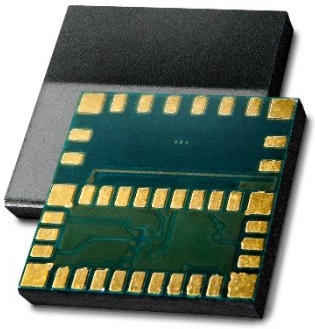
Application Note AN171102

ISP1302-BN Serialization User Guide



Introduction

Scope

This application note describes how to set up BLE serialization between an ISP1302-BN module and a main application CPU.

Contents

Insight SiP– Green Side – 400 avenue Roumanille – BP 309 – 06906 Sophia-Antipolis Cedex – France –www.insightsip.com

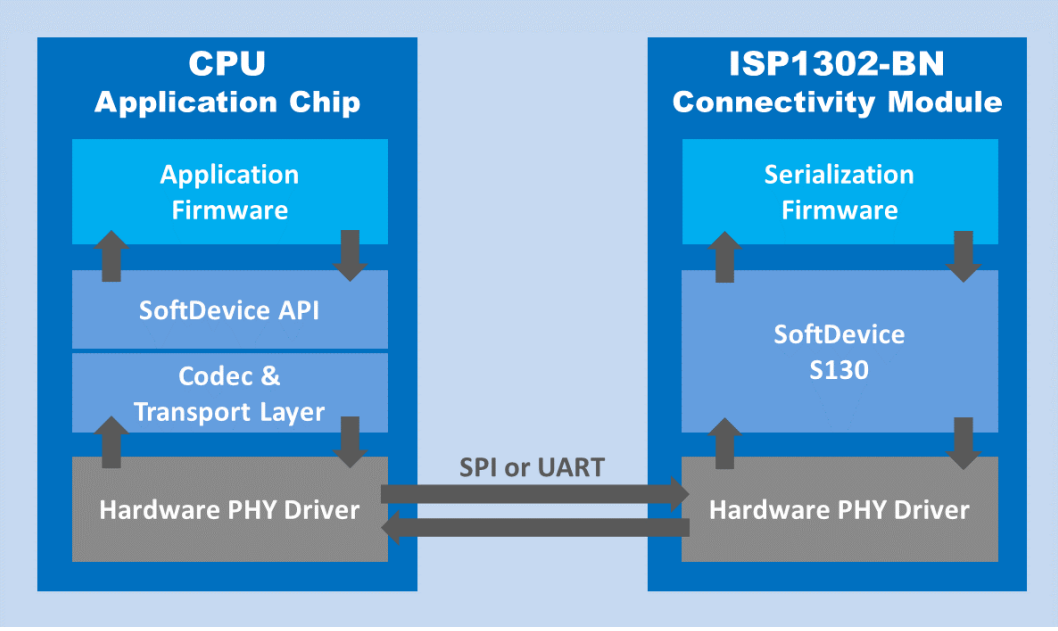
1. Generalities

1.1. Architectural Overview

Some applications need the addition of a BLE connectivity while developers want to keep their system architecture based on a specific CPU. Other applications are using BLE SoftDevice that cannot be ported to the ISP1302, for example because they use specific peripherals or need more resources like RAM, flash memory, or CPU speed.

In this case using the ISP1302-BN module preprogrammed with the Nordic BLE Serialization API may be the solution. Serialization makes it possible to place a Bluetooth application on an application module and connect it to a connectivity module that runs the SoftDevice.

The serialization libraries and the connectivity example simplify the serialization of an existing application, because only limited modifications are needed in the application itself.



1.2. Application module

The application module runs a serialized application, where the SoftDevice is replaced by a commands encoder and events decoder.

In this Application Note, a standard ISP1302-BM is used here merely as a demonstration device. After porting the hardware driver to the selected PHY layer, one can use a different microcontroller.

To port serialization libraries to another microcontroller, refer to the Nordic Infocenter:
http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v11.0.0%2Fserialization_porting_guide.html

The application chip does not need a SoftDevice. It is replaced by a Codec that implements the SoftDevice API.

All function calls to the Codec are serialized and transmitted to the connectivity chip using the transport layer drivers (UART or SPI).

1.3. Connectivity module

The connectivity module is an ISP1302-BN that is pre-programmed with a SoftDevice. It decodes serialized SoftDevice commands from the application chip and issues the corresponding call to the SoftDevice.

