ANT+ Device Profile

BIKE SPEED AND CADENCE

ANT+ Managed Network Document
D00001163 Rev 1.3
Dynastream Innovations Inc.

P +1 403.932.9292 F +1 403.932.6521

228 River Avenue, Cochrane, Alberta, Canada T4C 2C1 thisisant.com

Copyright Information and Usage Notice

Revision History

Revision	Effective Date	Description
1.0	November 22, 2007	Creation
1.1	February 7, 2008	Integrated all bike speed and cadence information into a single document
1.2	September 23, 2008	Included section for use case Added new data pages for speed and cadence
1.3	January 28, 2011	Edited "Copyright Information and Usage Notice" section

Table of Contents

List of Tables

List of Figures

1 Overview of ANT+

The ANT+ managed networks are designed to promote interoperability of a new generation of sports and health devices. They combine the global 2.4GHz ANT RF products with common branding and an interface specification to allow manufacturers to build interoperable communicating sports and health equipment.

Interoperating equipment includes both sensors and receiver/display devices.

Examples of sensors include:

1. Speed and Distance Monitors
2. Heart Rate Monitors
3. Bike Speed Sensors
4. Bike Cadence Sensors
5. Combined Bike Speed and Cadence Sensors
6. Bike Power Sensors
7. Bike Component Sensors
8. Environment Sensors
9. Weight Scales

Examples of receiver/display units include:

1. Watches and Wrist-top Computers
2. Bike Computers
3. Cell Phones / PDAs

ANT+ sensors use ANT 2.4GHz low-power communication to transmit their data to the remote display devices. Sensor messages are embedded in ANT serial communications packets and have appropriate header and checksum information in the link layer.

The ANT+ sensor is designed to interface to an ANT receiver, which is embedded in the device that wishes to receive this sensor data. The information in this document assumes the user has knowledge of the ANT protocol, and is intended to be used in conjunction with the ANT Message Protocol and Usage document.

This document details the wireless communication link between the sensor and the receiving display device. The ANT+ sensor's typical use case, channel configuration, data message format, and implementation guidelines are detailed.

1.1 Interoperability

Interoperability of sensors within the ANT+ managed network is of paramount importance.

IMPORTANT: To have received this document you have agreed to and signed the ANT+ Managed Network license agreement and have received the ANT+ Managed Network Key. By signing this agreement and receiving the ANT+ device profiles you are agreeing to implement and test your product to this specification in its entirety. You are also agreeing to implement only ANT+ defined messages on the ANT+ managed network. This is essential to maintaining interoperability of all devices on the ANT+ managed network.

2 Related Documents

Refer to current versions of the listed documents. To ensure you are using the current versions, check the website or contact your ANT+ representative.

1. ANT Message Protocol and Usage
2. ANT Reference Design User Manual
3. ANT+ Reference Code

3 Typical Use Case of Bike Speed and Cadence Sensors

Bike speed sensors are devices mounted on a bicycle that measure the speed the bicycle is traveling. This is typically done using a magnet mounted on the wheel spokes and a sensor on the bicycle frame that senses the magnet passing. Bike cadence sensors measure the speed at which the user is pedaling, typically using a magnet attached to the pedal shaft and a sensor mounted on the frame. The standard mode of operation is for the bike speed or cadence sensor to transmit its measured data to the receiving display device. Typically this device is a bike computer, but it could be any ANT+ display device capable of decoding bike speed and cadence information, such as a watch, cell phone, PDA, etc.

Figure 1 illustrates how the bike sensors are typically used. The sensors transmit the user's speed and cadence information in the main data pages. Some device-specific information is transmitted at a slower rate in the background data pages.

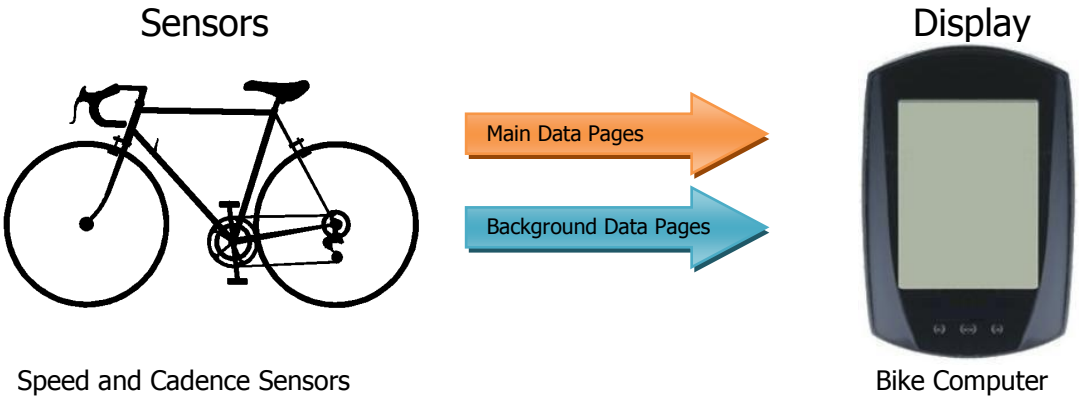

Figure 1: Typical Use Case of Bike Speed and Cadence Sensors

3.1 Various Use Cases of Bike Speed and Cadence Sensors

This document describes the wireless link between the transmitting bike sensors and the receiving display device. The sensors communicate with the receiving device in one of four modes:

1. Bike Speed Sensor Only – A single speed sensor on the bike requires a single ANT channel to communicate data.
2. Bike Cadence Sensor Only – A single cadence sensor on the bike requires a single ANT channel to communicate data.
3. Bike Speed and Cadence Sensors – Two sensors on the bike (speed and cadence) each require a single ANT channel to communicate data. The receiver must have two active channels, composed of modes 1 and 2, open.
4. Combined Bike Speed and Cadence Sensor – This mode has combined both the speed and cadence sensors into a single sensor. One sensor transmits data about the user's speed and cadence over a single ANT channel.

Figure 2 illustrates the described use cases. The receiving display device must be able to accept and display data from all four modes of operation. Therefore the use cases and technical content for the different sensors have been put into this single document.

Developers of devices intending to display bike speed and cadence information shall implement all the device definitions detailed in this document. The added firmware size to enable all three device profiles in the display device's firmware is a small cost for the large reward of interoperability with multiple devices.

