s120_nrf51 migration document

On this page

Introduction to the s120_nrf51 migration document

This document describes how to migrate to new versions of the s120_nrf51. The s120_nrf51 release notes should be read in conjunction with this document.

For each version, we have the following sections:

- "Required changes" describes how an application would have used the previous version of the SoftDevice, and how it must now use this version for the given change.
- "New functionality" describes how to use new features and functionality offered by this version of the SoftDevice. **Note:** Not all new functionality may be covered; the release notes will contain a full list of new features and functionality.

Each section describes how to migrate to a given version from the previous version. If you are migrating to the current version from the previous version, follow the instructions in that section. To migrate between versions that are more than one version apart, follow the migration steps for all intermediate versions in order.

s120_nrf51_2.0.0

Required changes

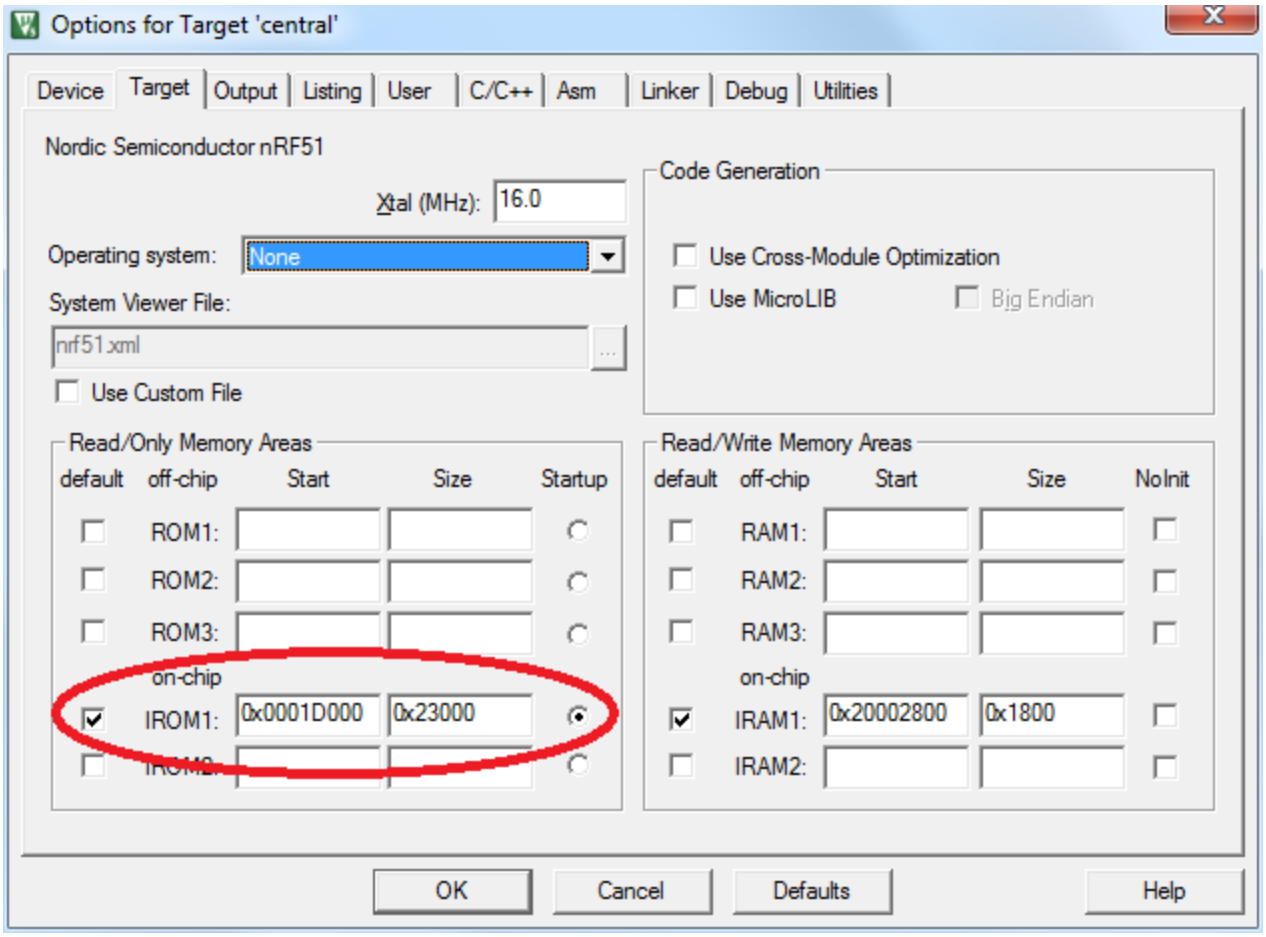
SoftDevice size

The size of the SoftDevice has changed requiring a change to the application project file. The new size (including MBR) is 116 kB.

For Keil this means:

- Go into the properties of the project and find the Target tab
- Change IROM1 Start to `0x1D000`
- Ensure that the IROM1 size is no more than `0x23000` (for the 256 kB IC variants).

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



SVC number changes

The SVC numbers in use by the stack have been changed so the application needs to be recompiled against the new header files.

The following SDM call has been renamed:

The `sd_softdevice_forward_to_application()` SV call to set the vector table base address has been renamed to:

`sd_softdevice_vector_table_base_set(uint32_t address)`

The `NRF_POWER_DCDC_MODES` enumeration has been simplified:

Instead of the previous OFF, ON and AUTOMATIC modes, it can now only be set to `NRF_POWER_DCDC_DISABLE` or `NRF_POWER_DCDC_ENABLE`. This affects the `sd_power_dcdc_mode_set()` SV call. Note that the DC/DC converter is only supported on nRF51 series IC revision 3.

An additional call has to be made after `sd_softdevice_enable()` before any BLE related functionality in the SoftDevice can be used:

- `sd_ble_enable(p_ble_enable_params)`

Using this SV call the application **must** choose a role by filling the `role` field in the `ble_gap_enable_params_t` structure. The choices are:

- `BLE_GAP_ROLE_PERIPH`: The SoftDevice will then be initialized in the peripheral role.
- `BLE_GAP_ROLE_CENTRAL`: The SoftDevice will then be initialized in the central role.

If the application desires to change the role during its execution, it needs to disable the SoftDevice (this will reset the radio and therefore interrupt any ongoing transactions) and re-enable it:

```
sd_softdevice_disable();
sd_softdevice_enable(params);
sd_ble_enable(new_role);
```

Using this new `sd_ble_enable()` SV call, the application can also select whether to include the Service Changed characteristic in the GATT Server. All previous releases of the S120 have included the Service Changed characteristic, but including this characteristic affects how GATT clients behave. Specifically, it requires clients to subscribe to this attribute and prevents them from caching attribute handles between connections unless the devices are bonded. If the application does not need to change the structure of the GATT server attributes at runtime this adds unnecessary complexity to the interaction with peer clients. If the SoftDevice is enabled with the Service Changed Characteristic turned off, then clients are allowed to cache attribute handles without the need to bond, making applications simpler on both sides. Please refer to the SoftDevice API documentation for details.

`sd_ble_gap_address_set()` now takes an additional argument which is used to describe the private address cycle mode:

- `BLE_GAP_ADDR_CYCLE_MODE_NONE`
- `BLE_GAP_ADDR_CYCLE_MODE_AUTO`

To set all types of device addresses explicitly, as with earlier versions of the SoftDevice, use

- `sd_ble_gap_address_set(BLE_GAP_ADDR_CYCLE_MODE_NONE, p_addr)`

To let the SoftDevice automatically cycle private addresses as defined by Bluetooth Core specification 4.1, use

- `sd_ble_gap_address_set(BLE_GAP_ADDR_CYCLE_MODE_AUTO, p_addr)`

where `p_addr->addr_type` is either `BLE_GAP_ADDR_TYPE_RANDOM_PRIVATE_RESOLVABLE` or `BLE_GAP_ADDR_TYPE_RANDOM_PRIVATE_NON_RESOLVABLE`.

`ble_gap_adv_params_t` now contains a additional `channel_mask` field that must be filled in:

When calling `sd_ble_gap_adv_start()`, a pointer to an instance of `ble_gap_adv_params_t` must be provided. This structure now contains an additional field (`channel_mask`) allowing the application to disable specific advertising channels, which must be filled in by the application. To enable all advertising channels simply set this whole field to zero. The SoftDevice behavior will then remain unchanged from previous versions.

The `BLE_GAP_EVT_CONNECTED` event now includes the device's own address:

The new `own_addr` field in the `ble_gap_evt_connected_t` structure allows the application to find out which address was used to initiate a particular connection, which can be useful when using privacy features.

The GATTS set and get operations for local values have changed:

The new function prototypes require a connection handle (since this is required for certain multi-value attributes) and also use a new structure `ble_gatts_value_t` for parameter input:

- `sd_ble_gatts_value_set(uint16_t conn_handle, uint16_t handle, ble_gatts_value_t *p_value)`
- `sd_ble_gatts_value_get(uint16_t conn_handle, uint16_t handle, ble_gatts_value_t *p_value)`

The GATTS set and get operations for system attributes have changed:

The new function prototypes include an additional parameter, `flags`, to allow for partial retrieval or storage of system attributes. If the application does not want to make use of this new functionality, it can simply set the `flags` parameter to 0.

- `sd_ble_gatts_sys_attr_set(uint16_t conn_handle, uint8_t const *p_sys_attr_data, uint16_t len, uint32_t flags)`
- `sd_ble_gatts_sys_attr_get(uint16_t conn_handle, uint8_t *p_sys_attr_data, uint16_t *p_len, uint32_t`

`flags)`

New functionality

A new command has been added to the MBR

The new `SD_MBR_COMMAND_VECTOR_TABLE_BASE_SET` allows the application to set the vector table base so that the MBR forwards all exceptions to the currently active vector table.

A new **Concurrent Multiprotocol Timeslot API has been introduced**

This enables the application to schedule timeslots during which the SoftDevice gives the application control over the RADIO and TIMER0 hardware peripherals. 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.

Three new API calls have been introduced:

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

Five new SoC Events have been introduced:

- `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`

Please see the SoftDevice Specification document and the SoftDevice API documentation for details and guidelines on how to use this feature.

A new option allows the application to enable radio and CPU mutual exclusion:

The new `BLE_COMMON_OPT_RADIO_CPU_MUTEX` option allows applications to enable mutual exclusion between the radio and the CPU.

This mutual exclusion is required to be enabled when running on nRF51 series IC revision 2 (devices affected by PAN #44 "CCM may exceed real time requirements" or PAN #45 "AAR may exceed real time requirements").

Connection RSSI reporting is now available:

The previously unsupported `sd_ble_gap_rssi_start()` and `sd_ble_gap_rssi_stop()` SV calls are now fully functional to use for connection RSSI reporting.

Two new parameters to the `sd_ble_gap_rssi_start()` SV call, `threshold_dbm` and `skip_count`, allow the application to further specify how often and under which conditions it wishes to receive connection RSSI events from the SoftDevice.

The new `sd_ble_gap_rssi_get()` SV call allows applications to poll the connection RSSI of the last connection interval at any time.

The following new options have been added to GAP:

- `BLE_GAP_OPT_LOCAL_CONN_LATENCY`
- `BLE_GAP_OPT_PASSKEY`
- `BLE_GAP_OPT_PRIVACY`
- `BLE_GAP_OPT_SCAN_REQ_REPORT`
- `BLE_GAP_OPT_COMPAT_MODE`

Please refer to the GAP API documentation for further information about their usage.

A new event, **`BLE_GAP_EVT_SCAN_REQ_REPORT`** has been introduced:

This event will only be issued by the SoftDevice when the `BLE_GAP_OPT_SCAN_REQ_REPORT` option has been enabled. The corresponding new `ble_gap_evt_scan_req_report_t` structure is now also part of `ble_gap_evt_t`.

Additional Advertising Data types approved by the SIG have been added to `ble_gap.h`

See the current set of `BLE_GAP_AD_TYPE_*` macros for a full list.

New GAP Appearance values approved by the SIG have been added to `ble_types.h`

See the current set of `BLE_APPEARANCE_*` macros for a full list.

s120_nrf51822_1.0.0

S120 1.0.0 is the first production release of the S120 SoftDevice.