1. Write the 4 byte header
2. Write the data shown in the feature command table shown below

In this case we will set the sensor to report at 400hz. The report interval is in microseconds. So we need 2500 microseconds interval –
00000000 00000000 00001001 11000100
MSB LSB

We will be sending 21 bytes in total (4 header bytes and 17 feature command bytes).
The header and feature command bytes are -

```
static const uint8_t req_quat[] = {21, 0, 2, 0, 0xFD, 0x05, 0, 0, 0, 0b11000100, 0b00001001, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```



6.5.4 Set Feature Command (0xFD)

Set-Feature commands are issued by prefixing the full feature report with 0xFD. These are sent on the SH-2 control channel from host to hub.

Byte	Description
0	Report ID = 0xFD
1	Feature Report ID
2	Feature flags
3	Change sensitivity [absolute | relative] LSB

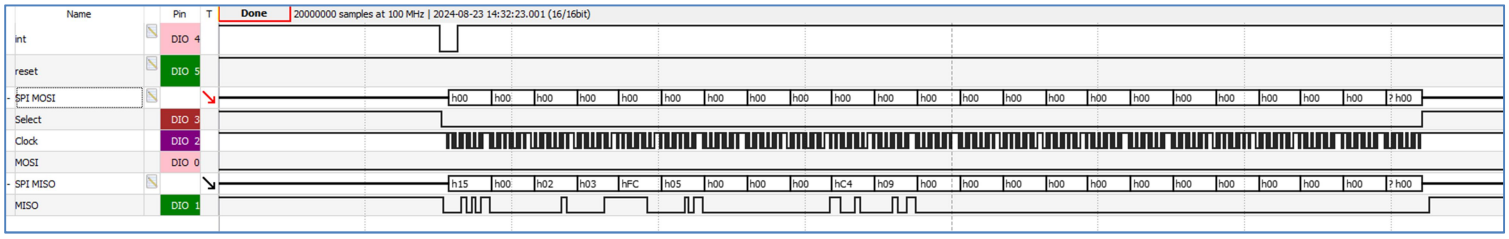
Byte	Description
4	Change sensitivity [absolute | relative] MSB
5	Report Interval LSB
6	Report Interval
7	Report Interval
8	Report Interval MSB
9	Batch Interval LSB
10	Batch Interval
11	Batch Interval
12	Batch Interval MSB
13	Sensor-specific configuration word LSB
14	Sensor-specific configuration word
15	Sensor-specific configuration word
16	Sensor-specific configuration word MSB

Figure 68: Set Feature Command (0xFD)

Get feature response

Once the INT pin is asserted we read the data. As you can see this is the Get Feature Response (0xFC).

It shows the feature report as 0x05 (rotation vector) with the report interval LSB as 0xC4 or 0b11000100. So far so good.



6.5.5 Get Feature Response (0xFC)

SH-2 will respond to Get Feature Requests (6.5.3) by sending a Get Feature Response on the control channel. These are the full feature report that was requested, prefixed with 0xFC.

Note that SH-2 protocol version 1.0.1 and higher will send Get Feature Response messages unsolicited if a sensor's rate changes (e.g. due to change in the rate of a related sensor).

Byte	Description
0	Report ID = 0xFC
1	Feature Report ID
2	Feature flags
3	Change sensitivity [absolute | relative] LSB
4	Change sensitivity [absolute | relative] MSB
5	Report Interval LSB
6	Report Interval
7	Report Interval
8	Report Interval MSB
9	Batch Interval LSB
10	Batch Interval
11	Batch Interval
12	Batch Interval MSB
13	Sensor-specific configuration word LSB
14	Sensor-specific configuration word
15	Sensor-specific configuration word
16	Sensor-specific configuration word MSB

Figure 69: Get Feature Response (0xFC)

Read sensor data

After the feature report we will start receiving the sensor data every 2500us. We must use an interrupt to read every time we see that INT falling edge.

The continuous report is different from the feature response report -

1. It comes in on channel 3 (input sensor reports)
2. It has a 5 byte base timestamp reference after the 4 byte header
3. The 14 byte rotation vector input report follows the base timestamp bytes

4 byte header + 5 byte timestamp + 14 byte input report = 23 or 0x17 bytes total.

7.2.1 Base Timestamp Reference (0xFB)

The Base Timestamp Reference provides a delta from the transport-protocol-defined reference point (the HINT assert for SHTP) and some arbitrary point in time.

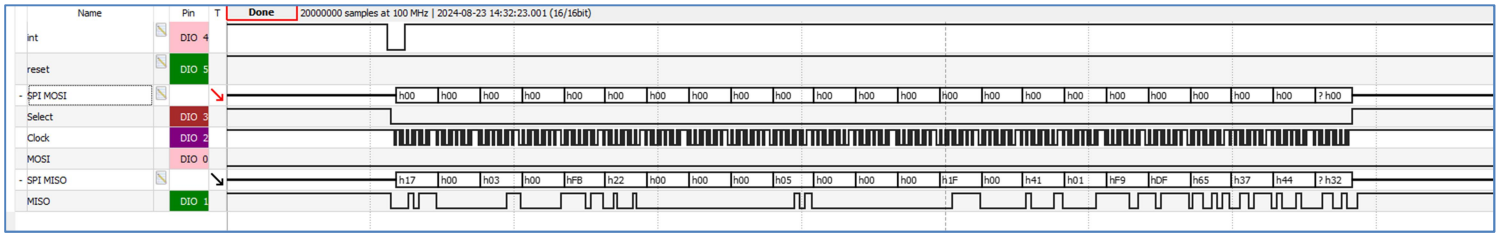
Byte	Description
0	Report ID=0xFB
1	Base Delta LSB: relative to transport-defined reference point. Signed. Units are 100 microsecond ticks.
2	Base Delta
3	Base Delta
4	Base Delta MSB

Figure 111: Base Timestamp Reference Record

6.5.18.2 Input Report

Byte	Description
0	Report ID = 0x05
1	Sequence number
2	Status
3	Delay
4	Unit quaternion i component LSB
5	Unit quaternion i component MSB
6	Unit quaternion j component LSB
7	Unit quaternion j component MSB
8	Unit quaternion k component LSB
9	Unit quaternion k component MSB
10	Unit quaternion real component LSB
11	Unit quaternion real component MSB
12	Accuracy estimate LSB
13	Accuracy estimate MSB

Figure 82: Rotation Vector Input Report



Code

To follow

Documentation

The issue here is that the information required to make this thing work is all over the place.
Read the SHTP document first.

Sensor Hub Transport protocol (best document to read first)
Sensor Hub Transport Protocol (sparkfun.com)

SH-2 SHTP Reference Manual
Microsoft Word - SH-2 SHTP Reference Manual.docx (ceva-ip.com)

SH-2 Reference Manual
SH-2 Reference Manual (sparkfun.com)

BNO08X Data Sheet
BNO08X Datasheet (ceva-ip.com)