Interfacing with ANT General Purpose Chipsets and Modules

Copyright Information and Usage Notice

TABLE OF CONTENTS

1 Introduction

ANT is a 2.4GHz bidirectional wireless sensor network communications technology optimized for transferring low-datarate, low-latency data between multiple ANT-enabled devices. The ultra-low power consumption of ANT guarantees an extended battery life even from low-capacity supplies such as a coin cell battery, enabling use in heart rate monitors, bicycle computers and wrist watches. ANT's small size and low implementation cost allow effortless integration into the tiny form factor of wrist watches, PDAs and mobile phones.

ANT provides carefree handling of the Physical, Network and Transport OSI layers. In addition, it incorporates key low-level security features that form the foundation for user-defined sophisticated network security implementations. ANT ensures adequate user control while considerably lightening computational burden in providing a simple yet effective wireless networking solution.

Application / Presentation Layers	} User Defined
Higher Level Security	
Network / Transport & Low Level Security	
Data Link Layer	} Implemented by ANT
Physical Layer	
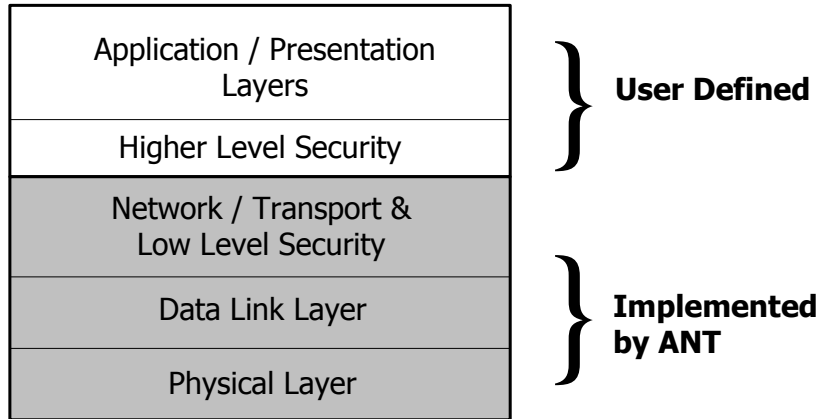
The interface between ANT and the Host application has been designed with the utmost simplicity in mind so that ANT can be easily and quickly implemented into new devices and applications. The encapsulation of wireless protocol complexity within the ANT chipsets vastly reduces the burden on the application host controller, allowing a low-cost 4-bit or 8-bit microcontroller to establish and maintain complex wireless networks. Data transfers can be scheduled in a deterministic or ad-hoc fashion. A burst mode allows for the efficient transfer of large amounts of stored data to and from a PC or other computing device. ANT aggressively balances functionality, cost, size and power consumption within the constraints of a wireless sensor network. Typical applications include sensor integration, tagging systems, remote monitoring etc.

The intent of this document is to detail the interface requirements between an application microcontroller and the ANT products. It provides insight into both interface signals and physical layer data formats.

A complete description of the ANT Message protocol is found in ANT Message Protocol and Usage document.

2   Asynchronous Serial Interface

2.1   Description

The Host MCU and ANT may communicate using the asynchronous mode of the serial interface. The connection diagram is shown below in Figure 2-1. Asynchronous mode is selected by the PORTSEL input being tied low.

Refer to Section 3 for details on the alternative synchronous mode.

2.2   Interconnect

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

* Not available on all ANT devices
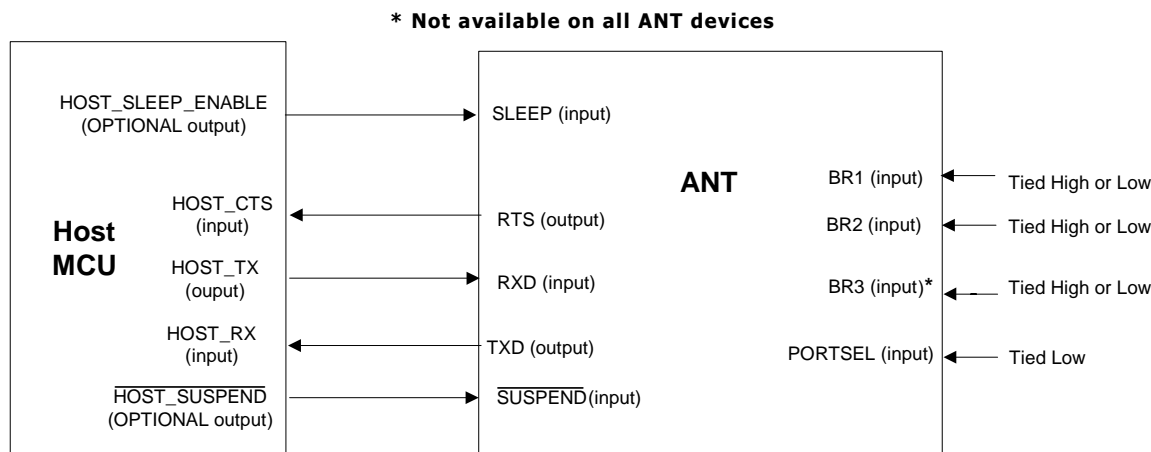
Figure 2-1: Asynchronous Mode Connections

