s145 migration document

Introduction to the s145 migration document

About the document

This document describes how to migrate to new versions of the s145 SoftDevice. The s145 release notes should be read in conjunction with this document.

For each version, we have the following sections:

- "Required changes" describes the changes that need to be done in the application when migrating from an older version of the SoftDevice.
- "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.

Example: To migrate from version 5.0.0 to version 5.2.0, first follow the instructions to migrate to 5.1.0 from 5.0.0, then follow the instructions to migrate to 5.2.0 from 5.1.0.

Contents

s145_10.0.0-1.prototype

This section describes how to adapt to changes and use the new features of s145_10.0.0-1.prototype when migrating from s145_9.0.0.

Required changes

Recommended changes

New functionality

s145_9.0.0

This section describes how to adapt to changes and use the new features of s145_9.0.0 when migrating from s145_9.0.0-3.prototype.

Required changes

- The fields `hfclk_latency` and `hfint_ctiv` have been added to `nrf_clock_lf_cfg_t`.

```
static const nrf_clock_lf_cfg_t clock_cfg = {
  .source = NRF_CLOCK_LF_SRC_XTAL,
  .rc_ctiv = 0,
  .rc_temp_ctiv = 0,
  .accuracy = NRF_CLOCK_LF_ACCURACY_250_PPM,
  .hfint_ctiv = 60, /* Time between HFINT calibrations in seconds */
  .hfclk_latency = 1500, /* Startup time of HFXO in microseconds */
};
sd_softdevice_enable(&clock_cfg, fault_handler);
```

s145_9.0.0-3.prototype

This section describes how to adapt to changes and use the new features of s145_9.0.0-3.prototype when migrating from s140_nrf52_7.2.0.

Required changes

- `sd_flash_protect` has been removed. RRAM Immutable Boot Region can be configured in UICR. Access privileges for RRAM regions can be configured using `RRAMC->REGION`. See the RRAM chapter in the datasheet for more information.
- The `nrf_nvic.h` header has been removed and CMSIS functions shall be used instead. The application shall not touch the SoftDevice interrupts while the SoftDevice is enabled. The SoftDevice does not support disabling interrupts globally.

    - Use `NVIC_ClearPendingIRQ` instead of `sd_nvic_ClearPendingIRQ`
    - Use `NVIC_DisableIRQ` instead of `sd_nvic_DisableIRQ`
    - Use `NVIC_EnableIRQ` instead of `sd_nvic_EnableIRQ`
    - Use `NVIC_GetPendingIRQ` instead of `sd_nvic_GetPendingIRQ`
    - Use `NVIC_GetPriority` instead of `sd_nvic_GetPriority`
    - Use `NVIC_SetPendingIRQ` instead of `sd_nvic_SetPendingIRQ`
    - Use `NVIC_SetPriority` instead of `sd_nvic_SetPriority`
    - Use `NVIC_SystemReset` instead of `sd_nvic_SystemReset`

  For critical regions the `BASEPRI` register can be used to disable all interrupts except the SoftDevice high priority interrupts:

```
/* Old API */
uint32_t nested;
sd_nvic_critical_region_enter(&nested);
[...] /* Critical region */
sd_nvic_critical_region_exit(nested);

/* New API */
uint32_t bp = __get_BASEPRI();
__set_BASEPRI_MAX(2U << (8U - __NVIC_PRIO_BITS));
__ISB();
[...] /* Critical region */
__set_BASEPRI(bp);
__ISB();
```

  It is not possible to call SoftDevice APIs while in a critical region.

- The SoftDevice PPI API has been removed. The application should use the `nrfx_dppi` and `nrfx_ppib` drivers or the `nrfx_gppi` helper provided by nrfx. The application shall not touch the PPI resources used by the SoftDevice (see `nrf_sd_def.h`).
- `sd_power_dcdc_mode_set` has been removed and `NRF_REGULATORS->VREGMAIN.DCDCEN` should be used instead.
- `sd_power_system_off` has been removed and `NRF_REGULATORS->SYSTEMOFF` should be used instead.
- `sd_power_reset_reason_get` and `sd_power_reset_reason_clr` have been removed and `NRF_RESET->RESETREAS` should be used instead.

- `sd_power_ram_power_set`, `sd_power_ram_power_get` and `sd_power_ram_power_clr` have been removed and `NRF_MEMCONF->CONTROL` should be used instead.
- It is required to seed the random number generator whenever the `NRF_EVT_RAND_SEED_REQUEST` event is raised. Seeding is done using the new `sd_rand_seed_set` API. Generating random numbers (`sd_rand_application_vector_get`) or enabling Bluetooth (`sd_ble_enable`) will fail if the random number generator has not been seeded.

  The following pseudo code shows how seeding can be done:

```
case NRF_EVT_RAND_SEED_REQUEST:
{
  uint8_t seed[SD_RAND_SEED_SIZE]:

  /* Fetch NIST SP 800-90B compliant entropy */
  trng_get(seed, sizeof(seed));
  sd_rand_seed_set(seed);

  break;
}
```

Recommended changes

- `sd_app_evt_wait` is deprecated and should no longer be used. Instead do the following:

```
/* Wait for an event. */
__WFE();

/* Clear Event Register */
__SEV();
__WFE();
```

- The SoftDevice no longer has pools for secure random numbers and secure random numbers are generated on-demand when `sd_rand_application_vector_get` is called. `sd_rand_application_pool_capacity_get` and `sd_rand_application_bytes_available_get` are deprecated and should not be used.