S212/S332 Migration Document

Contents

Introduction

This document will cover how to migrate an application to newer versions of the S212/S332 SoftDevices. Specifically, this document addresses the required application changes related to the ANT portion of the SoftDevice. This document should be used in conjunction with the S132 migration notes located in this package for the SoC/BLE portions of a migration.

1.0.2->2.0.0 Migration

2.0.0 is not backward compatible with 1.0.2. The following API modifications are in version 2.0.0:

- The macro `ANT_ENABLE_GET_REQUIRED_SPACE` now takes an extra parameter that denotes the size of the event queue. Anything below `MIN_ANT_EVENT_QUEUE_SIZE` will use the internal event queue memory. If you are using the Nordic nRF5 SDK you will need to define the event queue size in ant_stack_configs_defs.h.

  Example:
  ```
  #define TOTAL_SIZE (ANT_ENABLE_GET_REQUIRED_SPACE(NUM_CHANNELS, NUM_ENCRYPTED_CHANNELS,
  SIZE_BURST_EVENTS, NUM_QUEUE_EVENTS))
  static __align(4) uint8_t aucANTChannelBlock[TOTAL_SIZE];
  ```

- The `ANT_ENABLE` struct has a new field named `usNumberOfEvents` which contains the number of events used in the `ANT_ENABLE_GET_REQUIRED_SPACE` macro.

  Example:
  ```
  stANTChannelEnable.ucTotalNumberOfChannels = NUM_CHANNELS;
  stANTChannelEnable.ucNumberOfEncryptedChannels = 0;
  stANTChannelEnable.usNumberOfEvents = NUM_QUEUE_EVENTS;
  stANTChannelEnable.pucMemoryBlockStartLocation = aucANTChannelBlock;
  stANTChannelEnable.usMemoryBlockByteSize = TOTAL_SIZE;
  ulErrCode = sd_ant_enable(&stANTChannelEnable);
  APP_ERROR_CHECK(ulErrCode);
  ```

- A new structure called `ANT_BUFFER_PTR` was added. This structure will replace the buffers being passed into the functions `sd_ant_coex_config_get` and `sd_ant_coex_config_set`.

  Example:

  Was:
  ```
  static uint8_t* aucPriorityOverride = { 0x0E, 0x01, 0x01, 0x02, 0x3A, 0x00, 0x3A, 0x00 };
  static uint8_t* aucAdvPriorityOverride = {0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
  sd_ant_coex_config_set(ANT_MS_CHANNEL_NUMBER, aucPriorityOverride, aucAdvPriorityOverride);
  APP_ERROR_CHECK(ulErrCode);
  ```

  Now:
  ```
  static ANT_BUFFER_PTR stPriorityOverride = {8, { 0x0E, 0x01, 0x01, 0x02, 0x3A, 0x00, 0x3A, 0x00 }};
  static ANT_BUFFER_PTR stAdvPriorityOverride = {8, { 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }};
  sd_ant_coex_config_set(ANT_MS_CHANNEL_NUMBER, &stPriorityOverride, &stAdvPriorityOverride);
  APP_ERROR_CHECK(ulErrCode);
  ```

0.9.1->1.0.2 Migration

1.0.2 is backward compatible with 0.9.1. The following API modifications are in version 1.0.2:

- *sd_ant_stack_reset* return codes were extended to include a failure code if the operation times out
- *sd_ant_channel_rx_search_timeout_set* renamed to *sd_ant_channel_search_timeout_set*
    - a macro has been provided to allow existing applications to continue using *sd_ant_channel_rx_search_timeout_set*
- *sd_ant_channel_search_timeout_set* can be used to configure the Group Transmitter Initiation feature

See ant_interface.h for further details.

Group Transmitter Initiation

Configuring group transmitter initiation requires an additional API call at channel setup:

```
sd_ant_channel_assign(0, CHANNEL_TYPE_MASTER, 0, 0);

sd_ant_channel_id_set(0, 33, 1, 1);

sd_ant_channel_period_set(0, 8192);  // 4 Hz Channel

sd_ant_channel_radio_freq_set(0, 57);

sd_ant_channel_radio_tx_power_set(0, RADIO_TX_POWER_LVL_3, 0);

sd_ant_channel_search_timeout_set(0, 2);  // Set group transmit initiator timeout to 500 ms

sd_ant_channel_open(0);
```

The above configuration will allow the Group Transmitter Initiation feature to operate for up to 500 ms. If the channel does not start during this period due to interference, the channel will revert to the normal startup method. Group Transmitter Initiation timeouts can be configured in increments of 250 ms. A timeout of zero disables the feature.

0.6.0->0.9.1 Migration

There are no changes to the ANT stack between these versions. The S132 migration notes should be reviewed as changes were made to the SoC and BLE APIs.

0.5.x –> 0.6.0 Migration

This section describes how to migrate from a 0.5.x release to a 0.6.0 release

Search Uplink

There are no API changes for Search Uplink but there are behaviour changes to Background Scan.

- Acknowledged messages and Burst Transfers can now be received by Background Scan channels.
- Broadcast messages and Acknowledged messages can now be sent on Background Scan channels using the standard APIs.
- If the previous Background Scan behaviour is desired, adjust the channel assignment to be `CHANNEL_TYPE_SLAVE_RX_ONLY`.

For example, to keep the previous Background Scan behavior:

```
err_code = sd_ant_channel_assign(ANT_CHANNEL_NUMBER,
                                 CHANNEL_TYPE_SLAVE,
                                 ANT_NETWORK_NUMBER,
                                 EXT_PARAM_ALWAYS_SEARCH);
```

would become:

```
err_code = sd_ant_channel_assign(ANT_CHANNEL_NUMBER,
                                 CHANNEL_TYPE_SLAVE_RX_ONLY,
                                 ANT_NETWORK_NUMBER,
                                 EXT_PARAM_ALWAYS_SEARCH);
```

0.3.0 -> 0.5.x Migration

This section describes how to migrate from a 0.3.0 release to a 0.5.x release

License Key

A new parameter added to the `sd_softdevice_enable()` call, `p_license_key`, is used to enable SoftDevices that include ANT. Without this parameter, the SoftDevice will not initialize and will be non-functional. For evaluation purposes, an evaluation key is provided in nrf_sdm.h. It is a requirement to obtain and use a commercial use license key with the S332/S212 in any product that is sold or otherwise distributed for revenue-generating purposes. In order to use the evaluation key, two steps are required:

1. Open nrf_sdm.h and uncomment the `ANT_LICENSE_KEY` definition, as shown in the code snippet below:

```
/** @brief Evaluation key for the ANT SoftDevices. Use this key in sd_softdevice_enable call to start
           using the evaluation stacks.You MUST obtain a valid commercial license key BEFORE releasing a
           product to market that uses the ANT SoftDevices. For more information about licensing please
           visit the website below:
           https://www.thisisant.com/developer/ant/licensing */

//#define ANT_LICENSE_KEY "3831-521d-7df9-24d8-eff3-467b-225f-a00e"    ← UNCOMMENT

#ifndef ANT_LICENSE_KEY
#error "You must obtain a valid license key to use ANT. You may use the evaluation key for non commercial
use only by uncommenting it above this error. Commercial use license keys are available from ANT
Wireless."
#endif
```

2. Modify all calls to `sd_softdevice_enable()` to add the license key as the third variable. The call should be changed as follows from this:

   ```
   sd_softdevice_enable(lf_clock_source, softdevice_assert_callback);
   ```

   to this:

   ```
   sd_softdevice_enable(lf_clock_source, softdevice_assert_callback, ANT_LICENSE_KEY);
   ```