Please note that all UART communication settings are for one start bit, one stop bit, 8 bits of data and no parity. Data is sent and received LSBit first.

2.3   Port Select (PORTSEL)

The PORTSEL signal should be tied low for asynchronous serial mode.

2.4   Speed Select (BR1, BR2, BR3)

The baud rate of the asynchronous communication between the Host and ANT is controlled by the speed select signals BR1, BR2 and BR3. Not all of these inputs are available on all ANT products. Please refer to the datasheets for products of interest for more information.

The table below shows the relationship between the states of the speed select signals and the corresponding baud rates.

BR3*	BR2	BR1	Baud Rate
0	0	0	4800
0	1	0	19200
0	0	1	38400
0	1	1	50000
1	0	0	1200
1	1	0	2400
1	0	1	9600
1	1	1	57600

* Not available on all ANT devices. It is assumed to be of value 0 when not available.

Note that baud rate may have a significant impact on system current consumption. Refer to the Electrical Specifications section of the ANT product of interest for current consumption figures.

2.5   SUSPEND mode control ($\overline{\text{SUSPEND}}$)

The assertion of the $\overline{\text{SUSPEND}}$ signal will cause ANT to terminate all RF and serial port activity and power down. This will happen immediately, regardless of the state of the ANT system. This signal provides support for use in USB applications, where USB devices are required to quickly enter a low-power state through hardware control.

Entering and exiting from the $\overline{\text{SUSPEND}}$ mode require the use of the SLEEP signal, in addition to the $\overline{\text{SUSPEND}}$ signal. The assertion of $\overline{\text{SUSPEND}}$ is only recognized if SLEEP is also asserted at the time.

De-assertion of the SLEEP signal is the only method for exiting from $\overline{\text{SUSPEND}}$ mode, as shown in Figure 2-2. Following exit, all previous transactions and configurations will be lost – ANT will be in its power-up state.
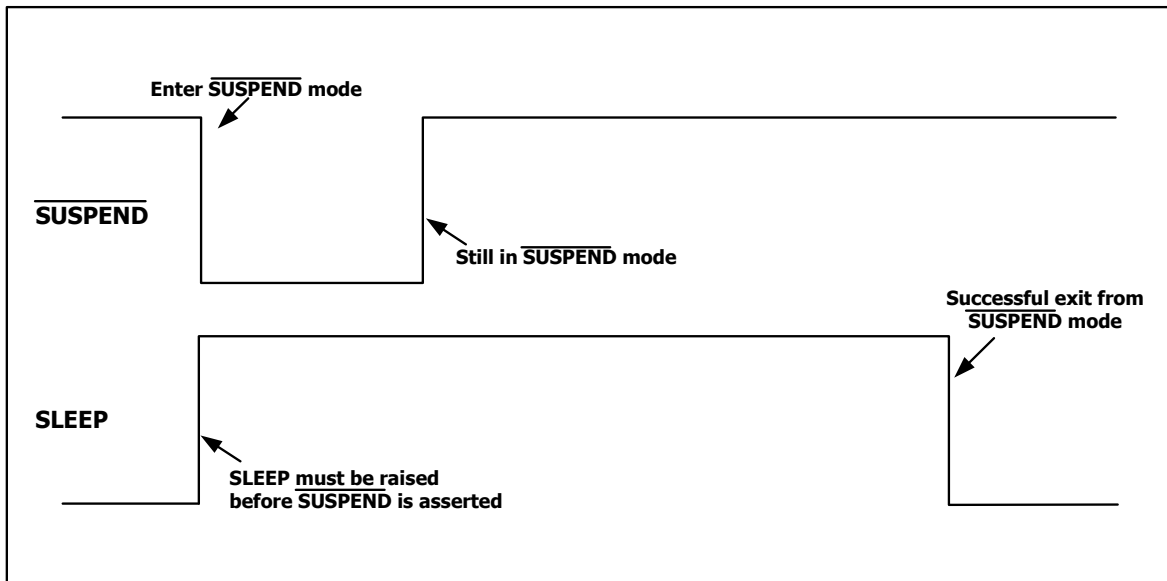
Figure 2-2: SUSPEND signal usage