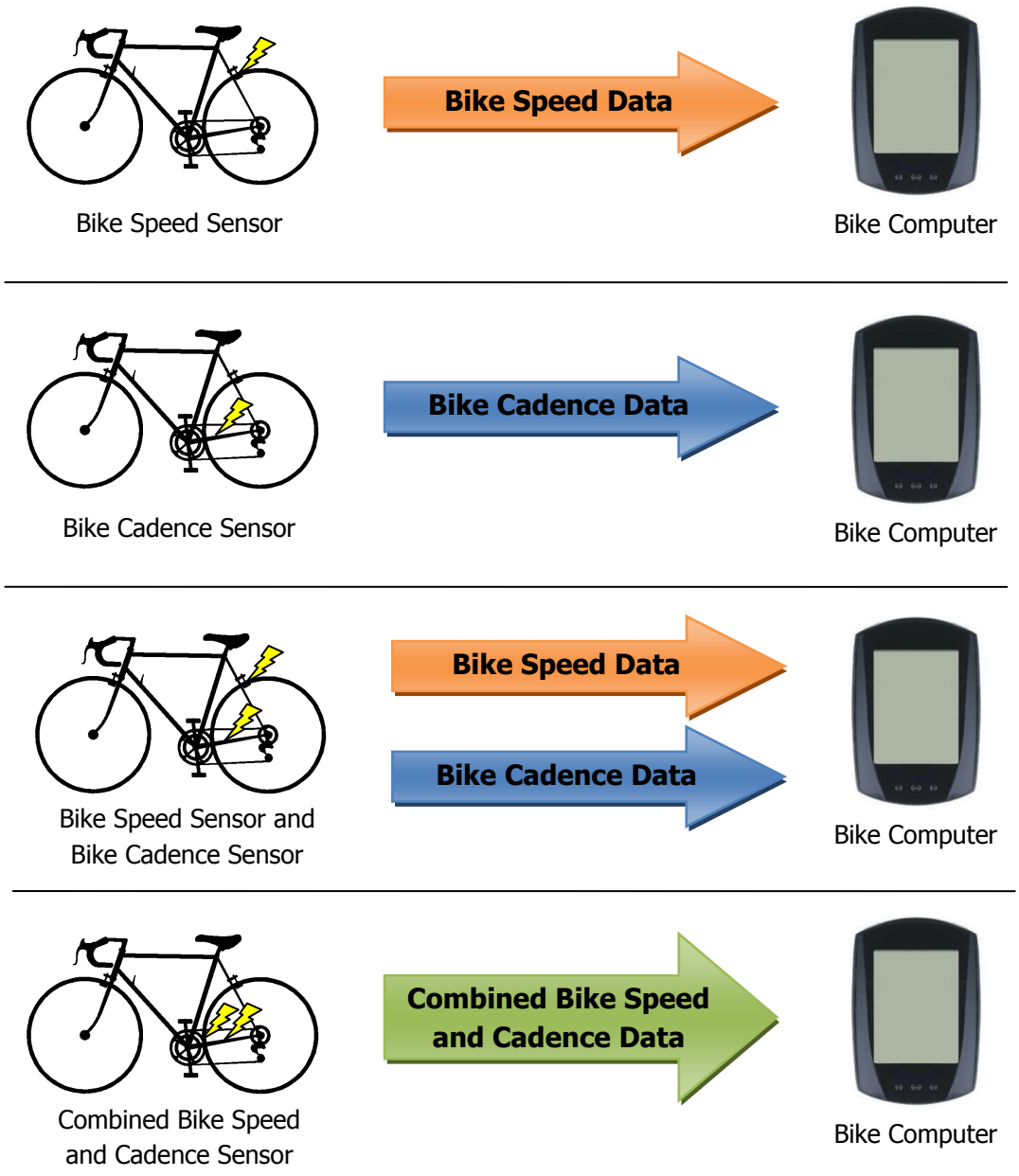

Figure 2: Various Use Cases of Bike Speed and Cadence Sensors

3.2 Messages Transmitted from the ANT+ Bike Speed and Cadence Sensors

It is important to note that all of the ANT+ messages sent by the ANT+ bike speed and cadence sensors — main data pages and background data pages — use page numbers to distinguish the different data page formats; the first byte of the data payload is always used to indicate the data page number. This message format allows a number of different pages to be sent by the sensor with different data in each page. All of the data pages transmit the user's current speed or cadence measurements; the critical speed or cadence information is sent in every message from the speed and cadence sensor. This enables new speed sensors, cadence sensors, and bike computers to be backwards compatible with existing ANT+ bike sensors and bike computers that adhere to an older version of the ANT+ specifications.

An ANT+ bike speed and/or cadence sensor transmits one main data page. Main data pages are sent at a rate of approximately 4Hz. The main data page definition varies by use case: speed only, cadence only, or combined speed and cadence. See sections 4.2.5, 5.2.5, and 6.2.2 accordingly.

An ANT+ bike speed and/or cadence sensor must transmit two background data pages. These required pages contain manufacturer information. An optional third page can be used to transmit cumulative operating time. A background data page is sent every 65th message. A further discussion of background data pages can be found in sections 4.2.2.1 and 5.2.2.1.

Figure 3 shows the different main data pages and background data pages that can be sent from the bike speed and/or cadence sensors. This figure also highlights how each data page contains the most recent speed or cadence information as shown by the highlighted box at the end of each arrow representing the respective ANT+ speed or cadence data pages.

NOTE: The combined speed and cadence sensor device profile does not define multiple data pages.

3.3 ANT+ Bike Speed and Cadence Display Device Implementations

It is important that the receiver be able to decode all of the data pages that can be sent from ANT+ bike speed and bike cadence sensors. This document gives code examples showing how to implement code that is compatible with all types of ANT+ bike speed and bike cadence sensors regardless of the data pages a specific implementation of a sensor supports. The ANT+ bike speed and bike cadence display device shall implement capabilities to decode all ANT+ bike speed, bike cadence, and combined bike speed and cadence data messages as outlined in this document.

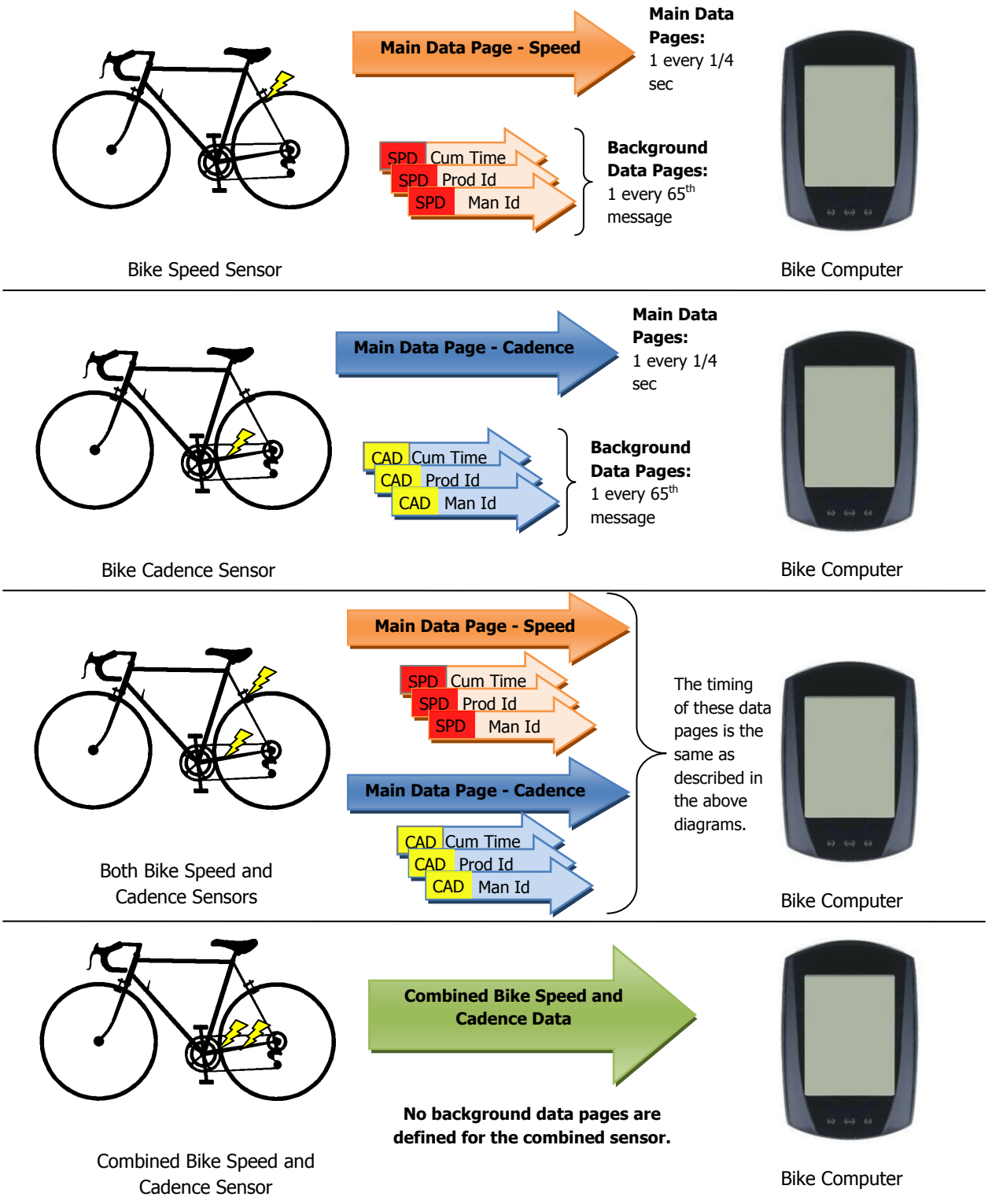

Figure 3: ANT+ Bike Speed and Cadence Use Cases and Associated Data Messages

4 Bike Speed Sensor

This section describes the channel parameters, data formats, and calculations specific to the ANT+ bike speed sensor. The single ANT channel is used in the first mode described in the use case – bike speed sensor only – and also as one of two channels that must be configured when both a bike speed sensor and a bike cadence sensor are mounted on the bike (third mode described in the use case).

4.1 Channel Configuration

The channel configuration parameters of the ANT+ bike speed sensor and any other ANT-enabled device is defined by the ANT protocol. Refer to the ANT Message Protocol and Usage document for definitions of the various channel parameters.

4.1.1 Receiver Channel Configuration

The device expected to receive data from an ANT+ bike speed sensor should configure an ANT channel with the parameters listed in Table 1.

Parameter	Value	Comment
Channel Type	Receive (0x00)	The bike speed sensor is a transmit channel device; therefore the display or storage device must be configured as the receiver.
Network Key	ANT+ Managed Network Key	The ANT+ Managed Network Key is governed by the ANT+ Managed Network licensing agreement.
RF Channel	57	Channel 57 is used for the ANT+ bike speed sensor.
Transmission Type	0 for pairing	The transmission type must be set to 0 for a pairing search. Once the transmission type is learned, the receiving device should remember the type for future searches. To be future compatible, any returned transmission type is valid. Future versions of this spec may allow additional bits to be set in the transmission type.
Device Type	123 (0x7B)	The device type shall be set to 123 (0x7B) when searching to pair to an ANT+ bike speed sensor. Please see the ANT Message Protocol and Usage document for more details.
Device Number	1 – 65535 0 for searching	The transmitting sensor contains a 16-bit number which uniquely identifies its transmissions. Set the Device Number parameter to zero to allow wildcard matching. Once the device number is learned, the receiving device should remember the number for future searches. Please see the ANT Message Protocol and Usage document for more details.
Message Period	8118 counts	Data is transmitted every 8118/32768 seconds (approximately 4.06Hz).
Search Timeout	(Default = 30 seconds)	The default search timeout is set to 30 seconds in the ANT protocol. This timeout is implementation specific and can be set by the designer to the appropriate value for the system.

Table 1: ANT Channel Configuration for Receiving Bike Speed Information

4.1.1.1 Message Period

The message period is set up so that the display device can receive data at the full rate (~4.06Hz) or at one half or one quarter of this rate; data can be received four times per second, twice per second, or once per second. The developer sets the message period count to receive data at one of the allowable receive rates:

- 8118 counts (~4.06Hz, 4 messages/second)
- 16236 counts (~2.02Hz, 2 messages/second)
- 32472 counts (~1.01Hz, 1 message/second)

The minimum receive rate allowed is 32472 counts (~1.01 Hz).

