Time Sync Application Note

Copyright Information and Usage Notice

Revision History

Revision	Effective Date	Description
1.0	November 2016	Initial Release.

Table of Contents

1 Overview

The Time Sync feature is designed to help two devices synchronize periodic data. Examples of this could include making LEDs blink at the same time or synchronizing sensor data. This document will cover details of how to use the time sync functionality and its limitations.

This Application Note is for use with the ANT stack on the NRF52.

2 Reference Documents

Below is a list of documents that were used to create this document.

Table 2-1. Reference Documents

Doc ID	Doc Title	Link
D00000652	ANT Message Protocol and Usage	https://www.thisisant.com/resources/ant-message-protocol-and-usage/

3 API

The section describes the APIs that are associated with the time sync feature.

3.1 Time Base API

This API allows the application to configure the receiver.

3.1.1 ANT_TIME_BASE_TYPE Definitions

```
#define ANT_TIME_BASE_ANT      ((uint8_t)0x00)
#define ANT_TIME_BASE_ALT1     ((uint8_t)0x01)
#define ANT_TIME_BASE_ALT2     ((uint8_t)0x02)
```

This enumeration contains the type of time base to use. Only `ANT_TIME_BASE_ALT1`(RTC1) and `ANT_TIME_BASE_ALT2`(RTC2) should be used with the time sync feature.

3.1.2 ANT_TIME_STAMP_CONFIG Structure

```
typedef struct
{
   uint8_t ucTimeBase;
   bool bTimeStampEnabled;
} ANT_TIME_STAMP_CONFIG;
```

This structure is passed into the configuration function to set the time base and enable timestamps if needed. It is to be used in the application of the receiver of time sync messages. The `ucTimeBase` field is the time base to use. When `bTimeStampEnabled` field is set to true it will enable timestamps that will appear in the extended data of the received time sync message. The receiver should have timestamps enabled in order to calculate the next period of the timing event.

3.1.3 sd_ant_time_stamp_config_get/set Functions

```
uint32_t sd_ant_time_stamp_config_set(ANT_TIME_STAMP_CONFIG* pstTimeStampConfig)
uint32_t sd_ant_time_stamp_config_get(ANT_TIME_STAMP_CONFIG* pstTimeStampConfig)
```

This function allows the application to configure the time stamp feature for the receiver. It will set the time base and enable/disable timestamps. All channels are affected by these settings.

3.2 Time Sync API

This API allows the application to configure the transmitter and allows the transmitter to send time sync packets

3.2.1 ANT_TIME_SYNC_CONFIG Structure

```
typedef struct
{
   uint8_t ucTimeBase;
   bool bInvalidationEnabled;
   uint8_t ucInvalidationByte;
} ANT_TIME_SYNC_CONFIG;
```

This structure allows the transmitter to configure the SoftDevice for time sync packets. The `ucTimeBase` sets the time base to be used. Only `ANT_TIME_BASE_ALT1`(RTC1) and `ANT_TIME_BASE_ALT2`(RTC2) should be used with the time sync feature. The `bInvalidationEnabled` field allows the page number to be invalidated. This is useful for detecting stale data on the receiver side in case where the transmitter hasn't updated the data from the last timing event. The `ucInvalidationByte` field sets the value that will be used to invalidate the page. This value will overwrite the page number in the case of error such as stale data.

3.2.2 sd_ant_time_sync_config_get/set Functions

```
uint32_t sd_ant_time_sync_config_set(ANT_TIME_SYNC_CONFIG* pstTimeSyncConfig)
uint32_t sd_ant_time_sync_config_get(ANT_TIME_SYNC_CONFIG* pstTimeSyncConfig)
```

This function allows the user to configure the time sync feature for the transmitter. It will set the time base and configure the invalidation byte. All channels are affected by these settings.

3.2.3 sd_ant_time_sync_broadcast_tx Function

```
uint32_t sd_ant_time_sync_broadcast_tx(uint8_t ucChannel, uint8_t ucSize, uint8_t* aucMesg)
```

This function allows the application to send time sync data from the transmitter to the receiver. It behaves similarly to `sd_ant_time_sync_broadcast_tx` and will set the data to be transmitted during the next ANT traffic period. `ucChannel` is the channel to transmit on. `ucSize` is the size of the message. A time sync packet will always be eight bytes long. `aucMesg` is the buffer that contains the time sync packet. It should be formatted as described in section 4.1.

4 Time Sync Operation

The time sync packets have to be formatted for the stack to work with them. This section will describe how to format the packet, what the stack does with the data, and how it is to be used upon reception.