2.6   Asynchronous Port Control (RTS)

When ANT is configured in asynchronous mode, a full duplex asynchronous serial port is provided with flow control for data transmission from the Host to ANT. The flow control is performed by the RTS signal, which conforms to standard hardware flow control CMOS signal levels. The signal may therefore be attached to a PC serial port (with use of an RS-232 level shifter), or to any other RS-232 device. The RTS signal is de-asserted for approximately 50 µs after each correctly formatted message has been received. This RTS signal duration is independent of the baud rate. Incorrect messages or partial messages are not acknowledged.

When ANT raises the RTS signal high, the Host MCU may not send any more data until the RTS signal is lowered again. There is no flow control for data being transmitted from ANT to the Host controller, and therefore the Host controller must be able to receive data at any time.

2.7 SLEEP ENABLE (SLEEP)

The SLEEP input signal allows ANT to sleep when the serial port is not required, helping conserve power. This control mechanism is illustrated below in

Figure 2-3.

This signal is essential for power savings in the nRF24AP1, but has less of an effect for the dual chip solutions.

Figure 2-3: ANT Sleep control

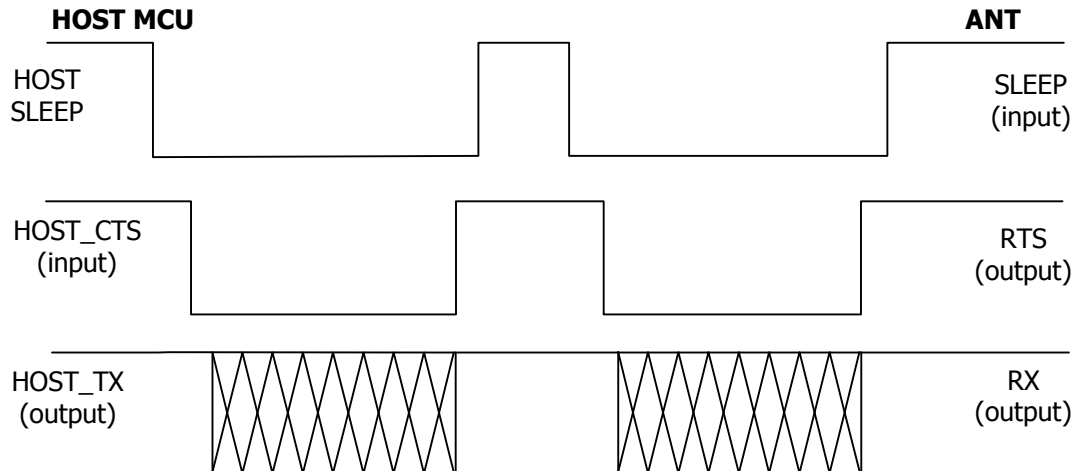If the SLEEP signal is not used, then it must be tied low. In this configuration, the ANT system will never sleep and will always be ready to receive data. The SUSPEND functionality cannot be used if the SLEEP signal is not used.

The SLEEP and RTS signals only affect the data being transferred from the Host MCU to ANT. ANT will send data to the Host, when available, regardless of the state of these two signals.

NOTE: The RTS signal is raised by ANT after the last byte of a message has been received, and ANT will therefore lose any bytes that were sent, or in the process of being sent, before the RTS signal is acted upon by the Host MCU, and the transmission is halted. To avoid this problem, either the messages need to be spaced apart by the Host MCU or 0-pad bytes need to be added to the end of each message being transmitted to handle whatever byte pipeline is in place. For example, when considering PC communication, two 0-bytes must be appended to every message, since PCs interpret CTS at the driver rather than the hardware level. ANT will discard 0-pad bytes received. This issue usually occurs only when using burst transfers from the Host to ANT and high data rates are expected.

2.8 Link Layer Protocol

2.8.1 Characteristics

The ANT interface protocol has the following characteristics:

- Binary protocol
- Packets are of variable length
- Each packet contains an 8-bit Checksum
- Asynchronous data is transmitted with 1 start, 8 data bits, 1 stop bit and no parity, with standard CMOS level signaling
- Full duplex serial port

2.8.2 Message Structure

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

SYNC	LENGTH	ID	DATA_1	DATA_2	…	DATA_N	CHECKSUM	Opt. Zero Pad1	Opt. Zero Pad2

Each variable-length message is sent starting with the SYNC byte and ending with the CHECKSUM. Bytes are sent LSBit first.

