3 Synchronous Serial Interface

3.1 Description

This section details the synchronous serial interface between ANT and a Host MCU. This mode is selected by connecting the PORTSEL input high, and may be used with a hardware SPI port.

Please refer to Section 2 for details on the alternative asynchronous mode.

Note, when operating in synchronous mode careful attention to reset behavior is required to prevent inadvertent deadlock conditions between ANT and the Host MCU. Please see Section 3.4 for more details on this subject.

In synchronous mode, ANT uses a half-duplex synchronous master serial interface with message flow control. The Host must be configured as a synchronous slave. The interface is meant to accommodate either a hardware synchronous slave port or a simple I/O control on the Host MCU. Full flow control is maintained in both directions, therefore the Host MCU retains full control of the message flow and can halt incoming messages as required.

3.2 Port Parameters

The synchronous serial interface between ANT and the Host MCU is shown below.

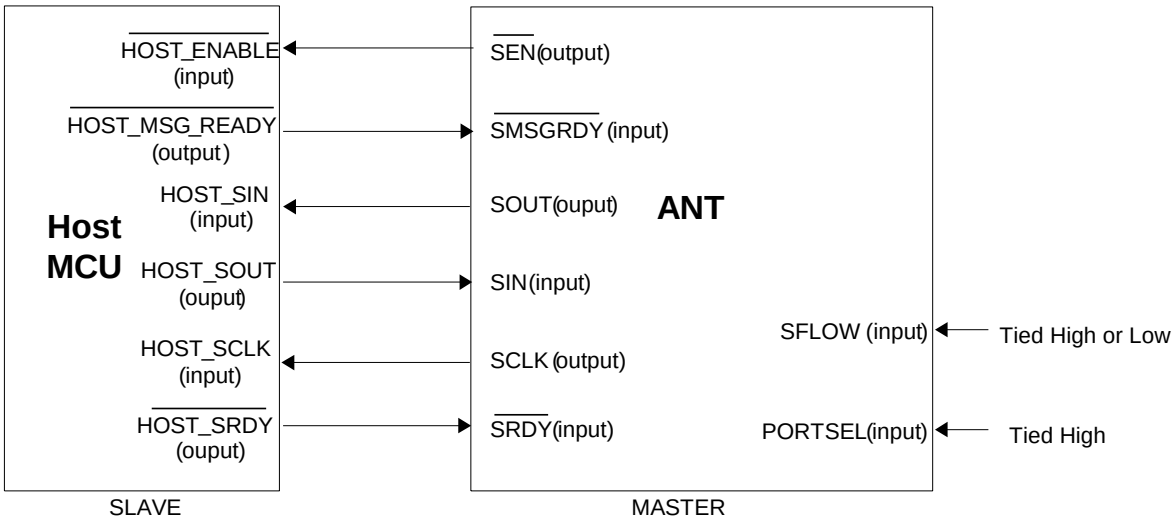

Figure 3-1. Synchronous Mode Connections

3.2.1 Port Select (PORTSEL)

The PORTSEL signal should be connected to logic high for synchronous serial mode.

3.2.2 Flow Control Select (SFLOW)

The Flow Control Select signal is used to configure the synchronous serial port for either Byte or Bit flow control.

SFLOW	Flow Control
0	Byte Flow Control
1	Bit Flow Control

Please note that Byte flow control assumes that the Host contains synchronous communication hardware which can be configured for synchronous slave communication. Bit flow control can be used when all serial lines are implemented in software on the Host MCU. The differences between byte and bit flow control are detailed in the sections below.

3.3 Link Layer Protocol

3.3.1 Characteristics

The ANT interface protocol has the following characteristics:

- Binary protocol
- Packets of variable length
- Each packet contains an 8-bit Checksum
- Data is transmitted LSB first

3.3.2 Message Structure

ANT and the Host MCU communicate by transmitting messages to each other. Each message is formatted as shown below.

