**Getting Started**

<https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v12.2.0/lib_serialization.html?resultof=%22%73%65%72%69%61%6c%69%7a%61%74%69%6f%6e%22%20%22%73%65%72%69%61%6c%22%20%22%6f%76%65%72%76%69%65%77%22%20> provides an overview of the BLE Serialization application.

Before beginning a porting effort, we recommended that you follow the instructions provided here <https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v12.2.0/nrf51_setups_serialization.html> 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.

**Initial Porting of Host Application**

The basic porting process is described here: <https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v12.2.0/serialization_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 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, 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

**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: <https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v12.2.0/serialization_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

4/10/2019 Fixed links … SFB