4.1 Packet format

The sync packet will be sent from a transmitter. It should be formatted as follows:

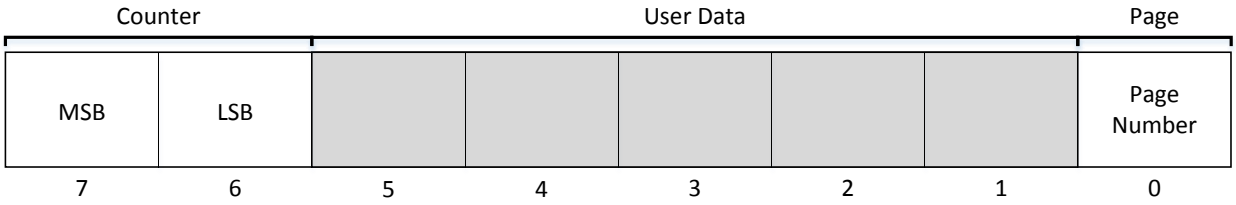

Figure 4-1. Time Sync Packet Format

The page number is one byte in size and is stored in offset zero. It is optional if the page is used or not. If it is used then the page value can be invalidated to let the receiver side know that there has been a problem on the transmit side.

The user data is not managed or manipulated by the stack. It can be used to help the application synchronize the events. For example, in the case of LEDs that should blink at the same time you may want to tell the receiver if the LED is on or off so the blink pattern is not reversed between the two LEDs.

The counter is the last time that the timing event occurred. This value is then used by the stack to calculate the offset between the timing event and the transmission time. This offset is used by the receiver along with the time stamp of the received packet to determine when the next interval should be.

4.2 Transmit side operation

This section describes the timeline for sending time sync packets to a receiver.

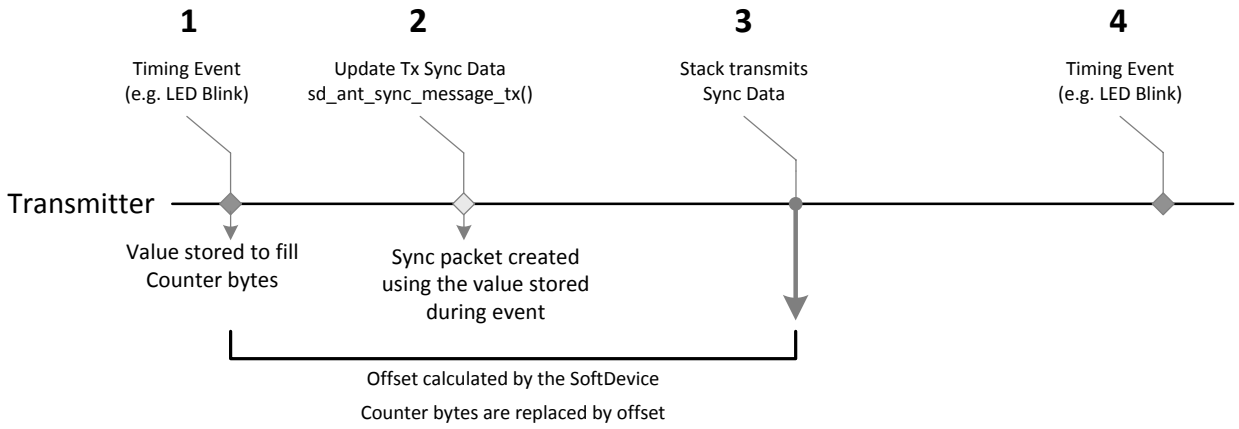

Figure 4-2. Transmitter Timeline

1. A timing event occurs that you want to synchronize with the receiver. The tick counter is captured, this value will be used to send to the ANT stack.
2. Format the time sync packet. This will include the two least significant bytes captured during 1. These two bytes will be placed into the counter value of the packet. A call to *sd_ant_time_sync_broadcast_tx* is made.
3. When the ANT stack is ready to transmit it will use the counter value passed in by *sd_ant_time_sync_broadcast_tx* to calculate the total offset between the timing event and the transmission. The counter value will be replaced by this offset.
4. Another event occurs. The receiver will be able to align itself with this event by using the offset in the time sync packet.

4.3 Receive side operation

This section is the timeline for receiving time sync data and using that data to calculate the next timing event.

