s210_nrf51422 migration document

Table of Contents

Introduction to the s210_nrf51422 migration document

This document describes how to migrate to a new version of the s210_nrf51422 SoftDevice. The s210_nrf51422 release notes should be read in conjunction with this document. This document has the following two main sections for each version of the s210_nrf51422:

- **Required changes** describes how an application would have used the previous version of the SoftDevice, and the changes that must be made in the application for it to work with a new version.
- **New functionality** describes how to use the new features and functionality offered by a new version of the SoftDevice. Note that not all new functionality may be covered; the release notes will contain a full list of new features and functionality.

How to use the document

Each section describes how to migrate to a new version from a previous version of the SoftDevice.

To migrate between versions that are more than one version apart, follow the migration steps for all intermediate versions in sequential order.

S210_nrf51422_4.0.0

This section describes how to migrate to s210_nrf51422_4.0.0 from s210_nrf51422_3.0.0

Required changes

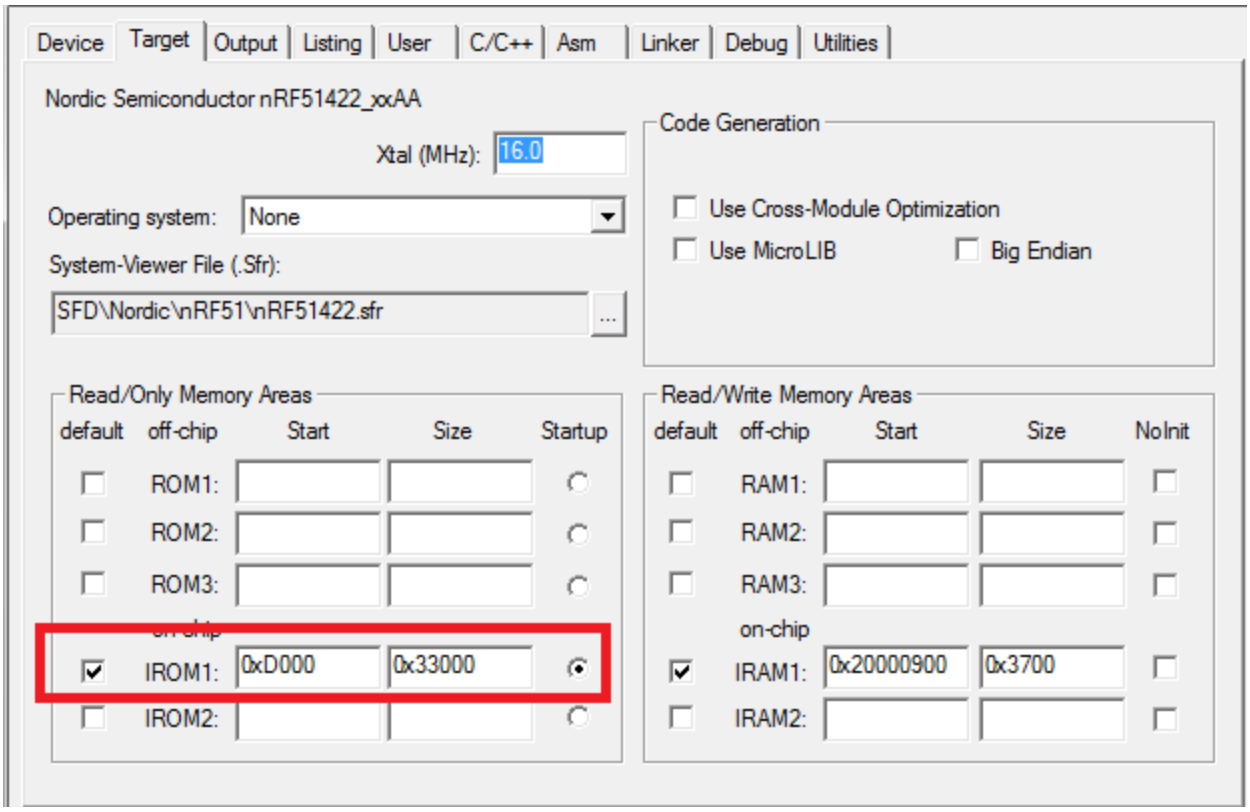
SoftDevice size

The size of the SoftDevice has changed to accommodate the inclusion of the Master Boot Record (MBR). The required application project file changes are as follows:

Example Keil project properties under ‘Target’ tab:

- Change IROM1 Start to 0xD000
- Ensure that IROM1 size is no more than 0x33000

If the project uses a scatter file instead of the settings from the Target tab, the scatter file must be updated accordingly.