The longer the count (i.e. lower receive rate) the more power is conserved by the receiver but a trade off is made for the latency of the data as it is being updated at a slower rate. The implementation of the receiving message rate by the display device is chosen by the developer.

The new paging scheme of the bike speed sensor data allows for different pages of data to be sent. To incorporate receivers set at a slower receive rate the page toggle bit changes every 4th message to ensure that all receivers see this toggle bit. For more information on the page toggle bit see section 4.2.4.1.

4.1.1.2 Receiver Channel Code Example

Figure 4 shows a code example of how to establish a receive channel using the channel parameters described in Table 1. This example uses the definitions in the file ANT_DLL.h for understanding and clarity as these definitions are less involved than those for an embedded device. For more details on this code and examples of how to implement PC software please see the reference code provided with the ANT DLL.

```
/************************************************************************************************************
 * These are the steps to set up an ANT channel to receive ANT+ bike speed data.  The function
 * calls used here are based on the ANT PC Interface Functions.
 * **********************************************************************************************************/

#include "ant_dll.h"

void ConfigureRxChannel( UCHAR ucTransType, USHORT usDeviceNum )
{
   UCHAR ucNetNum     = 0x00;             //use network number 0
   UCHAR ucChanNum    = 0x00;             //assign channel 0
   UCHAR ucChanType   = 0x00;             //Receive or Slave channel
   UCHAR ucDeviceType = 0x7B;             //Set the device type to 0x7B specific for bike speed
   UCHAR ucRF         = 0x39;             //Set the RF frequency to channel 57 - 2.457GHz
   UCHAR ucSearchTime = 0x0C;             //Set the search time to be 30 seconds (set in 2.5s increments)
   //substitute the ANT+ managed network key here
   UCHAR aucNetKey[8] = {0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__};

   USHORT usMessagePeriod = 8118;         //Set the message period to 8118 counts specific for bike speed

   // Calls to set up the channel and open it.
   // Each call waits for an acknowledgement from ANT before continuing with the next call
   ANT_SetNetworkKey(ucNetNum, aucNetKey);
   if (!WaitAck(MESG_NETWORK_KEY_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Network Key.\n");

   ANT_AssignChannel(ucChanNum, ucChanType, ucNetNum);
   if (!WaitAck(MESG_ASSIGN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Assigning the Channel.\n");

   ANT_SetChannelId(ucChanNum, usDeviceNum, ucDeviceType, ucTransType);
   if (!WaitAck(MESG_CHANNEL_ID_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Channel ID.\n");

   ANT_SetChannelPeriod(ucChanNum, usMessagePeriod);
   if (!WaitAck(MESG_CHANNEL_MESG_PERIOD_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Message Period.\n");

   ANT_SetChannelRFFreq(ucChanNum, ucRF);
   if (!WaitAck(MESG_CHANNEL_RADIO_FREQ_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Radio Frequency.\n");

   ANT_OpenChannel(ucChanNum);
   if (!WaitAck(MESG_OPEN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Opening the Channel.\n");
}
```

Figure 4: Code Example of Receiver Channel Configuration

To search for a new bike speed sensor the ucTransType parameter should use a value of 0x00 for pairing and the usDeviceNum should use a value of 0x00 to allow for the wildcard search to take place. For more details on wildcard searching refer to the ANT Message Protocol and Usage document.

The code example in Figure 5 assumes that the ANT receiver is already receiving information from a new device that it used wildcard parameters to find. When the code is implemented the receiver finds the device number and transmission type parameters of the channel.

```
/*****************************************************************************************************
 * These are the steps to get the ANT channel parameters from an active ANT channel and save the device
 * number and transmission type to variables to be used for later pairing.
 * ***************************************************************************************************/
#include "ant_dll.h"

static void Main_ProcessANTEvent( void )
{
   UCHAR *aucRxBuf = ANTInterface_GetPendingEvent();
   ... // existing code
   switch (aucRxBuf[BUFFER_INDEX_MESG_ID] )  // switch the current event that needs processing
   {
      ... // existing code
      case MESG_BROADCAST_DATA_ID:
      case MESG_ACKNOWLEDGED_DATA_ID:
      case MESG_BURST_DATA_ID:
      {
         ... // existing code
         if ( usDeviceNum == 0 )
         {
            // To find the ANT channel parameters of the found channel the following call is made
            ANT_RequestMessage ( ucChanNum, MESG_CHANNEL_ID_ID );
         }
         ... // continue existing code
      }
      case MESG_CHANNEL_ID_ID:
      {
         // This is the message response from ANT to the ANT_RequestMessage() function.
         // The channel parameters Device Number and Transmission Type are saved to the variables
         usDeviceNum = aucRxBuf[1];
         usDeviceNum |= aucRxBuf[2] << 8;
         ucTransType = aucRxBuf[4];
         break;
      }
      ... // continue existing code
   }
}
```

Figure 5: Code Example of Finding and Saving Channel Parameters

4.1.2 Transmitter Channel Configuration

The ANT+ bike speed sensor shall establish its ANT channel as shown in Table 2.

Parameter	Value	Comment
Channel Type	Transmit (0x10)	Within the ANT protocol the transmit channel (0x10) allows for bi-directional communication channels and utilizes the interference avoidance techniques and other features inherent to the ANT protocol.
Network Key	ANT+ Managed Network Key	The ANT+ Managed Network Key is governed by the ANT+ Managed Network licensing agreement.
RF Channel	57	Channel 57 is used for the ANT+ bike speed sensor.
Transmission Type	1 (0x01)	ANT+ devices follow the transmission type definitions as outlined in the ANT protocol.
Device Type	123 (0x7B)	The device shall transmit its device type as 123 (0x7B). Please see the ANT Message Protocol and Usage document for more details.
Device Number	1-65535	This is a two-byte field that allows for a unique identification of a given bike speed sensor. It is imperative that the implementation allow for a unique device number to be assigned to a given device. **NOTE: The device number for the transmitting sensor shall not be 0x0000.**
Message Period	8118 counts	Data is transmitted every 8118/32768 seconds (approximately 4.06Hz).

Table 2: ANT Channel Configuration for Transmitting Bike Speed Information

4.1.2.1 Transmit Channel Type

The main reason for using the transmit channel type (0x10) is to make use of the interference avoidance technology inherent to the ANT protocol. Transmit channel types other than 0x10 do not make use of the interference avoidance techniques and are susceptible to interference from other 2.4GHz sources including other ANT and ANT+ devices.

4.1.2.2 Device Number

The device number needs to be as unique as possible across production units. An example of achieving this specification is to use the lowest two bytes of the serial number of the device for the device number of the ANT channel parameter.

The device number of the bike speed sensor shall not be 0x0000. Be careful if the device number is derived from the lower 16 bits of a larger serial number that multiples of 0x10000 (65536) do not cause the device number to be set to 0.

Data page 2 has been created specifically to allow for the resolution of a four-byte serial number. This page provides the upper two bytes of the serial number and assumes the lower two bytes are used as the device number in the ANT channel parameters. Refer to section 4.2.7 for details.

4.1.2.3 Transmitter Channel Code Example

Figure 6 shows a code example of how to establish a transmit channel using the channel parameters described in Table 2. This example uses the definitions in file ANT_DLL.h for understanding and clarity. For more details on this code and examples of how to implement PC software please see the reference code provided with the ANT DLL.

```
/************************************************************************************************
 * These are the steps to set up an ANT channel to transmit ANT+ bike speed data.  The function
 * calls used here are based on the ANT PC Interface Functions.
 * **********************************************************************************************/
#include "ant_dll.h"

void ConfigureTxChannel( USHORT usDeviceNum )
{
   UCHAR ucNetNum     = 0x00;             //use network number 0
   UCHAR ucChanNum    = 0x01;             //assign channel 1
   UCHAR ucChanType   = 0x10;             //Transmit or Master channel
   UCHAR ucDeviceType = 0x7B;             //Set the device type to 0x7B specific for bike speed
   UCHAR ucTransType  = 0x01;             //Set the transmission type to 1 specific for bike speed
   UCHAR ucRF         = 0x39;             //Set the RF frequency to channel 57 (2.457GHz)
   UCHAR ucTxPower    = 0x03;             //Set the Tx Power to 0dbm
   //substitute the ANT+ managed network key here
   UCHAR aucNetKey[8] = {0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__};
   USHORT usMessagePeriod = 8118;         //Set the message period to 8118 counts specific for bike speed

   // Calls to set up the channel and open it.
   // Each command call waits for an acknowledgement from ANT before continuing to send the next command
   ANT_SetNetworkKey(ucNetNum, aucNetKey);
   if (!WaitAck(MESG_NETWORK_KEY_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Network Key.\n");

   ANT_AssignChannel(ucChanNum, ucChanType, ucNetNum);
   if (!WaitAck(MESG_ASSIGN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Assigning the Channel.\n");

   ANT_SetChannelId(ucChanNum, usDeviceNum, ucDeviceType, ucTransType);
   if (!WaitAck(MESG_CHANNEL_ID_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Channel ID.\n");

   ANT_SetChannelPeriod(ucChanNum, usMessagePeriod);
   if (!WaitAck(MESG_CHANNEL_MESG_PERIOD_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Message Period.\n");

   ANT_SetChannelRFFreq(ucChanNum, ucRF);
   if (!WaitAck(MESG_CHANNEL_RADIO_FREQ_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Radio Frequency.\n");

   ANT_OpenChannel(ucChanNum);
   if (!WaitAck(MESG_OPEN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Opening the Channel.\n");
}
```

Figure 6: Code Example of Transmitter Channel Configuration

4.2 Bike Speed Sensor Message Payload Format

4.2.1 ANT+ Message Data Formats

All ANT messages have an 8-byte payload. For ANT+ messages, the first byte contains the data page number and the remaining 7 bytes are used for sensor-specific data. The 8-byte ANT+ message is referred to as a data page.