2.8.3 Message Details

Byte #	Bit #	Name	Length	Description
0	7:0	SYNC	1 Byte	Fixed SYNC field = 10100100 (MSB:LSB)
1	-	LENGTH	1 Byte	Number of data byes in the message
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
N+3	-	CHECKSUM	1 Byte	XOR of all previous bytes (including SYNC)
N+4, N+5	-	Optional Zero PAD Bytes	1or2 Bytes	Zero PAD bytes may be required in conjunction with flow control when doing BURST transfers.

The following is an example of how to encode/decode an ANT serial message.

ANT_OpenChannel(1) -> SerialData (0xA4, 0x01, 0x4B, 0x01, 0xEF)

The details of the contents of this example serial message are shown in the table below.

Byte #	Name	Length	Data	Description
0	SYNC	1 Byte	0xA4	SYNC is always 0xA4
1	LENGTH	1 Byte	0x01	Number of Data bytes in this message = 1
2	ID	1 Byte	0x4B	ANT_OpenChannel message ID is 0x4B
3	DATA_1	1 Byte	0x01	There is 1 Data Byte in this message: This byte is Channel #. It has been set to Channel = 1
4	CHECKSUM	1 Byte	0xEF	0xA4 xor 0x01 xor 0x4B xor 0x01 = 0xEF

2.9   ANT Messages

Refer to ANT Message Protocol and Usage document for details on the different types of messages and overall ANT protocol description.

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.

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

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 3.11 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. The Host MCU retains full control of the message flow and can halt incoming messages as required.

3.2 Interconnect

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

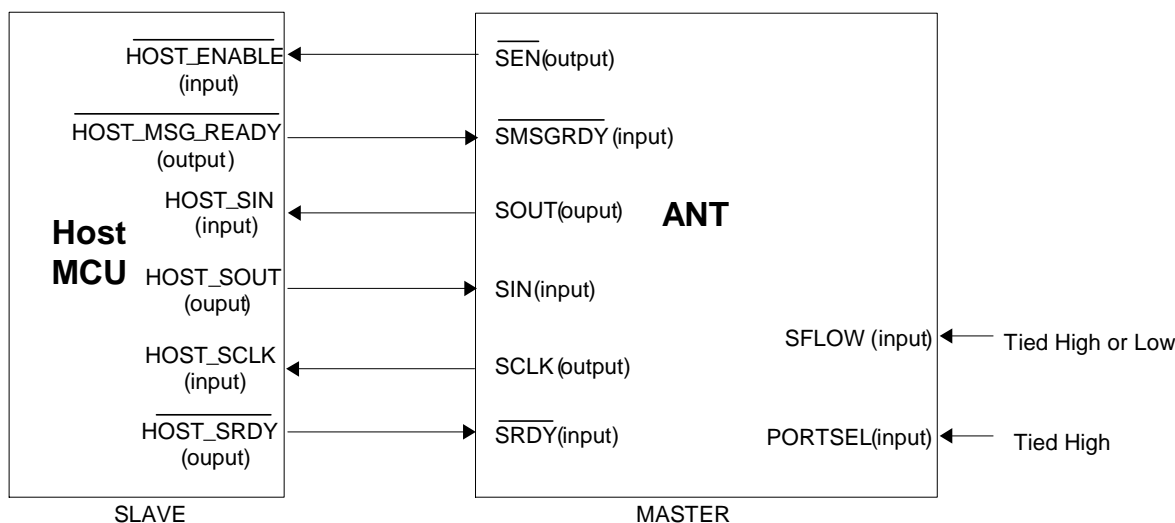**Figure 3-1: Synchronous Mode Connections**

3.3 Port Select (PORTSEL)

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

3.4 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.

3.5 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. The Host uses the $\overline{\text{SRDY}}$ flow control to signal its readiness for incoming messages.
- If the Host MCU wishes to send a message to ANT (for example to open a communications channel), the Host indicates it wishes to enter into transmit mode by asserting the $\overline{\text{SMSGRDY}}$ signal.
- $\overline{\text{SRDY}}$ must be asserted for communication to begin.
- In either receive or transmit mode, ANT always transmits the first byte of information output from SOUT, which is clocked with the SCLK signal (see Section on Electrical Specifications for details of clock frequency). The LSBit of this byte indicates the direction of future bytes (0 : Message Receive, ANT → Host; 1: Message Transmit, Host → ANT)
- If the Host MCU is in receive mode (default), additional message bytes will be transmitted the same way as the first byte, from ANT → Host MCU.
- If the Host MCU is in transmit mode, it must output its data to the ANT SIN input at the clock rate provided by the ANT SCLK signal.
- Data is transmitted least-significant-bit first

3.6 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 show in Section 3.11.

