6. After reset the PS0/WAKE signal is used as a 'wake' signal taking the BNO080 out of sleep if the host wants to initiate communication with the BNO080.
7. The BNO080 supports environmental sensors (e.g. pressure sensors, ambient light sensors) on a secondary I²C interface. This interface should be pulled up via resistors regardless of the presence of the external sensor as the SW polls for sensors at reset.

1.3.4.1 SPI Operation

SPI is a 4-wire synchronous serial protocol. SPI provides a full duplex communication path and has a master/slave relationship. The master provides the clock for all transactions. Multiple slave devices can exist on a SPI interface by the use of a chip select signal. The BNO080 is the slave in all communications.

SPI allows for two clock polarities and clock edge sampling. These options are typically called CPOL and CPHA. The BNO080 uses CPOL = 1 and CPHA = 1. In this configuration the clock idles high and data is captured on the rising edge of the clock:

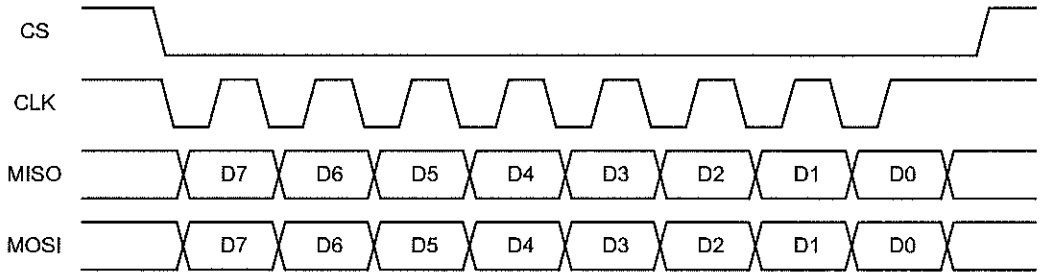

Figure 1-21: BNO080 SPI signaling

SPI transmits data MSB first and is byte oriented (i.e. all data is passed in 8-bit segments). Any number of bytes can be transferred in a single transaction (chip select assertion).

MISO is the data transferred from the slave to the master and MOSI from master to slave.

The BNO080 uses Hillcrest's SHTP protocol to communicate. More details are available at [2].

1.3.4.2 Wake operation

When the host want to initiate contact with the BNO080 it may be necessary to wake the processor from a sleep mode. To enable this function the BNO080 uses the PS0/wake pin. The pin is active low and should be driven low to initiate the wake procedure. The BNO080 will respond by asserting the interrupt line at which point the host can initiate SPI accesses. The BNO080 will de-assert the interrupt line as soon as the chip select is detected.

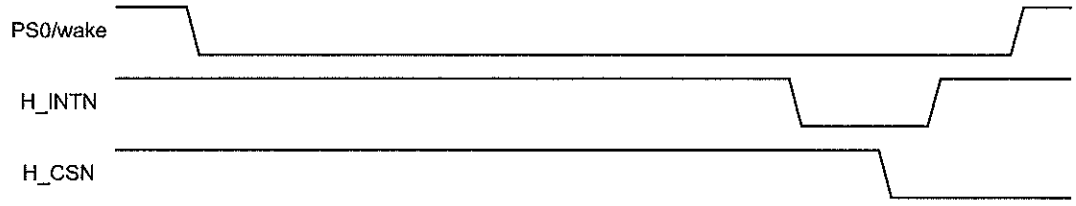

Figure 1-22: SPI Wake operation

1.3.5 UART-RVC interface

The BNO080 provides a simplified UART interface for use on robot vacuum cleaners (RVC). When configured in this mode the BNO080 simply transmits heading and sensor information at 100Hz over the UART TX pin. A typical connection is shown in Figure 1-23