Byte	Description	Length
0	Data Page Number	1 byte
1-7	Sensor Specific Data	7 bytes

Table 3: ANT+ General Message Format

4.2.2 Data Page Types

Four different data pages are supported for the ANT+ bike speed sensor. These pages are divided into two distinct types of data. The first type is background information — data that is meant to be sent at a slower update rate. The second type is main information that is sent for most of the data transmissions. Main data pages contain data that change quickly and need to be monitored.

4.2.2.1 Background Data Pages

The background data pages send information about the sensor and include data pages 1, 2, and 3. These pages give information on cumulative operating time and manufacturer information. Data pages 2 and 3 must be implemented. Page 1 is not required and its implementation is left to the discretion of the manufacturer.

4.2.2.1.1 Transmission Timing

A background message is sent every 65th message. This timing allows the full manufacturer information and possibly the cumulative operating time to be transmitted at least once every 64.03 seconds.

4.2.2.2 Main Data Pages

The main data page is page 0. This page is continuously sent from the bike speed sensor with the exception of every 65th message used by a background page.

4.2.3 Receiving Data Pages

An ANT+ receiver that wants to be compatible with the bike speed device should implement all of the defined data pages in the device profile. This implementation is the only way that a receiver will be interoperable with existing and future ANT+ bike speed sensors.

4.2.4 Data Page Formats

The bike speed data format was one of the first defined ANT+ message formats. This bike speed data format does not conform to the general ANT+ message definition. In order to add the ability to use page numbers and maintain backwards compatibility with existing ANT+ implementations, the following special rules apply to this message format:

1. The most significant bit of the Data Page Number is reserved for a Page Change Toggle Bit. This bit must be seen to toggle before the rest of the Data Page Number can be interpreted.
2. Bytes 1 – 3 are the only bytes that change definition in a page.
3. Bytes 4 – 7 have the same definition for every data page. These bytes are the only data that can be interpreted before the Page Change Toggle Bit is seen to change.

4.2.4.1 Page Change Toggle Bit

The first byte of the bike speed data format comprises two data fields. Bits 0–6 determine the page number being used and identify the definition of the following three bytes.

The 7th bit or most significant bit (msb) is used for the page change toggle. The transmitter toggles the state of the toggle bit every fourth message (~1Hz) if the transmitter is using any of the page formats other than the page 0 data format. This allows the receiving/display unit to receive data from a bike speed sensor at a slower rate than 4Hz and still be able to observe the page change toggle bit to know that other data page formats are being used.

Figure 7 shows how the toggle bit changes every fourth message. When the receiver sees the toggle bit change the new data pages can be decoded.

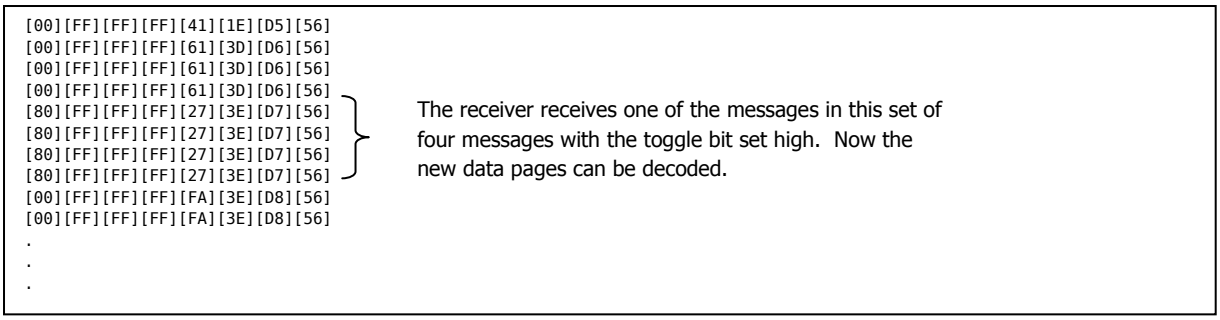

Figure 7: Example of Page Change Toggle Bit

4.2.5 Page 0 or Unknown Page Format

Page 0 transmits the latest bike speed event time and the cumulative speed revolution count of the bike wheel. Using these parameters a value for the user's speed can be calculated. See section 4.3 for details on these calculations.

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 bit (msb of Byte 0)	The transmitter must toggle this bit every 4th message. The receiver may not interpret Bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 bits (7 lsb of Byte 0, mask 0x7F)	Data Page Number = 0 (0x00)	N/A	N/A
1	Reserved	3 bytes	The transmitter shall set the value = 0xFF. The receiver does not interpret this data at this time.	N/A	N/A
2	Reserved				
3	Reserved				
4	Bike Speed Event Time LSB	2 bytes	Represents the time of the last valid bike speed event.	1/1024 second	64s
5	Bike Speed Event Time MSB				
6	Cumulative Speed Revolution Count LSB	2 bytes	Represents the total number of wheel revolutions.	Events	65536
7	Cumulative Speed Revolution Count MSB				

Table 4: Page 0 Bike Speed Data Format

4.2.5.1 Transmission Requirements

Page 0 is the main data page. Page 0 is sent consecutively 64 times, then a background data page is interleaved; this cycle is then repeated.

4.2.6 Page 1 Format – Cumulative Operating Time

Page 1 allows the receiver to determine the total time that the bike speed sensor has been active since the last battery change. The operating time increments by one count every two seconds, providing a maximum total time between rollovers of 33554432 seconds (9320 hours), which is greater than typical battery life. The operating time is reset when the battery is replaced.

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 bit (msb of Byte 0)	The transmitter must toggle this bit every 4th message. The receiver may not interpret Bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 bits (7 lsb of Byte 0, mask 0x7F)	Data Page Number = 1 (0x01)	N/A	N/A
1	Cumulative Operating Time (bits 0 - 7)	3 bytes	Increments every 2 seconds and is reset on battery replacement.	2- second intervals	33554432s
2	Cumulative Operating Time (bits 8 - 15)				
3	Cumulative Operating Time (bits 16 - 23)				
4	Bike Speed Event Time LSB	2 bytes	Represents the time of the last valid bike speed event.	1/1024 second	64s
5	Bike Speed Event Time MSB				
6	Cumulative Speed Revolution Count LSB	2 bytes	Represents the total number of wheel revolutions.	Events	65536
7	Cumulative Speed Revolution Count MSB				

Table 5: Page 1 Bike Speed Data Format – Cumulative Operating Time

4.2.6.1 Transmission Requirements

Page 1 is a background data page. A background message is sent every 65th message. The implementation of page 1 is left to the discretion of the manufacturer.

4.2.7 Page 2 Format – Manufacturer ID

Page 2 allows the manufacturer to uniquely identify the bike speed sensor on the ANT+ network by setting the manufacturer identification field and by populating the serial number. Although the serial number allows for only two bytes of data, if it is used in conjunction with the device number a four-byte serial number can be resolved.

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 bit (msb of Byte 0)	The transmitter must toggle this bit every 4th message. The receiver may not interpret Bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 bits (7 lsb of Byte 0, mask 0x7F)	Data Page Number = 2 (0x02)	N/A	N/A
1	Manufacturer ID	1 byte	Contact the ANT+ Alliance if you wish to be added to this list as a bike speed sensor manufacturer.	N/A	N/A
2	Serial Number LSB	2 bytes	This is the upper 16 bits of a 32-bit serial number.	N/A	N/A
3	Serial Number MSB				
4	Bike Speed Event Time LSB	2 bytes	Represents the time of the last valid bike speed event.	1/1024 second	64s
5	Bike Speed Event Time MSB				
6	Cumulative Speed Revolution Count LSB	2 bytes	Represents the total number of wheel revolutions.	Events	65536
7	Cumulative Speed Revolution Count MSB				

Table 6: Page 2 Bike Speed Data Format – Manufacturer ID

4.2.7.1 Transmission Requirements

Page 2 is a background data page. A background message is sent every 65th message. The implementation of page 2 is required by all manufacturers of ANT+ bike speed sensors.

4.2.7.2 Manufacturer ID

The list of manufacturer identification values is kept by the ANT+ Alliance. To obtain your unique manufacturer identification number please contact ANTAlliance@thisisant.com.

4.2.7.3 Serial Number Determination

The 16-bit device number allows a unique identification of the device in the RF domain, but cannot uniquely identify all manufactured bike speed sensors. When used in combination with the Manufacturer ID and the upper 16 bits of the serial number transmitted in this message, a unique identification of the bike speed sensor can be made.

The 32-bit serial number comprised of the upper serial number (most significant 16 bits) and the device number (least significant 16 bits) provides more than 4 billion serial numbers for each Manufacturer ID. The manufacturer must ensure that this data is unique for each bike speed sensor produced.

NOTE: The device ID must never be 0, therefore serial numbers that are integer multiples of 65536 must not be used.

4.2.8 Page 3 Format – Product ID

Page 3 allows the manufacturer to set and transmit hardware and software versions of the bike speed sensor as well as the model number.