3.7 Link Layer Protocol

3.7.1 Characteristics

The ANT interface protocol has the following characteristics:

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

3.7.2 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

3.7.3 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	0xA5 xor 0x02 xor 0x52 xor 0x01 xor 0x03 = 0xF6

3.8 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 either 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 in the transaction sequence is always sent from ANT to the Host MCU. The first bit of the first byte dictates the direction for the remaining bytes in the transaction.

Shown below in Figure 3-2 to Figure 3-5 are examples of transactions between the Host and ANT in byte synchronous mode.
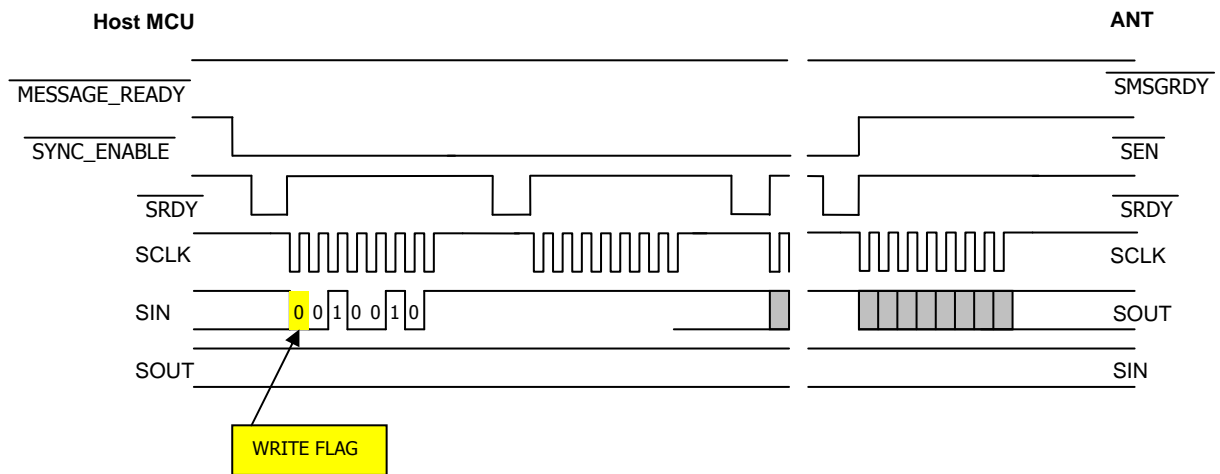

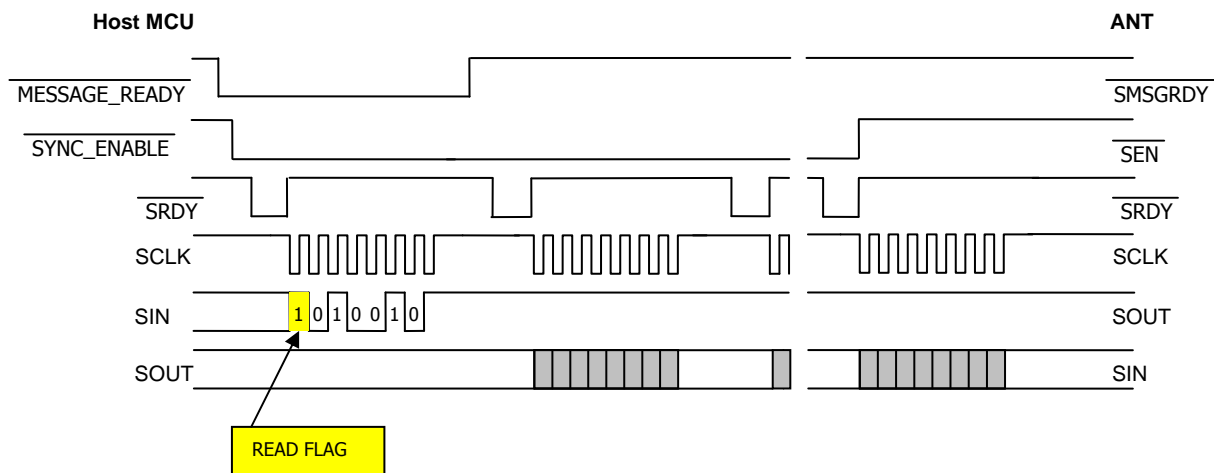Figure 3-2: ANT -> Host Transaction (Hardware $\overline{\text{SRDY}}$)