Disabled SoftDevice reserved RAM requirement changes:

The RAM requirement when the SoftDevice is disabled has been increased in order to support interrupt forwarding to be done by the MBR.

- Applications wishing to use the available RAM when the SoftDevice is disabled must not overwrite the first 8 bytes of RAM.

	s210_nrf51422_3.0.0 SoftDevice Disabled RAM requirements	**s210_nrf51422_4.0.0 SoftDevice Disabled RAM requirements**
Size	4 bytes	8 bytes
Application useable RAM start address	0x2000 0004	0x2000 0008

SVC number changes:

A limited set of SVC numbers have changed. The application is required to be re-compiled against the new headers.

`sd_ant_prox_search_set()` now takes an additional argument:

- `uint8_t ucCustomProxThreshold` parameter allows applications to specify a custom minimum RSSI threshold value instead of using predefined ANT indexed values in `uint8_t ucProxThreshold`. The custom value is only applied if the `uint8_t ucProxThreshold` is set with the `PROXIMITY_THRESHOLD_CUSTOM` bit. If the custom proximity field is not used, set it to 0.

Redirecting interrupts to an application from a bootloader has changed:

- `sd_softdevice_forward_to_application()` has been replaced with `sd_softdevice_vector_table_base_set(address)`.

Interrupts can now be directed to anywhere in the application flash area. This also enables using more than one application. See the SoftDevice API documentation for details on how to use this call.

The Radio Disable API functionality has been replaced by the Concurrent Multiprotocol Timeslot API:

The functionality of the previous Radio Disable API, which allowed the application to schedule timeslots of radio inactivity, is now a part of the new Concurrent Multiprotocol Timeslot API feature set.

- `nrf_radio_disable.h` header file removed. Definitions consolidated into `nrf_soc.h`.
- `nrf_radio_request_t` parameter type used in `sd_radio_request()` has been changed.
  - Structure of `nrf_radio_request_t` has changed to support two request types as defined by request_type field: `nrf_radio_request_normal_t` and `nrf_radio_request_earliest_t`.
  - Use `nrf_radio_request_earliest_t` and set `timeout_us = 100000`Linstead of using `distance_us` = 0 in a normal request type).
  - The member `hfclk` replaces `nrf_radio_request_reserved1` and should be set to `NRF_RADIO_HFCLK_CFG_DEFAULT`.
- The `nrf_radio_signal_callback_return_param_t` return parameter type has changed.
  - `return_code` field has been renamed to `callback_action`.

Refer to the SoftDevice API documentation for more details.

New functionality

The SoftDevice hex file no longer contains the SoftDevice size in UICR.CLENR0 register:

The SoftDevice region size is no longer specified in the UICR.CLENR0 in order to allow full flexibility in SoftDevice size changes for Device Firmware Updates. However, memory protection can be enabled during development to detect illegal memory/peripheral accesses.

- If programming the SoftDevice using nRFgo Studio 1.17 or newer, use the “Enable SoftDevice protection” checkbox in “Program SoftDevice” dialog to enable/disable protection.
- If programming using nrfjprog.exe version 5.1.1 or newer, using the `--programs` option will enable protection. Specifying `--programs --dfu` will disable protection.

Multiprotocol Timeslot API features not previously available in Radio Disable API:

The Concurrent Multi-Protocol Timeslot API implementation enables the application to schedule timeslots. During a timeslot the SoftDevice gives control over the RADIO and TIMER0 hardware peripherals to the application. This feature can be used to implement a separate radio protocol in application space that can run concurrently with the SoftDevice protocol, or to schedule timeslots where the SoftDevice is guaranteed to be idle, for example to improve latency for the application, or to reduce peak power consumption.

From the previous Radio Disable API feature, the Multiprotocol Timeslot API introduces more flexible scheduling options that allow applications to:

- Perform radio and timer0 operations during session callback.
- Tail chain radio session requests from a previous session callback.
- Manage ongoing application session activities through the use of session extension requests.

Existing APIs from Radio Disable feature transferred to Concurrent Timeslot API:

- `sd_radio_session_open()`
- `sd_radio_session_close()`
- `sd_radio_request()`

Existing SoC Events transferred to Concurrent Timeslot API:

- `NRF_EVT_RADIO_BLOCKED`
- `NRF_EVT_RADIO_CANCELED`
- `NRF_EVT_RADIO_SIGNAL_CALLBACK_INVALID_RETURN`
- `NRF_EVT_RADIO_SESSION_IDLE`
- `NRF_EVT_RADIO_SESSION_CLOSED`

The following is a list of additions to the Multiprotocol Timeslot API from Radio Disable API:

- Additional `p_radio_signal_callback` types have been added.

    - NRF_RADIO_CALLBACK_SIGNAL_TYPE_TIMER0 - generated whenever NRF_TIMER0 interrupts occur.
    - NRF_RADIO_CALLBACK_SIGNAL_TYPE_RADIO - generated whenever NRF_RADIO interrupts occur.
    - NRF_RADIO_CALLBACK_SIGNAL_TYPE_EXTEND_FAILED - generated whenever session extension has failed.
    - NRF_RADIO_CALLBACK_SIGNAL_TYPE_EXTEND_SUCCEEDED - generated whenever session extension has succeeded
- Additional return types for nrf_radio_signal_callback_return_param_t have been added:
    - NRF_RADIO_SIGNAL_CALLBACK_ACTION_EXTEND - used to request an extension to the current timeslot. Timeslot extension parameters must be specified in extend struct.
    - NRF_RADIO_SIGNAL_CALLBACK_ACTION_REQUEST_AND_END - used to request a new radio timeslot and end the current timeslot. New radio timeslot request parameters must be specified in request struct.

New LFCLK oscillator sources are available:

The following source types have been introduced:

- NRF_CLOCK_LFCLKSRC_RC_250_PPM_TEMP_1000MS_CALIBRATION
- NRF_CLOCK_LFCLKSRC_RC_250_PPM_TEMP_2000MS_CALIBRATION
- NRF_CLOCK_LFCLKSRC_RC_250_PPM_TEMP_4000MS_CALIBRATION
- NRF_CLOCK_LFCLKSRC_RC_250_PPM_TEMP_8000MS_CALIBRATION
- NRF_CLOCK_LFCLKSRC_RC_250_PPM_TEMP_16000MS_CALIBRATION

The new clock source types use the on-chip RC oscillator to generate a 250 PPM signal. Additional power saving is achieved by performing calibration at the specified interval only if the temperature has changed.

Master Boot Record (MBR) API included with SoftDevice:

An MBR API is introduced with the SoftDevice that allows for switching between bootloader(s) and application(s). In addition, the API can be used to replace the bootloader and SoftDevice when performing Device Firmware Updates.

- sd_mbr_command(command)

The following command types are supported:

- SD_MBR_COMMAND_COPY_BL - used to copy a new bootloader into place.
- SD_MBR_COMMAND_COPY_SD - used to copy a new SoftDevice into place.
- SD_MBR_COMMAND_INIT_SD - used to initialize SoftDevice from bootloader and enable interrupt forwarding to it.
- SD_MBR_COMMAND_COMPARE - used to compare flash memory blocks.
- SD_MBR_COMMAND_VECTOR_TABLE_BASE_SET - used to set the address to which the MRB will forward interrupts.

ANT RSSI proximity can now be configured and used in ANT RX scanning channel:

- Specifying ANT proximity settings or custom RSSI values using the sd_ant_prox_search_set() API will apply to the ANT RX scanning channel.

When running an ANT RX scanning channel, received packets that do not meet the specified minimum RSSI threshold will not be sent to the application.

Wildcard IDs can now be used when sending uplink transmission from an ANT RX scanning channel:

Previously, the ID of the target device must be known before uplink transmission from an ANT RX scanning channel can be sent to it. When implementing a system where the scanning channel has to receive and reply to multiple devices with different IDs, having to read and set the received ID prior to sending the reply transmission would result in delayed reply occurring at the next instance of the matching ID received packet.

In order to reduce the reply latency, the application can now assign wildcard (“0”) channel IDs to an ANT transmission channel and by pending transmission on that channel, the RX scanning channel will be able to reply back in the same instance of a received packet. Please note that the transmission channel ID will be assigned to the matching received ID once this has occurred; therefore the ID must be changed back to wildcard if the application is intending to reply back to multiple IDs.

Example:

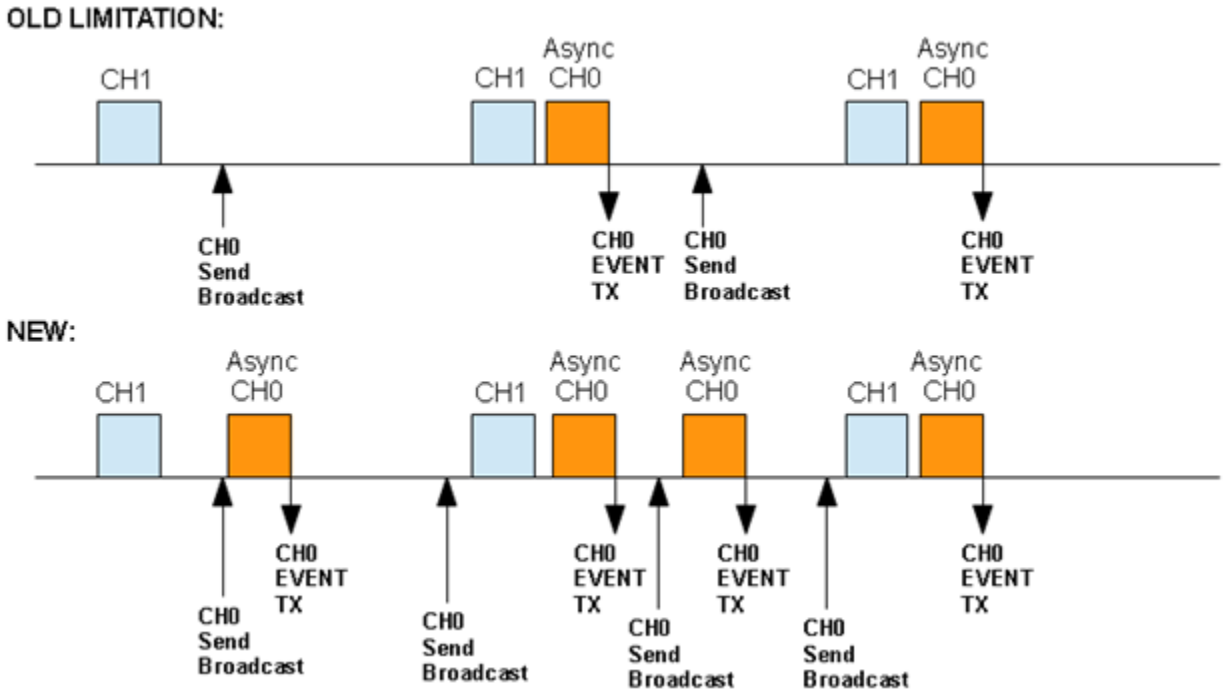
- …Prior channel configuration setup (e.g. RF frequency…etc.).
- sd_ant_channel_id_set (0, 0, 1, 1)
    - Channel 0 as RX scanning channel.
    - Wildcard device ID for RX scanning channel in order to listen for all device IDs.
- sd_ant_channel_id_set (1, 0, 1, 1)

    - Channel 1 used as the RX scanning transmission channel.
    - Wildcard device ID to allow transmission to any device ID packet received from scan.
- `sd_ant_broadcast_message_tx(1,8, buf)`
    - Set pending broadcast transmission on RX scanning transmission channel.
- `sd_ant_rx_scan_mode_start(0)`
    - Open channel RX scanning channel.
    - 0 parameter assigned to not listen for only synchronous RX packets.
- EVENT_RX received from channel 0 in application code.
    - Indicating data received from scanning channel.
- EVENT_TX received from channel 1 in application code.
    - Indicating uplink transmission occurrence of the pending broadcast transmission. Occurs in the same instance as EVENT_RX.
- `sd_ant_channel_id_set (1, 0, 1, 1)`
    - Re-assign RX scanning transmission channel back to wildcard as the channel ID of the last received packet would have been assigned to this channel.
- `sd_ant_broadcast_message_tx(1,8, buf)`
    - Prepare next pending broadcast transmission on RX scanning transmission channel.
- Etc….

ANT asynchronous transmit channel events now occur asynchronously in the presence of other ANT channels.

In previous SoftDevices, when sending a transmission via a channel configured with the EXT_PARAM_ASYNC_TX_MODE extended channel type in the presence of other running ANT channels, the transmission would not occur until after the next scheduled ANT activity has run.

This limitation has now been removed. Asynchronous transmissions in the presence of other running channels will now be performed as soon as possible unless there is insufficient time before the next scheduled activity, which will then cause the transmission to be performed after the previously scheduled ANT activity has run.

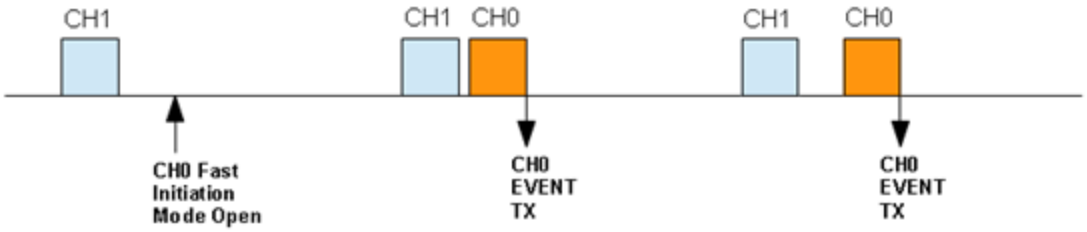


ANT fast initiation channel now start as soon as possible in the presence of other ANT channels.

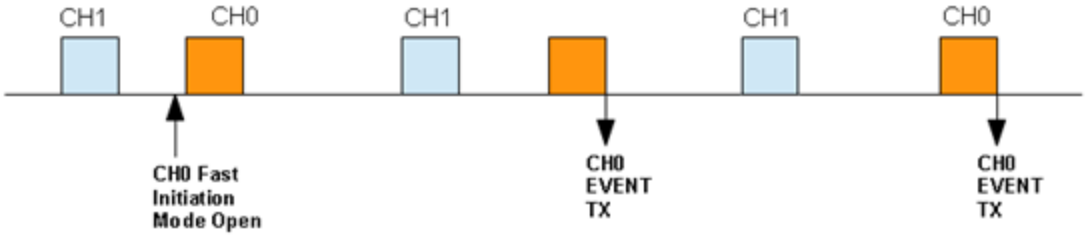
In previous SoftDevices, when opening a channel configured with the EXT_PARAM_FAST_INITATION_MODE extended channel type in the presence of other running ANT channels, the channel would not start until after the next scheduled ANT activity has run.

This limitation has now been removed. Channels configured with fast channel initiation will now start as soon as possible unless there is insufficient time before the next scheduled activity, which will then cause the channel to start after the previously scheduled ANT activity has run.

OLD LIMITATION:



NEW:



Improved ANT RX Scanning Channel operation during application flash write/timeslot activity:

ANT RX scanning channel behaviour has been optimized to use more of the available free time around concurrent application timeslot and flash scheduling activity.