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 bit (msb of Byte 0)	The transmitter must toggle this bit every 4th message. The receiver may not interpret Bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 bits (7 lsb of Byte 0, mask 0x7F)	Data Page Number = 3 (0x03)	N/A	N/A
1	Hardware Version	1 byte	To be set by the manufacturer.	N/A	N/A
2	Software Version	1 byte	To be set by the manufacturer.	N/A	N/A
3	Model Number	1 byte	To be set by the manufacturer.	N/A	N/A
4	Bike Speed Event Time LSB	2 bytes	Represents the time of the last valid bike speed event.	1/1024 second	64s
5	Bike Speed Event Time MSB				
6	Cumulative Speed Revolution Count LSB	2 bytes	Represents the total number of wheel revolutions.	Events	65536
7	Cumulative Speed Revolution Count MSB				

Table 7: Page 3 Bike Speed Data Format – Product ID

4.2.8.1 Transmission Requirements

Page 3 is a background data page. A background message is sent every 65th message. The implementation of page 3 is required by all manufacturers of ANT+ bike speed sensors.

4.2.9 Page 4 – 127 Formats

These pages are reserved for future use.

4.3 Bike Speed Message Data Filtering and Calculations

The bike speed sensor data provides only direct measurement of the number of wheel revolutions. Using the measurement time information an accurate estimate of speed can be made. The following calculation is recommended as a starting point. To calculate speed a calibration parameter for the wheel circumference must be available; in the calculations circumference is assumed to be in meters.

At time *t*, the last received event is event *N* and *N-1* is the event immediately preceding *N*.

NOTE: If the wheel or crank is revolving at less than 240RPM (4Hz), multiple messages may arrive that describe the same event.

4.3.1 Speed Computation

Instantaneous speed is computed as the distance difference between the two events, divided by the time difference of the two events:

$$Speed = \frac{Circumference * (RevCount_N - RevCount_{N-1}) * 1024}{(MeasTime_N - MeasTime_{N-1})} \text{ [meters/second]}$$

Equation 1: Speed Calculation

The accumulated distance can be calculated using the total number of counts (since the measurement was started, as shown) or by doing a time extrapolation to the display time.

$$Distance = Circumference * (RevCount_N - RevCount_0) \text{ [meters]}$$

Equation 2: Distance Calculation

More advanced distance algorithms could take into account longer term speed history. All distance calculations must account for measurement time and revolution count rollovers.

5 Bike Cadence Sensor

This section describes the channel parameters, data formats, and calculations specific to the ANT+ bike cadence sensor. The single ANT channel is used in the second mode described in the use case – bike cadence sensor only – and also as one of two channels that must be configured when both a bike speed sensor and a bike cadence sensor are mounted on the bike (third mode described in the use case).

5.1 Channel Configuration

The channel configuration parameters of the ANT+ bike cadence sensor and any other ANT-enabled device is defined by the ANT protocol. Refer to the ANT Message Protocol and Usage document for definitions of the various channel parameters.

5.1.1 Receiver Channel Configuration

The device expected to receive data from an ANT+ bike cadence sensor should configure an ANT channel with the parameters listed in Table 8.

Parameter	Value	Comment
Channel Type	Receive (0x00)	The bike cadence sensor is a transmit channel device; therefore the display or storage device must be configured as the receiver.
Network Key	ANT+ Managed Network Key	The ANT+ Managed Network Key is governed by the ANT+ Managed Network licensing agreement.
RF Channel	57	Channel 57 is used for the ANT+ bike cadence sensor.
Transmission Type	0 for pairing	The transmission type must be set to 0 for a pairing search. Once the transmission type is learned, the receiving device should remember the type for future searches. To be future compatible, any returned transmission type is valid. Future versions of this spec may allow additional bits to be set in the transmission type.
Device Type	122 (0x7A)	The device type shall be set to 122 (0x7A) when searching to pair to an ANT+ bike cadence sensor. Please see the ANT Message Protocol and Usage document for more details.
Device Number	1 – 65535 0 for searching	The transmitting sensor contains a 16-bit number which uniquely identifies its transmissions. Set the Device Number parameter to zero to allow wildcard matching. Once the device number is learned, the receiving device should remember the number for future searches. Please see the ANT Message Protocol and Usage document for more details.
Message Period	8102 counts	Data is transmitted every 8102/32768 seconds (approximately 4.04Hz).
Search Timeout	(Default = 30 seconds)	The default search timeout is set to 30 seconds in the ANT protocol. This timeout is implementation specific and can be set by the designer to the appropriate value for the system.

Table 8: ANT Channel Configuration for Receiving Bike Cadence Information

5.1.1.1 Message Period

The message period is set up so that the display device can receive data at the full rate (~4.04Hz) or at one half or one quarter of this rate; data can be received four times per second, twice per second, or once per second. The developer sets the message period count to receive data at one of the allowable receive rates:

- 8102 counts (~4.04Hz, 4 messages/second)
- 16204 counts (~2.02Hz, 2 messages/second)
- 32408 counts (~1.01Hz, 1 message/second)

The minimum receive rate allowed is 32408 counts (~1.01Hz).

The longer the count (i.e. lower receive rate) the more power is conserved by the receiver but a trade off is made for the latency of the data as it is being updated at a slower rate. The implementation of the receiving message rate by the display device is chosen by the developer.

The new paging scheme of the bike cadence sensor data allows for different pages of data to be sent. To incorporate receivers set at a slower receive rate the page toggle bit changes every 4th message to ensure that all receivers see this toggle bit. For more information on the page toggle bit see section 5.2.4.1.

5.1.1.2 Receiver Channel Code Example

Figure 8 shows a code example of how to establish a receive channel using the channel parameters described in Table 8. This example uses the definitions in the file ANT_DLL.h for understanding and clarity as these definitions are less involved than those for an embedded device. For more details on this code and examples of how to implement PC software please see the reference code provided with the ANT DLL.

```
/*****************************************************************************************************
 * These are the steps to set up an ANT channel to receive ANT+ bike cadence data.  The function
 * calls used here are based on the ANT PC Interface Functions.
 * ***************************************************************************************************/

#include "ant_dll.h"

void ConfigureRxChannel( UCHAR ucTransType, USHORT usDeviceNum )
{
   UCHAR ucNetNum     = 0x00;             //use network number 0
   UCHAR ucChanNum    = 0x00;             //assign channel 0
   UCHAR ucChanType   = 0x00;             //Receive or Slave channel
   UCHAR ucDeviceType = 0x7A;             //Set the device type to 0x7A specific for bike cadence
   UCHAR ucRF         = 0x39;             //Set the RF frequency to channel 57 - 2.457GHz
   UCHAR ucSearchTime = 0x0C;             //Set the search time to be 30 seconds (set in 2.5s increments)
   //substitute the ANT+ managed network key here
   UCHAR aucNetKey[8] = {0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__};

   USHORT usMessagePeriod = 8102;         //Set the message period to 8102 counts specific for bike cadence

   // Calls to set up the channel and open it.
   // Each call waits for an acknowledgement from ANT before continuing with the next call
   ANT_SetNetworkKey(ucNetNum, aucNetKey);
   if (!WaitAck(MESG_NETWORK_KEY_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Network Key.\n");

   ANT_AssignChannel(ucChanNum, ucChanType, ucNetNum);
   if (!WaitAck(MESG_ASSIGN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Assigning the Channel.\n");

   ANT_SetChannelId(ucChanNum, usDeviceNum, ucDeviceType, ucTransType);
   if (!WaitAck(MESG_CHANNEL_ID_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Channel ID.\n");

   ANT_SetChannelPeriod(ucChanNum, usMessagePeriod);
   if (!WaitAck(MESG_CHANNEL_MESG_PERIOD_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Message Period.\n");

   ANT_SetChannelRFFreq(ucChanNum, ucRF);
   if (!WaitAck(MESG_CHANNEL_RADIO_FREQ_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Radio Frequency.\n");

   ANT_OpenChannel(ucChanNum);
   if (!WaitAck(MESG_OPEN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Opening the Channel.\n");
}
```

Figure 8: Code Example of Receiver Channel Configuration

To search for a new bike cadence sensor the ucTransType parameter should use a value of 0x00 for pairing and the usDeviceNum should use a value of 0x00 to allow for the wildcard search to take place. For more details on wildcard searching refer to the ANT Message Protocol and Usage document.

Refer to the code example in Figure 5 for details on how to save channel parameters from a wildcard search.

5.1.2 Transmitter Channel Configuration

The ANT+ bike cadence sensor shall establish its ANT channel as shown in Table 9.

Parameter	Value	Comment
Channel Type	Transmit (0x10)	Within the ANT protocol the transmit channel (0x10) allows for bi-directional communication channels and utilizes the interference avoidance techniques and other features inherent to the ANT protocol.
Network Key	ANT+ Managed Network Key	The ANT+ Managed Network Key is governed by the ANT+ Managed Network licensing agreement.
RF Channel	57	Channel 57 is used for the ANT+ bike cadence sensor.
Transmission Type	1 (0x01)	ANT+ devices follow the transmission type definitions as outlined in the ANT protocol.
Device Type	122 (0x7A)	The device shall transmit its device type as 122 (0x7A). Please see the ANT Message Protocol and Usage document for more details.
Device Number	1-65535	This is a two-byte field that allows for a unique identification of a given bike cadence sensor. It is imperative that the implementation allow for a unique device number to be assigned to a given device. **NOTE: The device number for the transmitting sensor shall not be 0x0000.**
Message Period	8102 counts	Data is transmitted every 8102/32768 seconds (approximately 4.04Hz).