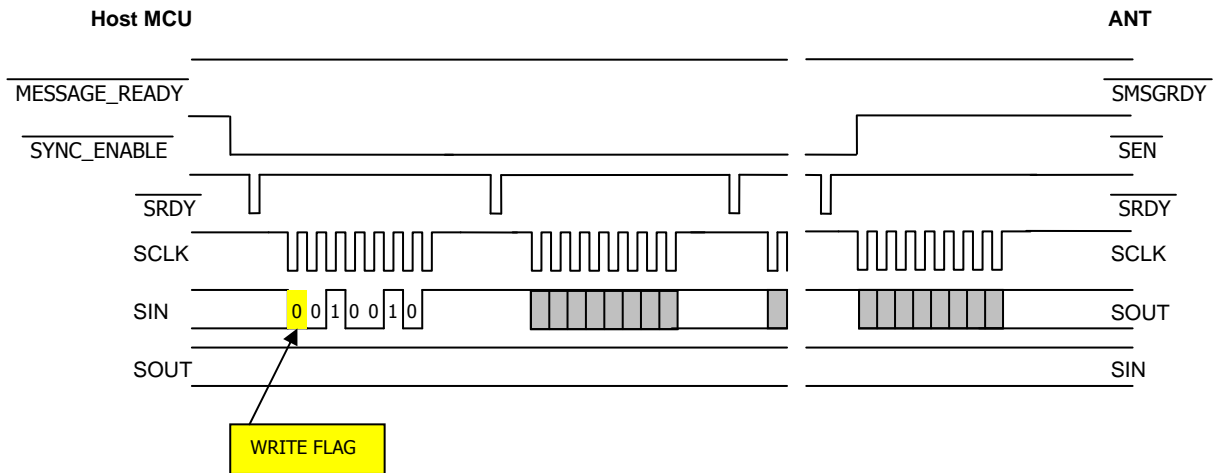Figure 3-3: Host -> ANT Transaction (Hardware $\overline{\text{SRDY}}$)

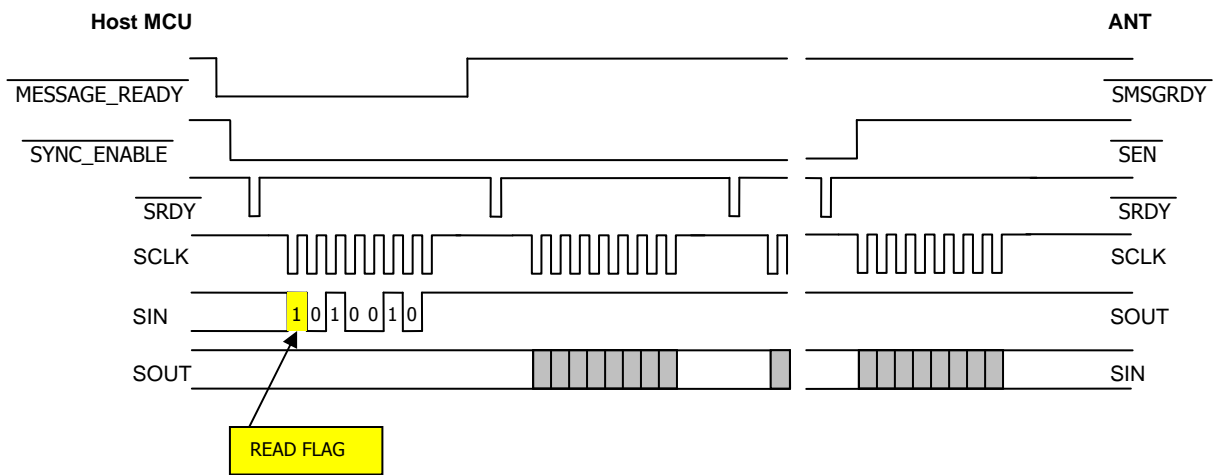Figure 3-4: ANT -> Host Transaction (Software $\overline{\text{SRDY}}$)