Any event from the SoftDevice is encoded by the Codec that implements the SoftDevice API. Through the transport layer, it is then transmitted to the application chip, where it is decoded and passed to the application.

In addition to the SoftDevice, the connectivity chip must be programmed with the connectivity software.

2. Hardware Configuration

2.1. Module Pin Description

Pin	Name	Pin function	Description
1	VSS	Ground	Should be connected to ground plane on application PCB
2	P0_24	Digital I/O	UART RXD
3	P0_21	Digital I/O	UART TXD
4	P0_22	Digital I/O	UART RTS
5	VSS	Ground	Should be connected to ground plane on application PCB
6	OUT_ANT	Antenna I/O	Must be connected to Pin 7 OUT_MOD for normal operation
7	OUT_MOD	Antenna I/O	Must be connected to Pin 6 OUT_ANT for normal operation
8	VSS	Ground	Should be connected to ground plane on application PCB
9	AVDD	Power	Should be connected to VCC in LDO mode, to 1.8V in low voltage mode; or should be connected to Pin 24 DCC through loading inductors and capacitors in DC/DC mode
10	NC	Not Connected	Not used
11	NC	Not Connected	Not used
12	NC	Not Connected	Not used
13	P0_15	Digital I/O	SPI/nUART - SPI active high / UART active low
14	P0_20	Digital I/O	SPI CSN Slave Select
15	VSS	Ground	Should be connected to ground plane on application PCB
16	nRESET	Digital I/O	System reset active low
17	P0_12	Digital I/O	SPI SCK Clock
18	NC	Not Connected	Not used
19	VSS	Ground	Should be connected to ground plane on application PCB
20	P0_06	Digital I/O	SPI MOSI Master Out Slave In
21	P0_03	Digital I/O	SPI MISO Master In Slave Out
22	P0_02	Digital I/O	SPI READY
23	P0_00	Digital I/O	SPI REQUEST
24	DCC	Power	Should not be connected in LDO or low voltage mode Should be connected to Pin 9 AVDD through loading inductors and capacitors in DC/DC mode
25	VCC_nRF	Power	Power supply (1.8 – 3.6V)
26	XL1	Analog input	Optional Crystal connection for 32.768 kHz crystal oscillator or external 32.768 kHz crystal reference
27	P0_23	Digital I/O	UART CTS
28	XL2	Analog input	Optional Crystal connection for 32.768 kHz crystal oscillator
29 to 40	NC	Not Connected	Isolated pad for mechanical stability Do not connect

2.2. SPI Connection Parameters

ISP1302-BN module is using the following SPI parameters. Please adjust SPI parameters on the Application chip accordingly:

- Mode 0 (CPOL= 0 and CPHA= 0)
- LSB First
- Connection speed 8 M bauds maximum

2.3. UART Connection Parameters

ISP1302-BN module is using the following UART parameters. Please adjust UART parameters on the Application chip accordingly:

- Flow Control: Enabled
- Parity: 1 bit
- Connection speed 1 M bauds

Insight SiP– Green Side – 400 avenue Roumanille – BP 309 – 06906 Sophia-Antipolis Cedex – France –www.insightsip.com