Table 9: ANT Channel Configuration for Transmitting Bike Cadence Information

5.1.2.1 Transmit Channel Type

The main reason for using the transmit channel type (0x10) is to make use of the interference avoidance technology inherent to the ANT protocol. Transmit channel types other than 0x10 do not make use of the interference avoidance techniques and are susceptible to interference from other 2.4GHz sources including other ANT and ANT+ devices.

5.1.3 Device Number

The device number needs to be as unique as possible across production units. An example of achieving this specification is to use the lowest two bytes of the serial number of the device for the device number of the ANT channel parameter.

The device number of the bike cadence sensor shall not be 0x0000. Be careful if the device number is derived from the lower 16 bits of a larger serial number that multiples of 0x10000 (65536) do not cause the device number to be set to 0.

Data page 2 has been created specifically to allow for the resolution of a four-byte serial number. This page provides the upper two bytes of the serial number and assumes the lower two bytes are used as the device number in the ANT channel parameters. Please refer to section 5.2.7 for details.

5.1.3.1 Transmitter Channel Code Example

Figure 9 shows a code example of how to establish a transmit channel using the channel parameters described in Table 9. This example uses the definitions in file ANT_DLL.h for understanding and clarity. For more details on this code and examples of how to implement PC software please see the reference code provided with the ANT DLL.

```
/*****************************************************************************************************
 * These are the steps to set up an ANT channel to transmit ANT+ bike cadence data.  The function
 * calls used here are based on the ANT PC Interface Functions.
 * ***************************************************************************************************/
#include "ant_dll.h"

void ConfigureTxChannel( USHORT usDeviceNum )
{
   UCHAR ucNetNum     = 0x00;             //use network number 0
   UCHAR ucChanNum    = 0x01;             //assign channel 1
   UCHAR ucChanType   = 0x10;             //Transmit or Master channel
   UCHAR ucDeviceType = 0x7A;             //Set the device type to 0x7A specific for bike cadence
   UCHAR ucTransType  = 0x01;             //Set the transmission type to 1 specific for bike cadence
   UCHAR ucRF         = 0x39;             //Set the RF frequency to channel 57 (2.457GHz)
   UCHAR ucTxPower    = 0x03;             //Set the Tx Power to 0dbm
   //substitute the ANT+ managed network key here
   UCHAR aucNetKey[8] = {0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__};
   USHORT usMessagePeriod = 8102;         //Set the message period to 8102 counts specific for bike cadence

   // Calls to set up the channel and open it.
   // Each command call waits for an acknowledgement from ANT before continuing to send the next command
   ANT_SetNetworkKey(ucNetNum, aucNetKey);
   if (!WaitAck(MESG_NETWORK_KEY_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Network Key.\n");

   ANT_AssignChannel(ucChanNum, ucChanType, ucNetNum);
   if (!WaitAck(MESG_ASSIGN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Assigning the Channel.\n");

   ANT_SetChannelId(ucChanNum, usDeviceNum, ucDeviceType, ucTransType);
   if (!WaitAck(MESG_CHANNEL_ID_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Channel ID.\n");

   ANT_SetChannelPeriod(ucChanNum, usMessagePeriod);
   if (!WaitAck(MESG_CHANNEL_MESG_PERIOD_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Message Period.\n");

   ANT_SetChannelRFFreq(ucChanNum, ucRF);
   if (!WaitAck(MESG_CHANNEL_RADIO_FREQ_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Radio Frequency.\n");

   ANT_OpenChannel(ucChanNum);
   if (!WaitAck(MESG_OPEN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Opening the Channel.\n");
}
```

Figure 9: Code Example of Transmitter Channel Configuration

5.2 Bike Cadence Sensor Message Payload Format

5.2.1 ANT+ Message Data Formats

All ANT messages have an 8-byte payload. For ANT+ messages, the first byte contains the data page number and the remaining 7 bytes are used for sensor-specific data. The 8-byte ANT+ message is referred to as a data page.

Byte	Description	Length
0	Data Page Number	1 byte
1-7	Sensor Specific Data	7 bytes

Table 10: ANT+ General Message Format

5.2.2 Data Page Types

Four different data pages are supported for the ANT+ bike cadence sensor. These pages are divided into two distinct types of data. The first type is background information — data that is meant to be sent at a slower update rate. The second type is main information that is sent for most of the data transmissions. Main data pages contain data that change quickly and need to be monitored.

5.2.2.1 Background Data Pages

The background data pages send information about the device and include data pages 1, 2, and 3. These pages give information on cumulative operating time and manufacturer information. Data pages 2 and 3 must be implemented. Page 1 is not required and its implementation is left to the discretion of the manufacturer.

5.2.2.1.1 Transmission Timing

A background message is sent every 65th message. This timing allows the full manufacturer information and possibly the cumulative operating time to be transmitted at least once every 64.03 seconds.

5.2.2.2 Main Data Pages

The main data page is page 0. This page is continuously sent from the bike cadence sensor with the exception of every 65th message used by a background page.

5.2.3 Receiving Data Pages

An ANT+ receiver that wants to be compatible with the bike cadence device should implement all of the defined data pages in the device profile. This implementation is the only way that a receiver will be interoperable with existing and future ANT+ bike cadence sensors.

5.2.4 Data Page Formats

The bike cadence data format was one of the first defined ANT+ message formats. This bike cadence data format does not conform to the general ANT+ message definition. In order to add pages and maintain backwards compatibility, the following special rules apply to this message format:

4. The most significant bit of the Data Page Number is reserved for a Page Change Toggle Bit. This bit must be seen to toggle before the rest of the Data Page Number can be interpreted.
5. Bytes 1 – 3 are the only bytes that change definition in a page.
6. Bytes 4 – 7 have the same definition for every data page. These bytes are the only data that can be interpreted before the Page Change Toggle Bit is seen to change.

5.2.4.1 Page Change Toggle Bit

The first byte of the bike cadence data format comprises two data fields. Bits 0–6 determine the data page number being used and identify the definition of the following three bytes.

The 7th bit or most significant bit (msb) is used for the page change toggle. The transmitter toggles the state of the toggle bit every fourth message (~1Hz) if the transmitter is using any of the page formats other than the page 0 data format. This allows the receiving/display unit to receive data from a bike cadence sensor at a slower rate than 4Hz and still be able to observe the page change toggle bit to know that other data page formats are being used.

Figure 10 shows how the toggle bit changes every fourth message. When the receiver sees the toggle bit change the new data pages can be decoded.

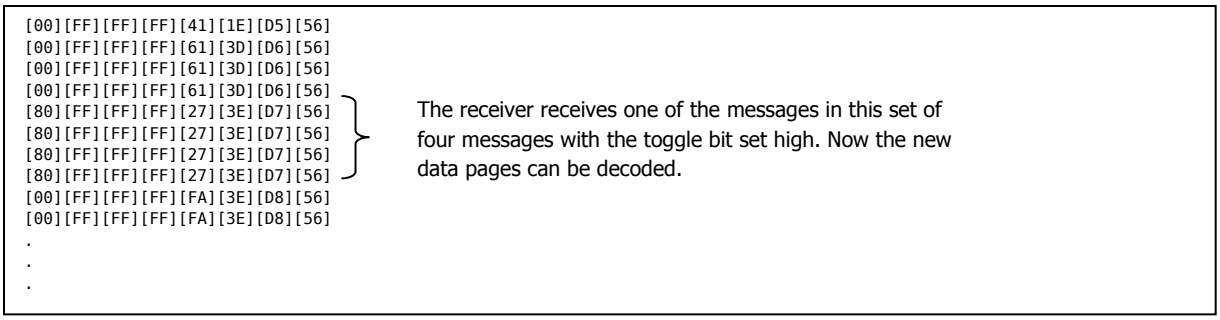

Figure 10: Example of Page Change Toggle Bit

5.2.5 Page 0 or Unknown Page Format

Page 0 transmits the latest bike cadence event time and the cumulative cadence revolution count of the bike wheel. Using these parameters a value for the user's cadence can be calculated. See section 5.3 for details on these calculations.

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 bit (msb of Byte 0)	The transmitter must toggle this bit every 4th message. The receiver may not interpret Bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 bits (7 lsb of Byte 0, mask 0x7F)	Data Page Number = 0 (0x00)	N/A	N/A
1	Reserved	3 bytes	The transmitter shall set the value = 0xFF. The receiver does not interpret this data at this time.	N/A	N/A
2	Reserved				
3	Reserved				
4	Bike Cadence Event Time LSB	2 bytes	Represents the time of the last valid bike cadence event.	1/1024 second	64s
5	Bike Cadence Event Time MSB				
6	Cumulative Cadence Revolution Count LSB	2 bytes	Represents the total number of pedal revolutions.	Events	65536
7	Cumulative Cadence Revolution Count MSB				

Table 11: Page 0 Bike Cadence Data Format

5.2.5.1 Transmission Requirements

Page 0 is the main data page. Page 0 is sent consecutively 64 times, then a background data page is interleaved; this cycle is then repeated.