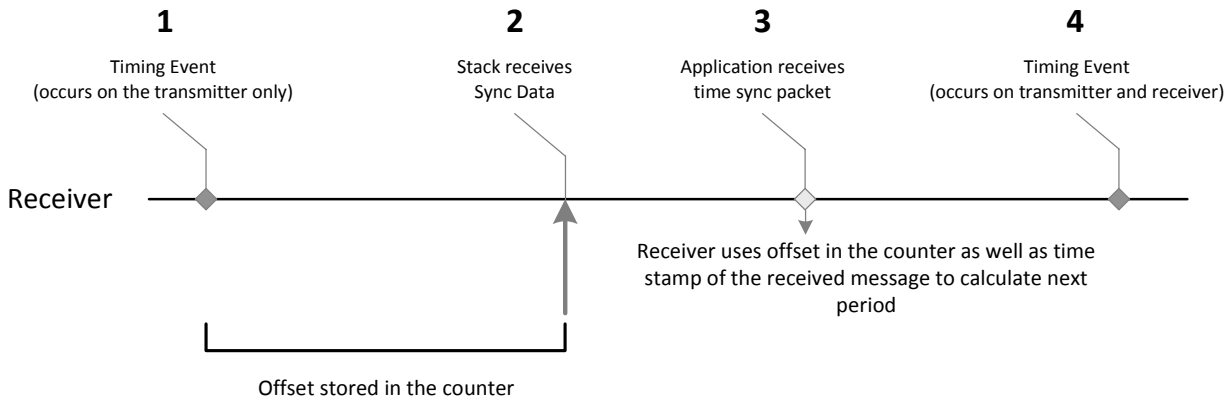

Figure 4-3. Receiver Timeline

1. This event happens on the transmitter side. No information has been given to the receiver yet.
2. The receiver receives the time sync packet. The ANT stack will time stamp the message (This has to be configured ahead of time).
3. The application receives an event that a time sync packet has arrived. It uses the offset stored in the counter value and the time stamp to determine the next timing event.
4. The receiver fires a timing event. At this point the devices will be synchronized.

4.4 Receive side calculation

The receiver will need to calculate the when the next event will happen on the transmitter device. To do this you can use the following calculation:

Next = RxTS – Offset + Period

Next: The tick count of the next event, the RTC's CC should be set to this.

RxTS: The timestamp of the received time sync packet

Offset: The value in the counter portion of the time sync packet

Period: The time between timing events in ticks

5 ANT Frequency

This section talks about what happens when you use the time sync feature with different ANT frequencies.

5.1 Normalized ANT traffic

If you run the timing event frequency at a similar frequency to the ANT channel the traffic will appear like:

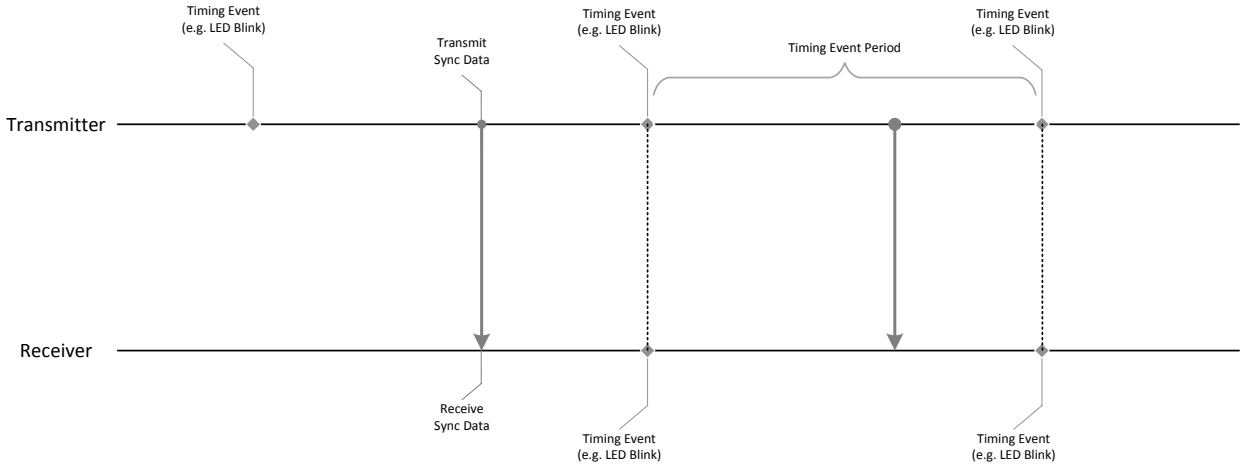

Figure 5-1. Normalized ANT frequency

Each timing event should have a call to replace the time sync data. When a transmission occurs it will update the receiver each timing period.