SYNC R / W	MSG LENGTH	MSG ID	DATA_1	DATA_2	.....	DATA_N	CHECKSUM

3.3.3 Message Details

Byte #	Bit #	Description	Length	Description
0	7:1	SYNC	7 bits	Fixed SYNC field = 1010010 (MSB:LSB)
0	0	R/W	1 bit	0 : Write (Message ANT → Host) 1 : Read (Message Host → ANT)
1	-	LENGTH	1 Byte	Number of data byes in the message (Length should be between 1 and 9)
2	-	ID	1 Byte	Data type identifier 0 : Invalid 1..255 : Valid data type ID
3..N+2	-	DATA_1 ... DATA_N	N Bytes	Message data bytes (There may be between 1 and 9 data bytes)
N+3	-	CHECKSUM	1 Byte	XOR of all previous bytes (**including** SYNC)

The following is an example of how to encode a message to send from the Host to ANT.

ANT_OpenChannel(1)

← SerialData (0xA5) // 0xA5 is read indicating that the Host may send a message to ANT

SerialData(0x01, 0x4B, 0x01, 0xEE) // The Host can then send the 4-byte message to ANT

Byte #	Name	Length	Direction	Data	Description
0	SYNC	1 Byte	ANT->Host	0xA5	SYNC is 0xA5 for a Host->ANT transaction
1	LENGTH	1 Byte	Host->ANT	0x01	Number of data bytes in this message = 1
2	ID	1 Byte	Host->ANT	0x4B	ANT_OpenChannel message ID is 0x4B
3	DATA_1	1 Byte	Host->ANT	0x01	There is 1 Data Byte in this message: This byte is Channel #. It has been set to Channel = 1
4	CHECKSUM	1 Byte	Host->ANT	0xEE	0xA5 xor 0x01 xor 0x4B xor 0x01 = 0xEE

The following is an example of how the Host would decode a message received from ANT.

← SerialData (0xA4, 0x02, 0x52, 0x01, 0x03, 0xF6) // The Host receives 6-byte message

← Channel_Status(1, 3) // Decodes into a channel status message

Byte #	Name	Length	Direction	Data	Description
0	SYNC	1 Byte	ANT->Host	0xA4	SYNC is 0xA4 for an ANT->Host transaction
1	LENGTH	1 Byte	ANT->Host	0x02	Number of data bytes in this message = 2
2	ID	1 Byte	ANT->Host	0x52	Channel_Status Message is 0x52
3	DATA_1	1 Byte	ANT->Host	0x01	There are 2 data bytes in this message: This byte is Channel #. Channel = 1
4	DATA_2	1 Byte	ANT->Host	0x03	This byte is the status. Status = 3, which indicates the channel is tracking.
5	CHECKSUM	1 Byte	ANT->Host	0xF6	0xA4 xor 0x02 xor 0x52 xor 0x01 xor 0x03 = 0xF6

3.4 Synchronization and Reset

In order for the Host MCU to guarantee synchronization with ANT following a power up condition, it must reset ANT immediately after power-up or after any condition that results in loss of synchronization. The reset method depends on the specific ANT chipset or module being used, and the availability of a reset pin.

There are several methods available to reset the ANT chip. For all reset methods, the configuration is cleared and goes back to default values, all channels are closed (and unassigned). In synchronous mode the power state transitions are built into the serial protocol, therefore the SLEEP and SUSPEND reset options are not available.

3.4.1 Resetting with the Reset Pin

If available, asserting the $\overline{\text{RESET}}$ signal on an ANT chip or module is the preferred reset method. A reset via this pin will have the same effect as a synchronization sequence. The electrical characteristics of the reset pin are detailed in the datasheet of the specific ANT chipset or module. Please note that not all ANT chips or modules have a reset pin available. If a reset pin is not available, then the Host MCU must apply a synchronous reset as detailed in the section below. While the reset pin is a simpler and quicker way of resetting an ANT part, care should be taken to ensure product will be compatible with all chips or chipsets that may be designed in.

3.4.2 Resetting with Synchronous Reset Sequence

For ANT chips and modules that do not have a reset pin, a specific sequence known as the synchronous reset may be applied to ANT (Figure 3-2) instead. This synchronous reset is applicable to synchronous mode communication only, however it is supported by all ANT chipsets and modules (including those that support a reset pin).