5.2.6 Page 1 Format – Cumulative Operating Time

Page 1 allows the receiver to determine the total time that the bike cadence sensor has been active since the last battery change. The operating time increments by one count every two seconds, providing a maximum total time between rollovers of 33554432 seconds (9320 hours), which is greater than typical battery life. The operating time is reset when the battery is replaced.

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 bit (msb of Byte 0)	The transmitter must toggle this bit every 4th message. The receiver may not interpret Bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 bits (7 lsb of Byte 0, mask 0x7F)	Data Page Number = 1 (0x01)	N/A	N/A
1	Cumulative Operating Time (bits 0 - 7)	3 bytes	Increments every 2 seconds and is reset on battery replacement.	2-second intervals	33554432s
2	Cumulative Operating Time (bits 8 - 15)				
3	Cumulative Operating Time (bits 16 - 23)				
4	Bike Cadence Event Time LSB	2 bytes	Represents the time of the last valid bike cadence event.	1/1024 second	64s
5	Bike Cadence Event Time MSB				
6	Cumulative Cadence Revolution Count LSB	2 bytes	Represents the total number of pedal revolutions.	Events	65536
7	Cumulative Cadence Revolution Count MSB				

Table 12: Page 1 Bike Cadence Data Format – Cumulative Operating Time

5.2.6.1 Transmission Requirements

Page 1 is a background data page. A background message is sent every 65th message. The implementation of page 1 is left to the discretion of the manufacturer.

5.2.7 Page 2 Format – Manufacturer ID

Page 2 allows the manufacturer to uniquely identify the bike cadence sensor on the ANT+ network by setting the manufacturer identification field and by populating the serial number. Although the serial number allows for only two bytes of data, if it is used in conjunction with the device number a four-byte serial number can be resolved.

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 bit (msb of Byte 0)	The transmitter must toggle this bit every 4th message. The receiver may not interpret Bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 bits (7 lsb of Byte 0, mask 0x7F)	Data Page Number = 2 (0x02)	N/A	N/A
1	Manufacturer ID	1 byte	Contact the ANT+ Alliance if you wish to be added to this list as a bike cadence sensor manufacturer.	N/A	N/A
2	Serial Number LSB	2 bytes	This is the upper 16 bits of a 32-bit serial number.	N/A	N/A
3	Serial Number MSB				
4	Bike Cadence Event Time LSB	2 bytes	Represents the time of the last valid bike cadence event.	1/1024 second	64s
5	Bike Cadence Event Time MSB				
6	Cumulative Cadence Revolution Count LSB	2 bytes	Represents the total number of pedal revolutions.	Events	65536
7	Cumulative Cadence Revolution Count MSB				

Table 13: Page 2 Bike Cadence Data Format – Manufacturer ID

5.2.7.1 Transmission Requirements

Page 2 is a background data page. A background message is sent 65th message. The implementation of page 2 is required by all manufacturers of ANT+ bike cadence sensors.

5.2.7.2 Manufacturer ID

The list of manufacturer identification values is kept by the ANT+ Alliance. To obtain your unique manufacturer identification number please contact ANTAlliance@thisisant.com.

5.2.7.3 Serial Number Determination

The 16-bit device number allows a unique identification of the device in the RF domain, but cannot uniquely identify all manufactured bike cadence sensors. When used in combination with the Manufacturer ID and the upper 16 bits of the serial number transmitted in this message, a unique identification of the bike cadence sensor can be made.

The 32-bit serial number comprised of the upper serial number (most significant 16 bits) and the device number (least significant 16 bits) provides more than 4 billion serial numbers for each Manufacturer ID. The manufacturer must ensure that this data is unique for each bike cadence sensor produced.

NOTE: The device ID must never be 0, therefore serial numbers that are integer multiples of 65536 must not be used.

5.2.8 Page 3 Format – Product ID

Page 3 allows the manufacturer to set and transmit hardware and software versions of the bike cadence sensor as well as the model number.

Byte	Description	Length	Value	Units	Rollover
0	Page Change Toggle	1 bit (msb of Byte 0)	The transmitter must toggle this bit every 4th message. The receiver may not interpret Bytes 0-3 until it has seen this bit set to both a 0 and a 1.	N/A	N/A
0	Data Page Number	7 bits (7 lsb of Byte 0, mask 0x7F)	Data Page Number = 3 (0x03)	N/A	N/A
1	Hardware Version	1 byte	To be set by the manufacturer.	N/A	N/A
2	Software Version	1 byte	To be set by the manufacturer.	N/A	N/A
3	Model Number	1 byte	To be set by the manufacturer.	N/A	N/A
4	Bike Cadence Event Time LSB	2 bytes	Represents the time of the last valid bike cadence event.	1/1024 second	64s
5	Bike Cadence Event Time MSB				
6	Cumulative Cadence Revolution Count LSB	2 bytes	Represents the total number of pedal revolutions.	Events	65536
7	Cumulative Cadence Revolution Count MSB				

Table 14: Page 3 Bike Cadence Data Format – Product ID

5.2.8.1 Transmission Requirements

Page 3 is a background data page. A background message is sent every 65th message. The implementation of page 3 is required by all manufacturers of ANT+ bike cadence sensors.

5.2.9 Page 4 – 127 Formats

These pages are reserved for future use.

5.3 Bike Cadence Message Data Filtering and Calculations

The bike cadence sensor data provides only direct measurement of the number of pedal revolutions. Using the measurement time information an accurate estimate of cadence can be made. The following calculation is recommended as a starting point.

At time *t*, the last received event is event *N* and *N-1* is the event immediately preceding *N*.

NOTE: If the wheel or crank is revolving slowly, multiple messages may arrive that describe the same event.

5.3.1 Cadence Computation

Instantaneous cadence is computed as the revolution count difference between the two events, divided by the time difference of the two events:

$$Cadence = \frac{60 * (RevCount_{N} - RevCount_{N-1}) * 1024}{(MeasTime_{N} - MeasTime_{N-1})} \quad \text{[RPM]}$$

Equation 3: Cadence Calculation

More advanced revolution count algorithms could take into account longer term cadence history. All cadence calculations must account for measurement time and revolution count rollovers.

6 Combined Bike Speed and Cadence Sensor

This section describes the channel parameters, data formats, and calculations specific to the ANT+ combined bike speed and cadence sensor.

6.1 Channel Configuration

The channel configuration parameters of the ANT+ combined bike speed and cadence sensor and any other ANT-enabled device is defined by the ANT protocol. Refer to the ANT Message Protocol and Usage document for definitions of the various channel parameters.

6.1.1 Receiver Channel Configuration

The device expected to receive data from an ANT+ combined bike speed and cadence sensor should configure an ANT channel with the parameters listed in Table 15.

Parameter	Value	Comment
Channel Type	Receive (0x00)	The combined bike speed and cadence sensor is a transmit channel device; therefore the display or storage device must be configured as the receiver.
Network Key	ANT+ Managed Network Key	The ANT+ Managed Network Key is governed by the ANT+ Managed Network licensing agreement.
RF Channel	57	Channel 57 is used for the ANT+ combined bike speed and cadence sensor.
Transmission Type	0 for pairing	The transmission type must be set to 0 for a pairing search. Once the transmission type is learned, the receiving device should remember the type for future searches. To be future compatible, any returned transmission type is valid. Future versions of this spec may allow additional bits to be set in the transmission type.
Device Type	121 (0x79)	The device type shall be set to 121 (0x79) when searching to pair to an ANT+ combined bike speed and cadence sensor. Please see the ANT Message Protocol and Usage document for more details.
Device Number	1 – 65535 0 for searching	The transmitting sensor contains a 16-bit number which uniquely identifies its transmissions. Set the Device Number parameter to zero to allow wildcard matching. Once the device number is learned, the receiving device should remember the number for future searches. Please see the ANT Message Protocol and Usage document for more details.
Message Period	8086 counts	Data is transmitted every 8086/32768 seconds (approximately 4.05Hz).
Search Timeout	(Default = 30 seconds)	The default search timeout is set to 30 seconds in the ANT protocol. This timeout is implementation specific and can be set by the designer to the appropriate value for the system.

Table 15: ANT Channel Configuration for Receiving Combined Bike Speed and Cadence Information

6.1.1.1 Message Period

The message period is set up so that the display device can receive data at the full rate (~4.05Hz) or at one half or one quarter of this rate; data can be received four times per second, twice per second, or once per second. The developer sets the message period count to receive data at one of the allowable receive rates:

- 8086 counts (~4.05Hz, 4 messages/second)
- 16172 counts (~2.03Hz, 2 messages/second)
- 32344 counts (~1.01Hz, 1 message/second)

The minimum receive rate allowed is 32344 counts (~1.01Hz).

The longer the count (i.e. lower receive rate) the more power is conserved by the receiver but a trade off is made for the latency of the data as it is being updated at a slower rate. The implementation of the receiving message rate by the display device is chosen by the developer.