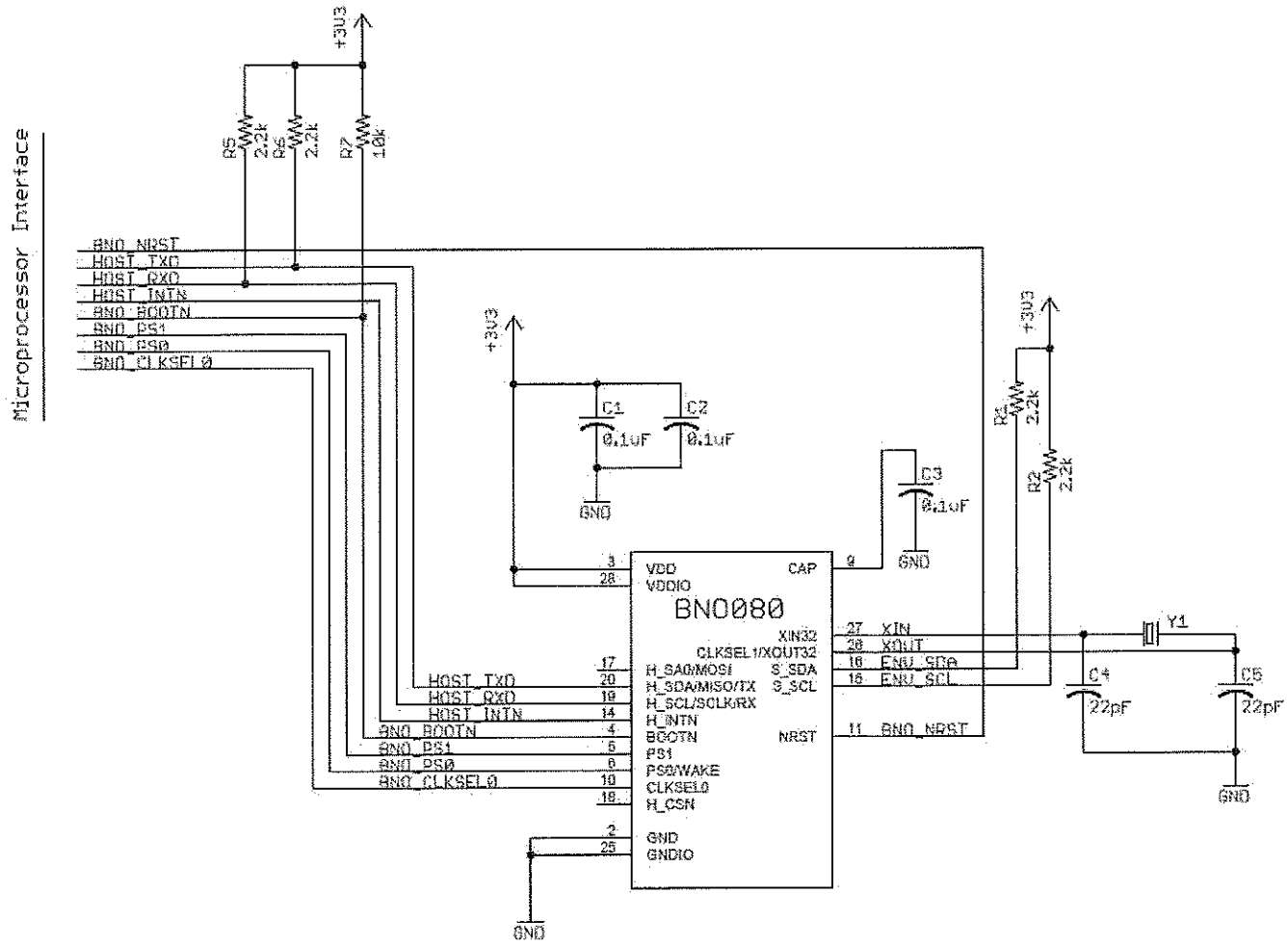

Figure 1-23: BNO080 UART-RVC connection diagram

Figure 1-23 shows how the BNO080 can be connected to an external microcontroller via a UART interface. The following notes are provided as guidelines for connecting the BNO080 in a system design.