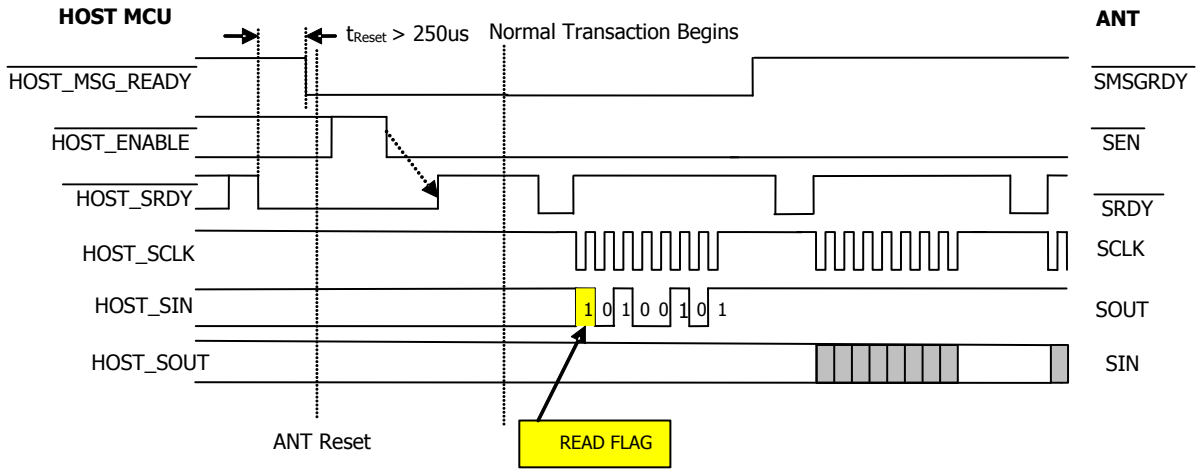

Figure 3-2. Synchronization with ANT upon start-up

3.4.3 Resetting with the Software Reset

Once communication is established, the reset command (0x4A, ANT_ResetSystem) can be sent via the serial interface to reset the ANT chip. The synchronization sequence does not need to be reapplied after the software reset. The major difference between the pin reset and the software reset is that the reset pin is slightly more thorough, resetting all I/O input states. Software reset is also only valid after communication is synchronized, and is not applicable in a power up state.

3.5 Operating Mechanism

A basic description of the communications mechanism follows.

- The synchronous serial port provided by ANT is a half duplex synchronous master, with full flow control in both directions of communication.
- Flow control of data transmitted to the Host MCU is controlled by the $\overline{\text{SRDY}}$ signal, and flow control of data transmitted to ANT is controlled by the master SCLK signal.
- By default, the Host is in receive mode and ANT is in transmit mode. In this state, ANT will forward all incoming radio messages to the Host as they become available. ANT uses the $\overline{\text{SEN}}$ signal to indicate the start of a message transaction and the Host will use the $\overline{\text{SRDY}}$ flow control to signal its readiness for incoming messages.

- $\overline{\text{SRDY}}$ must be asserted for communication to begin.