6.1.1.2 Receiver Channel Code Example

Figure 11 shows a code example of how to establish a receive channel using the channel parameters described in Table 15. This example uses the definitions in the file ANT_DLL.h for understanding and clarity as these definitions are less involved than those for an embedded device. For more details on this code and examples of how to implement PC software please see the reference code provided with the ANT DLL.

```
/*****************************************************************************************************
 * These are the steps to set up an ANT channel to receive ANT+ combined bike speed and cadence data.
 * The function calls used here are based on the ANT PC Interface Functions.
 * ***************************************************************************************************/

#include "ant_dll.h"

void ConfigureRxChannel( UCHAR ucTransType, USHORT usDeviceNum )
{
   UCHAR ucNetNum     = 0x00;             //use network number 0
   UCHAR ucChanNum    = 0x00;             //assign channel 0
   UCHAR ucChanType   = 0x00;             //Receive or Slave channel
   UCHAR ucDeviceType = 0x79;             //Set the device type to 0x79 specific for combined bike sensor
   UCHAR ucRF         = 0x39;             //Set the RF frequency to channel 57 - 2.457GHz
   UCHAR ucSearchTime = 0x0C;             //Set the search time to be 30 seconds (set in 2.5s increments)
   //substitute the ANT+ managed network key here
   UCHAR aucNetKey[8] = {0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__};

   USHORT usMessagePeriod = 8086;         //Set the message period to 8086 counts specific for combined bike sensor

   // Calls to set up the channel and open it.
   // Each call waits for an acknowledgement from ANT before continuing with the next call
   ANT_SetNetworkKey(ucNetNum, aucNetKey);
   if (!WaitAck(MESG_NETWORK_KEY_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Network Key.\n");

   ANT_AssignChannel(ucChanNum, ucChanType, ucNetNum);
   if (!WaitAck(MESG_ASSIGN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Assigning the Channel.\n");

   ANT_SetChannelId(ucChanNum, usDeviceNum, ucDeviceType, ucTransType);
   if (!WaitAck(MESG_CHANNEL_ID_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Channel ID.\n");

   ANT_SetChannelPeriod(ucChanNum, usMessagePeriod);
   if (!WaitAck(MESG_CHANNEL_MESG_PERIOD_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Message Period.\n");

   ANT_SetChannelRFFreq(ucChanNum, ucRF);
   if (!WaitAck(MESG_CHANNEL_RADIO_FREQ_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Radio Frequency.\n");

   ANT_OpenChannel(ucChanNum);
   if (!WaitAck(MESG_OPEN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Opening the Channel.\n");
}
```

Figure 11: Code Example of Receiver Channel Configuration

To search for a new combined bike speed and cadence sensor the ucTransType parameter should use a value of 0x00 for pairing and the usDeviceNum should use a value of 0x00 to allow for the wildcard search to take place. For more details on wildcard searching refer to the ANT Message Protocol and Usage document.

Refer to the code example in Figure 5 for details on how to save channel parameters from a wildcard search.

6.1.2 Transmitter Channel Configuration

The ANT+ combined bike speed and cadence sensor shall establish its ANT channel as shown in Table 16.

Parameter	Value	Comment
Channel Type	Transmit (0x10)	Within the ANT protocol the transmit channel (0x10) allows for bi-directional communication channels and utilizes the interference avoidance techniques and other features inherent to the ANT protocol.
Network Key	ANT+ Managed Network Key	The ANT+ Managed Network Key is governed by the ANT+ Managed Network licensing agreement.
RF Channel	57	Channel 57 is used for the ANT+ combined bike speed and cadence sensor.
Transmission Type	1 (0x01)	ANT+ devices follow the transmission type definitions as outlined in the ANT protocol.
Device Type	121 (0x79)	The device shall transmit its device type as 121 (0x79). Please see the ANT Message Protocol and Usage document for more details.
Device Number	1-65535	This is a two-byte field that allows for a unique identification of a given bike cadence sensor. It is imperative that the implementation allow for a unique device number to be assigned to a given device. **NOTE: The device number for the transmitting sensor shall not be 0x0000.**
Message Period	8086 counts	Data is transmitted every 8086/32768 seconds (approximately 4.05Hz).

Table 16: ANT Channel Configuration for Transmitting Combined Bike Speed and Cadence Information

6.1.2.1 Transmit Channel Type

The main reason for using the transmit channel type (0x10) is to make use of the interference avoidance technology inherent to the ANT protocol. Transmit channel types other than 0x10 do not make use of the interference avoidance techniques and are susceptible to interference from other 2.4GHz sources including other ANT and ANT+ devices.

6.1.3 Device Number

The device number needs to be as unique as possible across production units. An example of achieving this specification is to use the lowest two bytes of the serial number of the device for the device number of the ANT channel parameter.

The device number of the combined bike speed and cadence sensor shall not be 0x0000. Be careful if the device number is derived from the lower 16 bits of a larger serial number that multiples of 0x10000 (65536) do not cause the device number to be set to 0.

6.1.3.1 Transmitter Channel Code Example

Figure 12 shows a code example of how to establish a transmit channel using the channel parameters described in Table 16. This example uses the definitions in file ANT_DLL.h for understanding and clarity. For more details on this code and examples of how to implement PC software please see the reference code provided with the ANT DLL.

```
/*****************************************************************************************************
 * These are the steps to set up an ANT channel to transmit ANT+ bike cadence data.  The function
 * calls used here are based on the ANT PC Interface Functions.
 * ***************************************************************************************************/
#include "ant_dll.h"

void ConfigureTxChannel( USHORT usDeviceNum )
{
   UCHAR ucNetNum     = 0x00;             //use network number 0
   UCHAR ucChanNum    = 0x01;             //assign channel 1
   UCHAR ucChanType   = 0x10;             //Transmit or Master channel
   UCHAR ucDeviceType = 0x7A;             //Set the device type to 0x7A specific for bike cadence
   UCHAR ucTransType  = 0x01;             //Set the transmission type to 1 specific for bike cadence
   UCHAR ucRF         = 0x39;             //Set the RF frequency to channel 57 (2.457GHz)
   UCHAR ucTxPower    = 0x03;             //Set the Tx Power to 0dbm
   //substitute the ANT+ managed network key here
   UCHAR aucNetKey[8] = {0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__, 0x__};
   USHORT usMessagePeriod = 8102;         //Set the message period to 8102 counts specific for bike cadence

   // Calls to set up the channel and open it.
   // Each command call waits for an acknowledgement from ANT before continuing to send the next command
   ANT_SetNetworkKey(ucNetNum, aucNetKey);
   if (!WaitAck(MESG_NETWORK_KEY_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Network Key.\n");

   ANT_AssignChannel(ucChanNum, ucChanType, ucNetNum);
   if (!WaitAck(MESG_ASSIGN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Assigning the Channel.\n");

   ANT_SetChannelId(ucChanNum, usDeviceNum, ucDeviceType, ucTransType);
   if (!WaitAck(MESG_CHANNEL_ID_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Channel ID.\n");

   ANT_SetChannelPeriod(ucChanNum, usMessagePeriod);
   if (!WaitAck(MESG_CHANNEL_MESG_PERIOD_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Message Period.\n");

   ANT_SetChannelRFFreq(ucChanNum, ucRF);
   if (!WaitAck(MESG_CHANNEL_RADIO_FREQ_ID, MESSAGE_TIMEOUT))
      printf("Failed Setting the Radio Frequency.\n");

   ANT_OpenChannel(ucChanNum);
   if (!WaitAck(MESG_OPEN_CHANNEL_ID, MESSAGE_TIMEOUT))
      printf("Failed Opening the Channel.\n");
}
```

Figure 12: Code Example of Transmitter Channel Configuration

6.2 Combined Bike Speed and Cadence Sensor Message Payload Format

6.2.1 Data Page Formats

The combined bike speed and cadence data format was one of the first defined ANT+ message format and does not conform the standard ANT+ message definition. In order to add pages and maintain backwards compatibility the combined bike speed and cadence sensor does not have the ability to use a data paging system and therefore only has one data format that can be used.

6.2.2 Page 0 or Unknown Page Format

Page 0 transmits the latest bike cadence event time and the cumulative cadence revolution count of the bike wheel. Using these parameters a value for the user's cadence can be calculated. Page 0 also transmits the latest bike speed event time and the cumulative speed revolution count of the bike wheel. Using these parameters a value for the user's speed can be calculated. See sections 4.3 and 5.3 for calculations.

Byte	Description	Length	Value	Units	Rollover
0	Bike Cadence Event Time LSB	2 bytes	Represents the time of the last valid bike cadence event.	1/1024 second	64s
1	Bike Cadence Event Time MSB				
2	Cumulative Cadence Revolution Count LSB	2 bytes	Represents the total number of pedal revolutions.	Events	65536
3	Cumulative Cadence Revolution Count MSB				
4	Bike Speed Event Time LSB	2 bytes	Represents the time of the last valid bike speed event.	1/1024 second	64s
5	Bike Speed Event Time MSB				
6	Cumulative Speed Revolution Count LSB	2 bytes	Represents the total number of wheel revolutions.	Events	65536
7	Cumulative Speed Revolution Count MSB				

Table 17: Page 0 Combined Bike Speed and Cadence Data Format

6.2.2.1 Transmission Requirements

Page 0 is the only data page. This page is sent on every transmission of the combined bike speed and cadence sensor.

7 Bike Speed and Cadence Implementation Code Example

The ANT+ Reference Code contains files that show embedded sample code for implementing both a receiving device and a transmitting device. These code files have been written in C and have been compiled for a Texas Instruments MSP430 micro-processor.

Most of this code is re-usable on any embedded platform with necessary changes in the code to the serial drivers and some hardware specific code.

This code used in conjunction with the ANT+ Sensor Simulator and the ANT+ Display Simulator provides the necessary tools for ANT+ sensor and receiver development. For further details on the ANT+ reference code visit the ANT+ website at www.thisisant.com or contact the ANT+ Alliance at ANTAlliance@thisisant.com.