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

3.9   Synchronous Messaging with Bit Flow Control

If no hardware serial port is available on the Host MCU, ANT can be 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, but it becomes the following at the byte level.
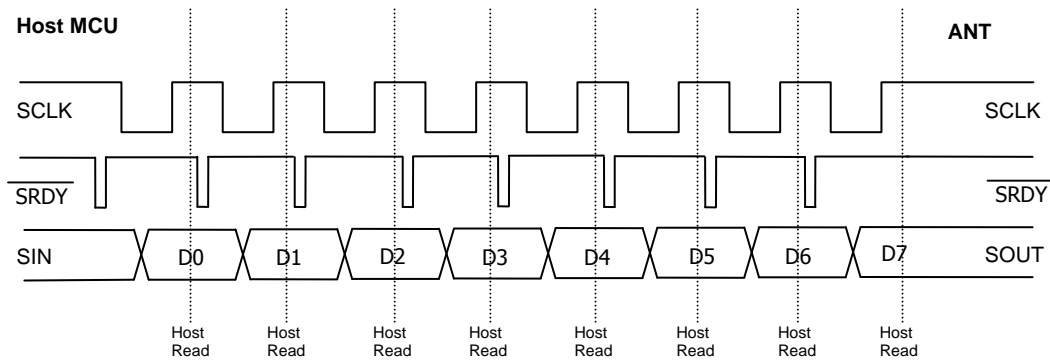
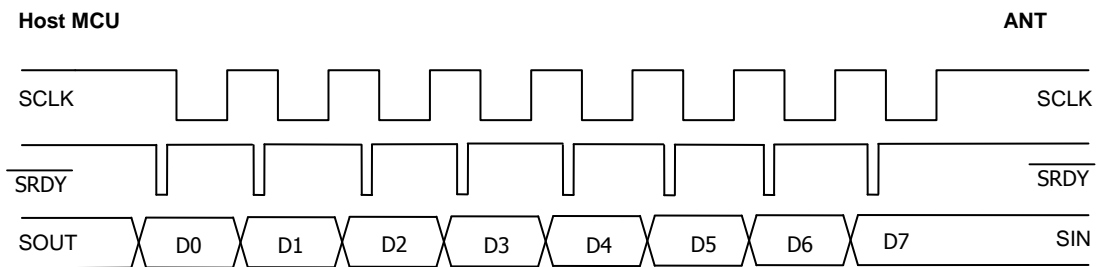**Figure 3-6: ANT -> Host Byte Transaction (Software Bit Flow Control)**

Figure 3-7: 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 when the byte is being transmitted from the Host MCU to ANT, where the first data bit will need to be asserted before the first clock edge. The final rising edge of the byte transaction will be the event to drive byte processing.

3.10 Serial Enable Control (ANT → Host)

The $\overline{\text{SEN}}$ signal, which is driven by ANT, will be asserted by ANT 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.
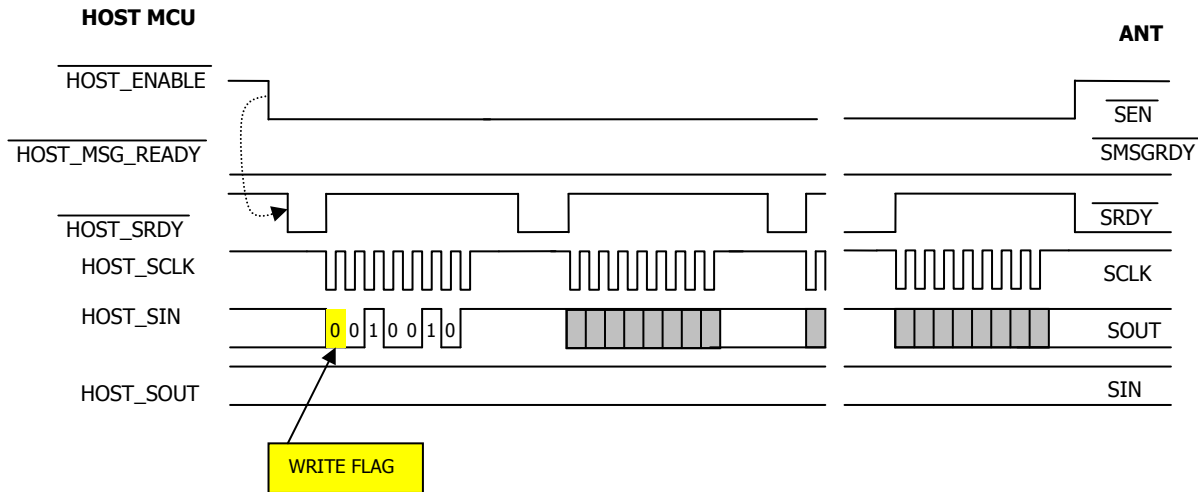
Figure 3-8: Serial Enable Control using ANT