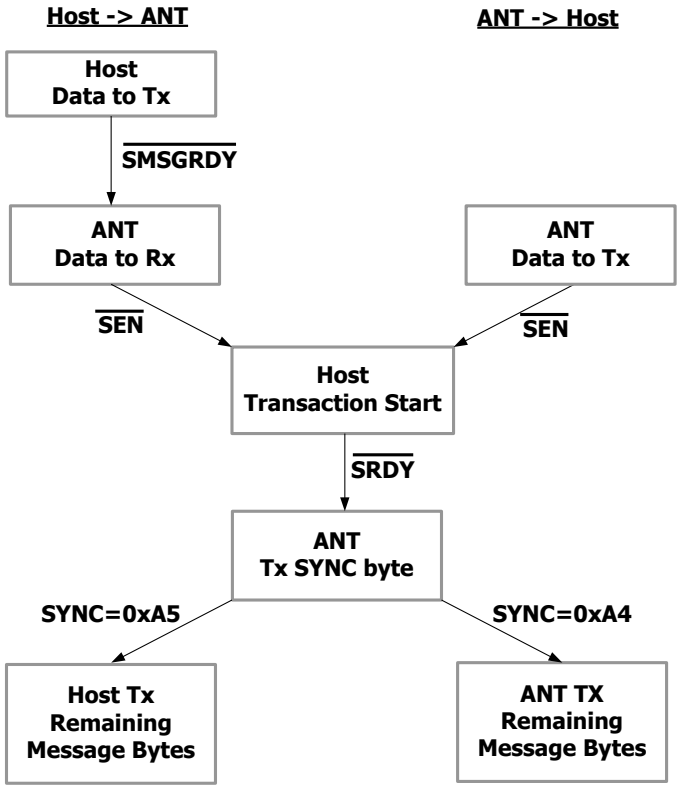

Figure 3-3. Synchronous Serial Communication

- Figure 3-3 and the timing diagrams in Figure 3-4 through Figure 3-7 for the basic message transaction sequence:
  - For a message from Host->ANT:
    1. The Host will assert the $\overline{\text{SMSGRDY}}$ signal indicating it wishes to enter into transmit mode.
  - In **either** receive or transmit mode:
    2. ANT will assert $\overline{\text{SEN}}$ to indicate the start of a message transfer
    3. After $\overline{\text{SEN}}$ has been asserted, the Host will assert $\overline{\text{SRDY}}$ to indicate it is ready for communication
    4. After $\overline{\text{SEN}}$ and $\overline{\text{SRDY}}$ are both asserted, ANT **always** transmits the first (i.e. SYNC) byte. This is output from SOUT, and clocked with SCLK (see Section on Electrical Specifications for details of clock frequency). The lsb of the SYNC byte indicates the direction of the remaining message bytes (0 : Message Receive, ANT → Host; 1: Message Transmit, Host → ANT)
    5. If the SYNC byte indicates a message receive (ANT->Host), the additional message bytes will be transmitted the same way as the SYNC byte.
    6. If the SYNC byte indicates a message transmit (Host->ANT), the Host must output its data to ANT SIN at the clock rate provided by ANT SCLK.
  - Data is transmitted least-significant-bit (lsb) first.

3.6 Synchronous Messaging with Byte Flow Control

Byte flow-control mode is used when a synchronous hardware serial port is available.

The Host MCU flow-control signal $\overline{\text{SRDY}}$ can be implemented with a software controlled IO line or, in some cases, may be controlled by the Host's hardware serial port (e.g. EPSON MCU USART support for $\overline{\text{SRDY}}$).

Data bits change state on the falling edge of SCLK and are read on the rising edge of SCLK. This is true for transactions in either direction.

The first byte (i.e. SYNC byte) in the transaction sequence is always sent from ANT to the Host MCU. The first bit of the SYNC byte dictates the direction for the remaining bytes in the transaction.

Shown below in Figure 3-4 to Figure 3-7 are examples of transactions between the Host and ANT in byte synchronous mode, using hardware or software $\overline{\text{SRDY}}$ respectively.

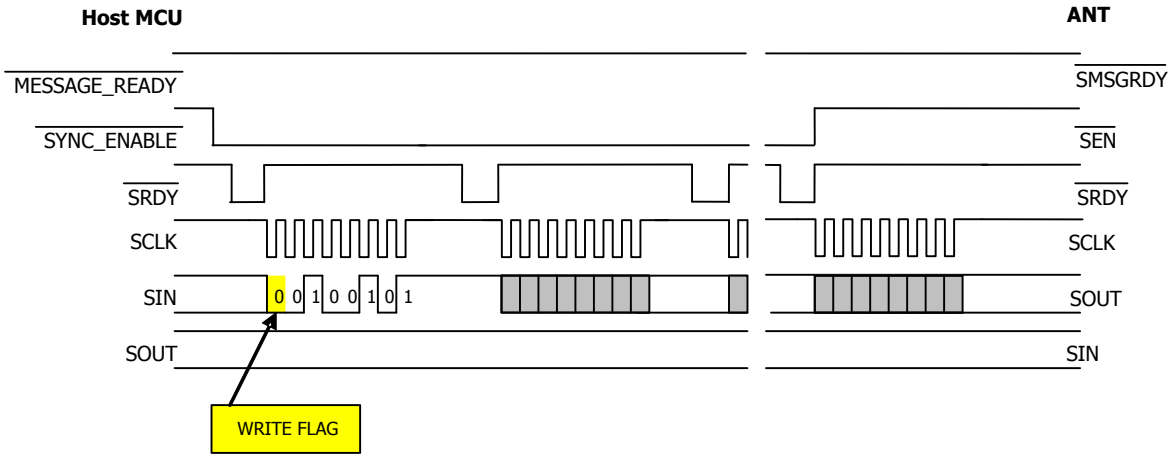

Figure 3-4. ANT -> Host Transaction (Hardware $\overline{\text{SRDY}}$)