5.2 High frequency ANT traffic

If the ANT channel is running at a higher frequency then the timing events you will see traffic like this:

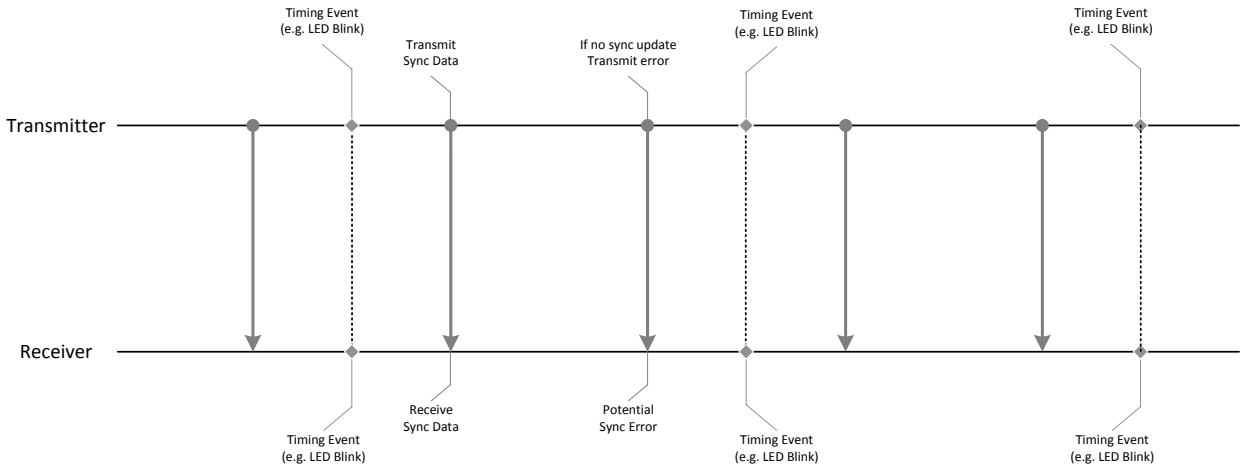

Figure 5-2. High ANT frequency

Unless you update the time sync packet each ANT interval on the transmitter the receiver will receive a stale data error. This will only happen in the page invalidation is enabled. If the page invalidation is not turned on the receiver may receive information that is incorrect.

5.3 Low frequency ANT traffic

If the ANT channel is running at a lower frequency then the timing events you will see traffic like this:

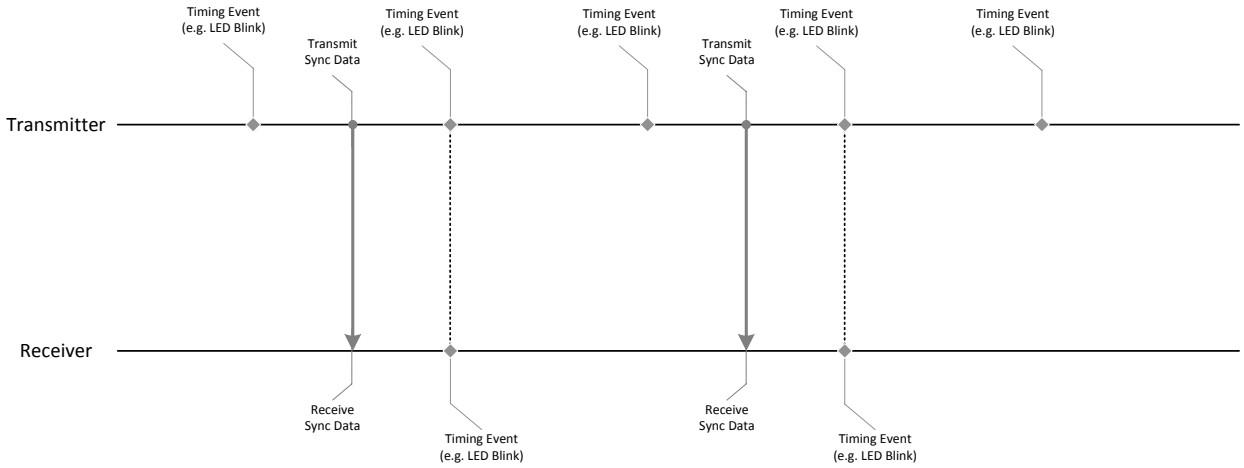

Figure 5-3. Low ANT frequency

The time sync won't keep up with the timing events and only the next timing event after a time sync packet will be able to synchronize. The receiver can derive that more events should be occurring.

6 Limitations

This section describes the limitations of the time sync features

6.1 Margin of Error

The time sync feature uses a clock tick for its time base. This means that there will be a margin of error the width of two ticks, one for each clock being used. In the case of a 32k RTC the margin of error will be 61us.

6.2 Prescaler

The prescaler shall not be used with the time sync feature.

6.3 Counter Size

The counter size is two bytes. The period between two timing events should not exceed the amount of time that can be stored in two bytes. For a 32k RTC this means that the timing period should be less than two seconds.

6.4 ANT Frequency

It is recommended that the ANT channel frequency is not the same as the frequency of the timing events. There can be some issues if the time sync packet data is updated in sync with the ANT channel.