3.11 Synchronization

In order for the Host MCU to guarantee synchronization with ANT in startup conditions, a reset sequence must be applied to ANT. This is applicable to Synchronous mode communication only.
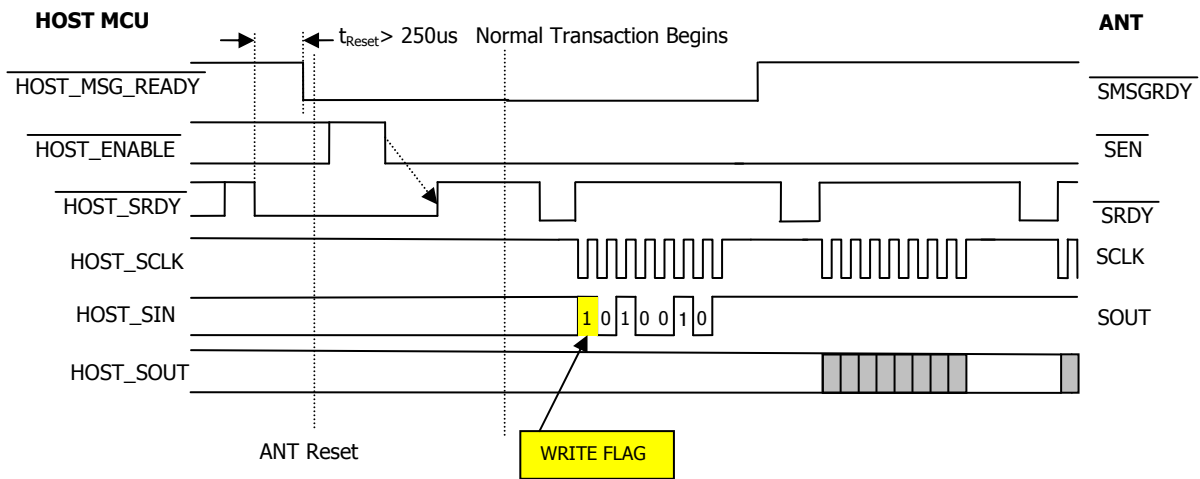

Figure 3-9: Synchronization with ANT upon start-up

3.12 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.
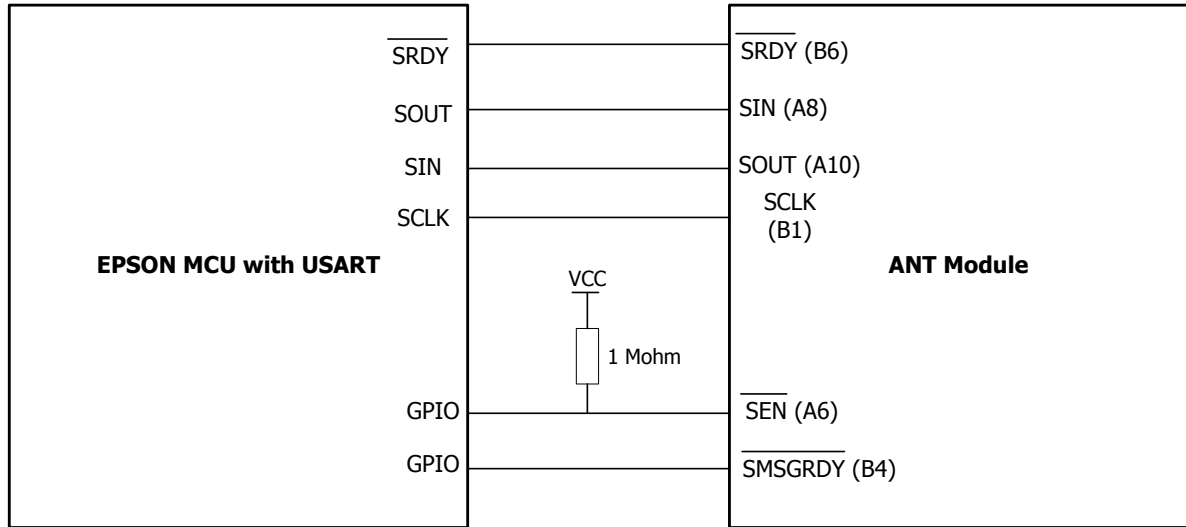
Figure 3-10: Example EPSON configuration for Byte Synchronous serial interface

For proper implementation of the above setup:

1. The GPIO connected to $\overline{\text{SEN}}$ must be configured as input.

2. The GPIO connected to $\overline{\text{SMSGRDY}}$ must be configured as output.

3. The EPSON USART must be configured as a synchronous slave.

4. Configure the $\overline{\text{SRDY}}$ pin to be controlled by the USART. Note that the register flag that causes the $\overline{\text{SRDY}}$ pin to go low while waiting for a new byte must not be cleared until the $\overline{\text{SEN}}$ signal is seen to go low from the ANT device. This is to avoid causing a synchronization condition as described in Section 3.11 above.

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