For ANT to host transactions with hardware $\overline{\text{SRDY}}$ (Figure 3-4), ANT asserts $\overline{\text{SEN}}$ and waits for the host to assert $\overline{\text{SRDY}}$. Once both $\overline{\text{SEN}}$ and $\overline{\text{SRDY}}$ have been asserted, ANT will the send the SYNC byte from SOUT. For hardware $\overline{\text{SRDY}}$, this signal will be de-asserted on the first SCLK transition. The first bit of the SYNC byte will notify the host of the message direction (i.e. ANT -> host), and the host will once again assert $\overline{\text{SRDY}}$ to receive the next message byte from ANT. Again, the hardware $\overline{\text{SRDY}}$ will de-assert on the first SCLK transition and re-assert after each byte until the entire message has been transferred. After the last message byte, $\overline{\text{SRDY}}$ will remain de-asserted until the next message transaction is requested.

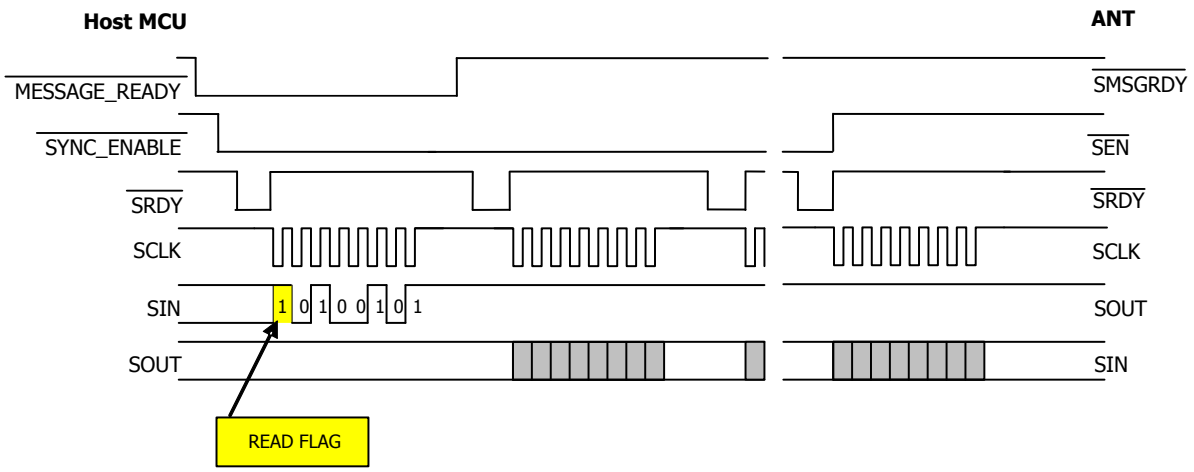

Figure 3-5. Host -> ANT Transaction (Hardware $\overline{\text{SRDY}}$)

For host to ANT transactions with hardware $\overline{\text{SRDY}}$ (Figure 3-5), the process is very similar. The main difference is that the host first asserts $\overline{\text{SMSGRDY}}$ to inform ANT that it wished to send a message. ANT will respond by asserting $\overline{\text{SEN}}$ and then waiting for the host to assert $\overline{\text{SRDY}}$. Once both $\overline{\text{SEN}}$ and have been asserted, ANT will the send the SYNC byte. For hardware $\overline{\text{SRDY}}$, this signal will be de-asserted on the first SCLK transition. The first bit of the SYNC byte will notify the host of the message direction (i.e. host->), and the host will once again assert $\overline{\text{SRDY}}$ and then send the next message byte to ANT on host SOUT at the rate of SCLK. Again, the hardware $\overline{\text{SRDY}}$ will de-assert on the first SCLK transition and re-assert after each byte until the entire message has been transferred. After the last message byte, $\overline{\text{SRDY}}$ will remain de-asserted until the next message transaction is requested.

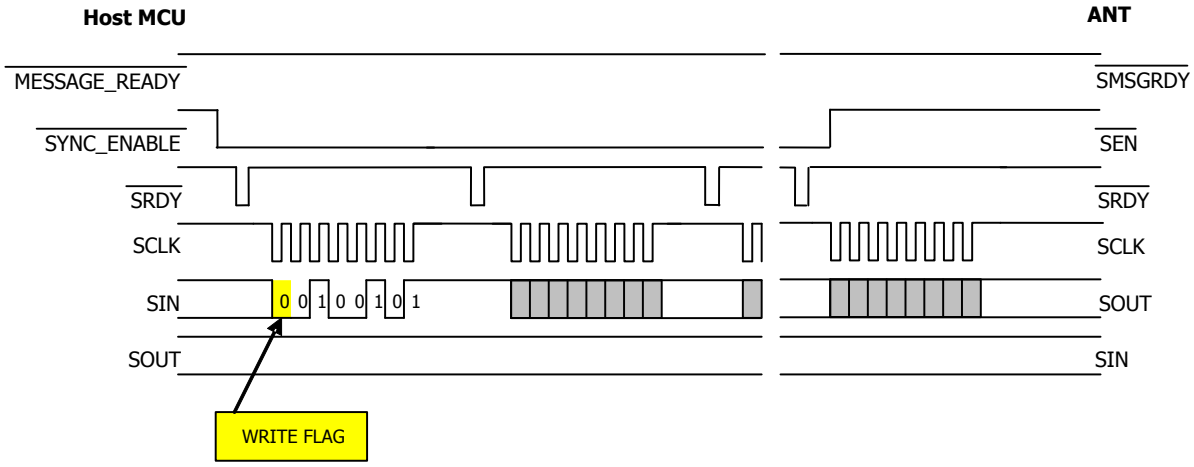

Figure 3-6. ANT -> Host Transaction (Software $\overline{\text{SRDY}}$)

The process for ANT to host transactions with software $\overline{SRDY}$ (Figure 3-6) is very similar as for hardware $\overline{SRDY}$. The sole difference is that the host can just pulse $\overline{SRDY}$ and does not have to wait until the first SCLK transition.

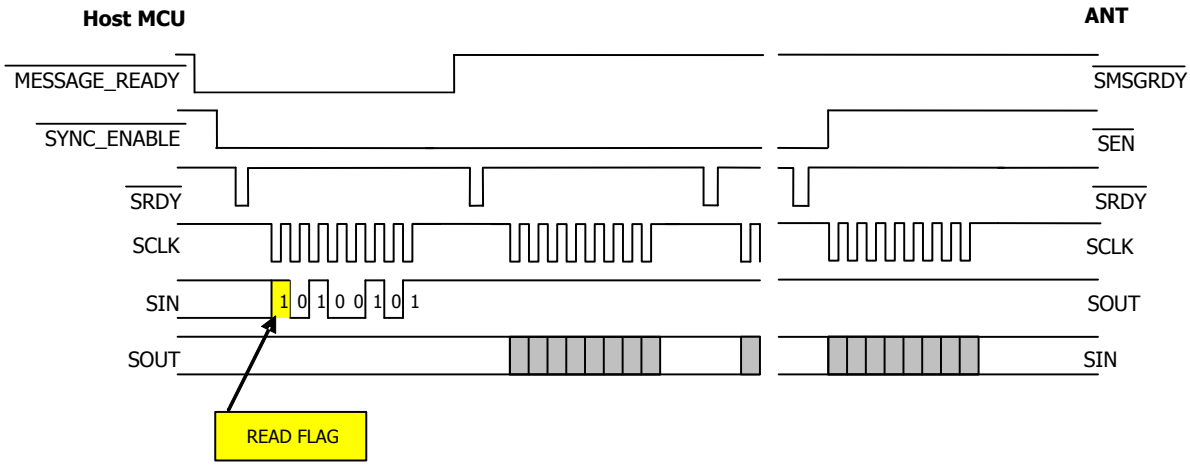

Figure 3-7. Host -> ANT Transaction (Software $\overline{SRDY}$)

The process for host to ANT transactions with software $\overline{SRDY}$ (Figure 3-7) is very similar as for hardware $\overline{SRDY}$. The only difference is that the host can pulse $\overline{SRDY}$ and does not have to wait until the first SCLK transition.

3.7 Synchronous Messaging with Bit Flow Control

If no SPI unit is available on the Host MCU, ANT can still be controlled using bit flow control. Using this method, the serial lines are implemented with software controlled IO lines. All of the signaling at the message transaction level remains the same as above; however, instead of pulsing after every byte, $\overline{SRDY}$ is pulsed for each bit of the message as shown below in .

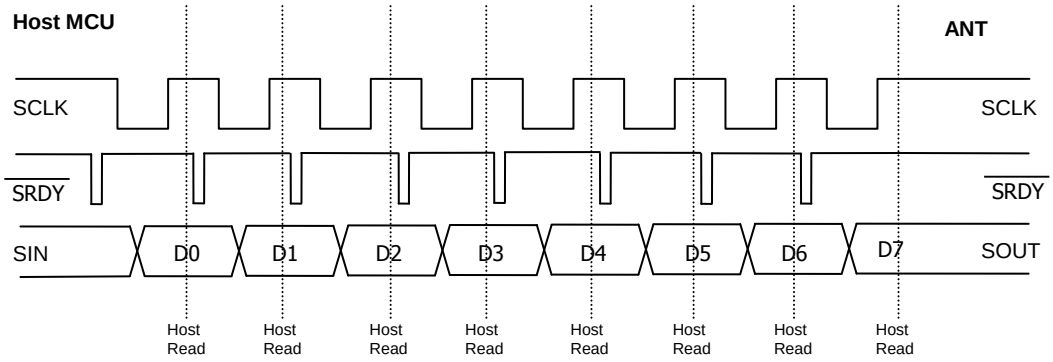

Figure 3-8. ANT -> Host Byte Transaction (Software Bit Flow Control)

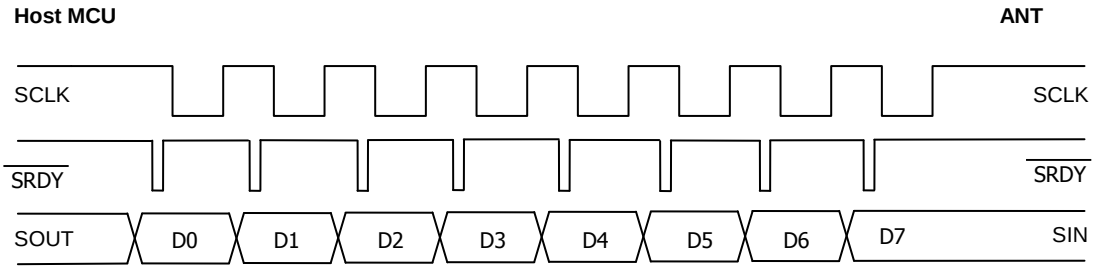

Figure 3-9. Host -> ANT Transaction (Software Bit Flow Control)

It is important to note that the Host MCU will do all bit processing on the rising edge of the SCLK signal, with the exception being when the byte is transmitted from Host MCU to ANT, and the first data bit will need to be asserted prior to the first clock edge. The final rising edge of the byte transaction will be the event to drive byte processing.

3.8 Power Up / Power Down

ANT will automatically place itself into deep sleep mode when all radio channels are closed and there is no activity on the $\overline{\text{SMSGRDY}}$ input signal. The Host MCU should ensure these conditions during times that the ANT radio is not required in order to maximize product battery life.

Upon every power up, the host must apply the Synchronous Reset sequence as described in Section 3.4.

3.9 Serial Enable Control (ANT → Host)

The $\overline{\text{SEN}}$ signal, which is driven by ANT, will be asserted prior to message transmission. It can therefore be used as a serial port enable signal, which is useful in cases where the Host serial port requires hardware activation (Figure 3-10). The $\overline{\text{SEN}}$ signal will be de-asserted by ANT on the first SCLK pulse of the last message byte.

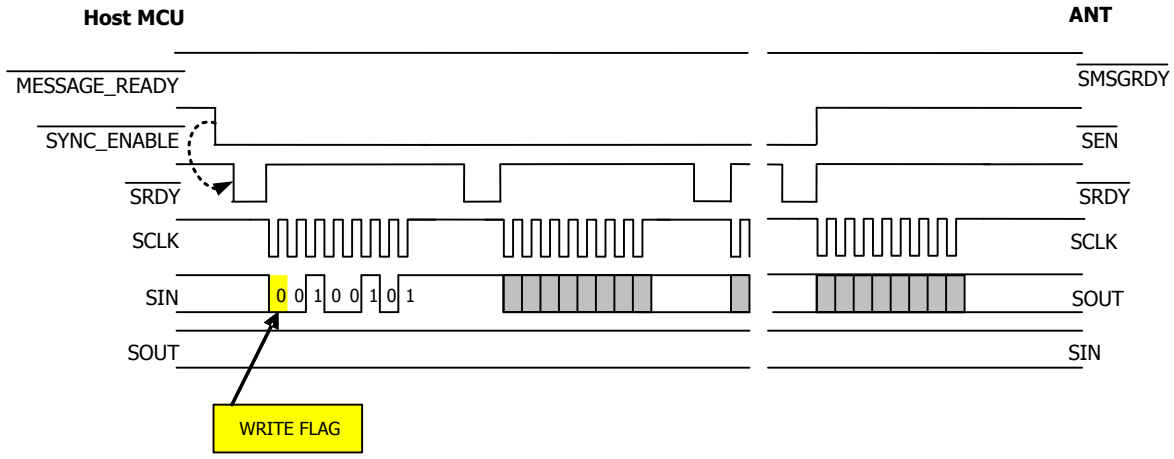

Figure 3-10. Serial Enable Control using ANT

3.10 Using an Epson MCU as a Host controller

The interface has been designed to easily communicate with an EPSON microcontroller with a built-in USART. The EPSON should be configured in the following manner:

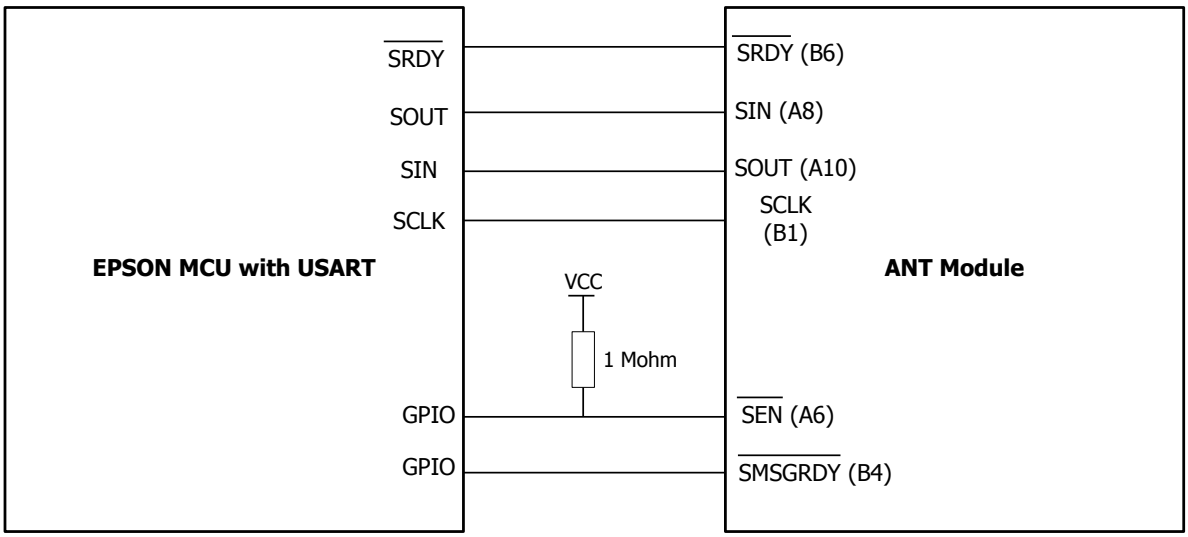

Figure 3-11. Example EPSON configuration for Byte Synchronous serial interface

For proper implementation of the above setup:

1. The GPIO connected to $\overline{SEN}$ must be configured as input.
2. The GPIO connected to $\overline{SMSGRDY}$ must be configured as output.
3. The EPSON USART must be configured as a synchronous slave.
4. Configure the $\overline{SRDY}$ pin to be controlled by the USART. Note that the register flag that causes the $\overline{SRDY}$ pin to go low while waiting for a new byte must not be cleared until the $\overline{SEN}$ signal is seen to go low from the ANT device. This is to avoid causing a synchronization condition as mentioned in Section 3.4 above.

With the above setup, the EPSON MCU hardware USART will control the signaling on the $\overline{SRDY}$, SIN, SOUT and SCLK pins while the firmware (on the MCU) will handle signals on GPIOs connected to $\overline{SEN}$ and $\overline{SMSGRDY}$.