1. NRST is the reset line for the BNO080 and can be either driven by the application processor or the board reset.
2. BOOTN is sampled at reset. If low the BNO080 will enter bootloader mode.
3. Pin 4 (BOOTN) should be pulled high through a 10K Ohms resistor. To use the device firmware update (DFU) capability of the BNO080, it is recommended to connect Pin 4 to a GPIO pin on the external microcontroller.
4. Pin 5 (PS1) and Pin 6 (PS0) are the host interface protocol selection pins. These pins should be tied to ground and VDDIO respectively to select the UART-RVC interface.
5. The BNO080 supports environmental sensors (e.g. pressure sensors, ambient light sensors) on a secondary I²C interface. This interface should be pulled up via resistors regardless of the presence of the external sensor as the SW polls for sensors at reset.

1.3.5.1 UART-RVC operation

The UART operates at 115200 b/s, 8 data bits, 1 stop bit and no parity. The UART protocol relies on an idle line being 'high'. A transmission is started with the assertion of a start bit (pulling the line low), followed by the data, LSB first. After the data segment is sent (in this case 8-bits), the line is pulled high (the stop signal) for a minimum number of bits (1 for the BNO080) to indicate end of that segment.

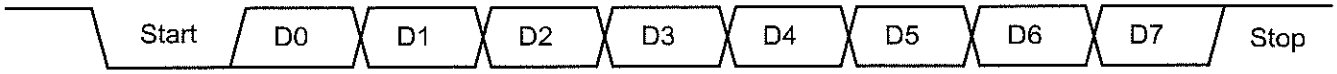

Figure 1-24: UART signaling

1.3.5.2 UART-RVC protocol

The BNO080 transmits the following data at a rate of 100Hz.

Header	Index	Yaw		Pitch		Roll		X-axis accel		Y-axis accel		Z-axis accel		Reserved			Csum
0xAAAA		LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	0	0	0	

Figure 1-25: BNO080 UART-RVC packet format

The 19-byte message has the following fields:

Header: Each report is prefixed with a 0xAAAA header
Index: A monotonically increasing 8-bit count is provided (0-255) per report
Yaw: The yaw is a measure of the rotation around the Z-axis since reset. The yaw has a range of +/- 180° and is provided in 0.01° increments, i.e. a report of 8734 is equivalent to 87.34°.
Pitch: The pitch is a measure of the rotation around the Y-axis. The pitch has a range of +/- 90° and is provided in 0.01° increments, i.e. a report of 1072 is equivalent to 10.72°.
Roll: The roll is a measure of the rotation around the X-axis. The roll has a range of +/- 180° and is provided in 0.01° increments, i.e. a report of 1072 is equivalent to 10.72°.
X-axis acceleration: The acceleration along the X-axis, presented in mg
Y-axis acceleration: The acceleration along the Y-axis, presented in mg
Z-axis acceleration: The acceleration along the Z-axis, presented in mg
Reserved: The message is terminated with three reserved bytes, currently set to zero
Checksum (Csum): The Index, yaw, pitch, roll, acceleration and reserved data bytes are added to produce the checksum.

To determine the actual orientation of the module, the rotations should be applied in the order yaw, pitch then roll.

An example complete message and checksum calculation is as follows:

Message: 0xAA AA DE 01 00 92 FF 25 08 8D FE EC FF D1 03 00 00 00 E7

Where:

Index = 0xDE = 222
Yaw = 00.01° (1 = 0x0001)
Pitch = -1.10° (-110 = 0xFF92)
Roll = 20.85° (2085 = 0x0825)
X-acceleration = -371 mg = -3.638 m/s² (-371 = 0xFE8D)
Y-acceleration = -20 mg = -0.196 m/s² (-20 = 0xFFEC)
Z-acceleration = 977 mg = -9.581 m/s² (977 = 0x03D1)
Checksum = 0xE7

1.4 Host Communication

1.4.1 SHTP

The BNO080 uses Hillcrest's SHTP (Sensor Hub Transport Protocol) to communicate for all interface styles except UART-RVC. SHTP provides a means of passing data between the BNO080 and a host with support for multiple channels. The BNO080 does not currently support the inclusion of 3rd party applications, but the SHTP protocol allows for separation of traffic via these channels such that applications on a host can communicate over this channel.

All data is prefixed with a 4-byte header:

