Porting the Serialization Application – Addendum

3/19/2018

Getting Started

http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Flib_serialization.html provides an overview of the BLE Serialization application.

Before beginning a porting effort, we recommended that you follow the instructions provided here http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Fnrf51_setups_serialization.html&cp=4_0_1_1_4 to create a complete nRF based test and validation platform based on your desired serial interface. This will allow the initial code modifications and host connectivity to be easily validated.

IMPORTANT NOTE: When creating the test setup, the SoftDevice should not be programmed into the host DK. It is only required for the connectivity DK.

Once you have validated the example application and hardware setup, it is a good idea to capture the data exchange between them to assist in future validation and debugging of your target platform.

Porting the Serialization Application – Addendum

3/19/2018

Initial Porting of Host Application

The basic porting process is described here: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Fserialization_porting_guide.html. However, it is important to note that if your application requires bonding, `nrf_nvmc.c` must also be ported.

Though applicable to all porting efforts, the remainder of this document is based on experience with:

1) Two nRF52832-DKs
2) The connectivity processor example
   `…\SDK_12.2.0_f012efa\examples\ble_central_and_peripheral\ble_connectivety\pca10040\ser_s132_spi`
3) The host processor example
   `…\SDK_12.2.0_f012efa\examples\ble_peripheral\ble_app_hts\pca10040\ser_s132_spi`

IMPORTANT NOTE: The very first step in porting the code is to remove nRF dependent SRAM routines from the host processor's application.

Use the following modification for conditional `#ifdef` statements.

To remove SRAM dependent routines from the host application, `softdevice_handler.c` must be modified to remove the functions `ram_total_size_get()`, `ram_end_address_get()`, and `sd_check_ram_start()`.

Also within `softdevice_handler.c`, two modifications are required to the `softdevice_enable()` function:

1) Remove the call to `ram_end_address_get()`. This call is with a NRF_LOG_WARNING message.
2) Change `app_ram_base = ram_start;` to `app_ram_base = 0;`

Once the above steps have been completed, recompile and retest the test platform. There should be no functional changes.

Porting the Serialization Application – Addendum

3/19/2018

Porting the Connectivity Application

Porting of the connectivity application is dependent on the target hardware.

I/O Pins

If the connectivity nRF device does not use the same the I/O lines as your target, reconfigure the connection between the host nRF DK and the connectivity nRF DK. Then change the I/O pins as needed within the connectivity application.

The pin assignments are found in pca10040.h. For example:

```
#define SER_CON_SPIS_SCK_PIN   27  // SPI SCK signal
#define SER_CON_SPIS_MOSI_PIN  2   // SPI MOSI signal
#define SER_CON_SPIS_MISO_PIN  26  // SPI MISO signal
#define SER_CON_SPIS_CSN_PIN   23  // SPI CSN signal
#define SER_CON_SPIS_RDY_PIN   25  // SPI READY GPIO pin number
#define SER_CON_SPIS_REQ_PIN   24  // SPI REQUEST GPIO pin number
```

After changing the I/O pins configuration, now would be a good time to recompile and test that the example functionality has not changed.

Low Frequency Clock

The example connectivity application depends on the nRF DK's external 32 KHz crystal.

If the target hardware does not include an external 32 KHz crystal, `CLOCK_CONFIG_LF_SRC` must be changed to the appropriate source. This can be done using the configuration wizard or within `sdk_config.h`.

Low Power DC/DC

For the lowest possible power consumption, both the external 32 KHz crystal and the internal DC/DC converter should be used.

If the target uses the DC\DC and has the proper external components installed (for example: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Fref_circuitry.html&cp=2_2_0_52_1&anchor=schematic_qfn48_dcdc), enable the DC/DC converter with a SoftDevice call after initializing the SoftDevice.

Within `main.c, main()`:

```
SOFTDEVICE_HANDLER_INIT(&clock_lf_cfg, NULL);  // after this line
sd_power_dcdc_mode_set(NRF_POWER_DCDC_ENABLE); // add this line
```

Porting the Serialization Application – Addendum

3/19/2018

Completing the Application Port

Once the procedures in this document have been followed, the application porting process can begin.

Important Note: Before beginning a host example porting effort, carefully consider if the targeted host process has DMA and how the DMA interface may differ from the Nordic EasyDMA feature. For this reason, it may be best to start with an example that does not have EasyDMA enabled.

The remaining key application components that need to be ported are described here: http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.2.0%2Fserialization_porting_guide.html.

It is up to the developer to perform any porting required for their development environment and into their specific application. If possible, it is recommended that the Nordic provided host application serve as the starting point for the final application

Change Log:

3/24/17	Added Change log. Added **Important Note** in the **Completing the Application Port** section.