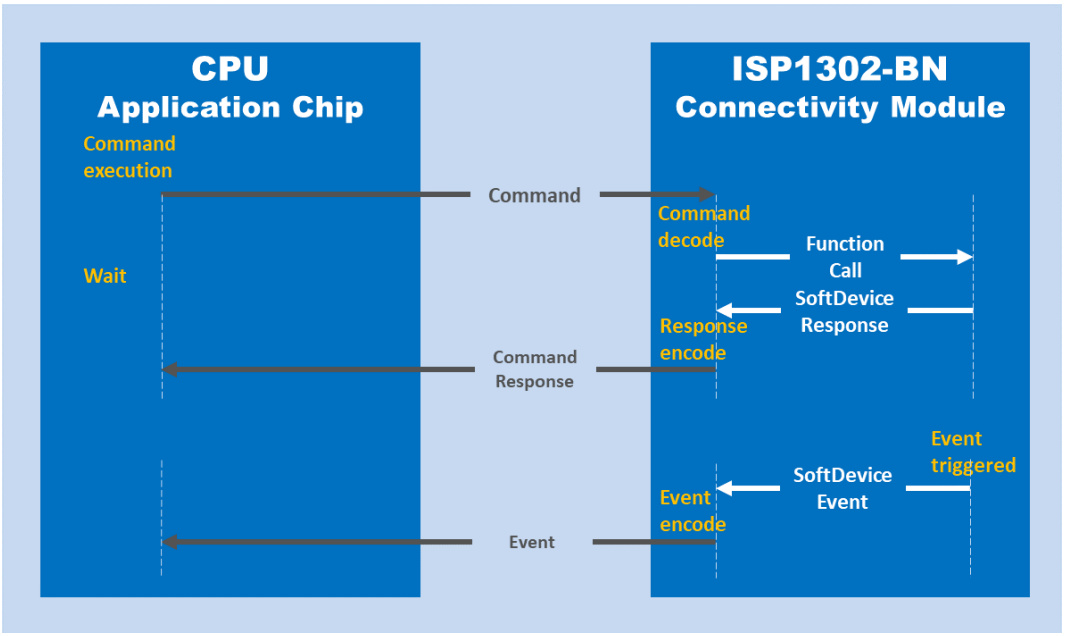
3. Software Functions and Events

Please refer to Nordic Infocenter nRF5 SDK v11.0.0 documentation for detailed information on Command and Events packet format:
http://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v11.0.0/ser_codecs_130.html

3.1. Packet Encoding Format

All frames have the following format:

Command, Response and Event packet course:



Packet Type		Description
0x00	Command	The packet is sent from the application chip to the BLE Connectivity Chip, where it will be decoded and the corresponding function in the SoftDevice is executed.
0x01	Command Response	After a function in the SoftDevice is received, the response is encoded in the BLE Connectivity Chip and a response packet is sent to the application chip.
0x02	Event	If an event is triggered in the SoftDevice, an event packet is is sent from the BLE Connectivity Chip to the application chip.

3.2. Command List

Operation Code		Description
0x3C	sd_power_system_off	Puts the chip in System OFF mode
0x4D	sd_ecb_block_encrypt	Encrypts a block according to the specified parameters
0x53	sd_temp_get	Get the temperature measured on the chip
0x60	sd_ble_enable	Enable the BLE stack
0x62	sd_ble_tx_packet_count_get	Get the total number of available guaranteed application transmission packets for a particular connection
0x63	sd_ble_uuid_vs_add	Add a Vendor Specific UUID
0x64	sd_ble_uuid_decode	Decode little endian raw UUID bytes (16-bit or 128-bit) into a 24-bit ble_uuid_t structure
0x65	sd_ble_uuid_encode	Encode a ble_uuid_t structure into little endian raw UUID bytes (16-bit or 128-bit)
0x66	sd_ble_version_get	Get Version Information
0x67	sd_ble_user_mem_reply	Provide a user memory block
0x68	sd_ble_opt_set	Set a BLE option
0x69	sd_ble_opt_get	Get a BLE option
0x70	sd_ble_gap_address_set	Set local Bluetooth address
0x71	sd_ble_gap_address_get	Get local Bluetooth address
0x72	sd_ble_gap_adv_data_set	Set, clear or update advertising and scan response data
0x73	sd_ble_gap_adv_start	Start advertising (GAP Discoverable, Connectable modes, Broadcast Procedure)
0x74	sd_ble_gap_adv_stop	Stop advertising (GAP Discoverable, Connectable modes, Broadcast Procedure)
0x75	sd_ble_gap_conn_param_update	Update connection parameters
0x76	sd_ble_gap_disconnect	Disconnect (GAP Link Termination)
0x77	sd_ble_gap_tx_power_set	Set the radio's transmit power
0x78	sd_ble_gap_appearance_set	Set GAP Appearance value
0x79	sd_ble_gap_appearance_get	Get GAP Appearance value
0x7A	sd_ble_gap_ppcp_set	Set GAP Peripheral Preferred Connection Parameters
0x7B	sd_ble_gap_ppcp_get	Get GAP Peripheral Preferred Connection Parameters
0x7C	sd_ble_gap_device_name_set	Set GAP device name
0x7D	sd_ble_gap_device_name_get	Get GAP device name
0x7E	sd_ble_gap_authenticate	Initiate the GAP Authentication procedure
0x7F	sd_ble_gap_sec_params_reply	Reply with GAP security parameters
0x80	sd_ble_gap_auth_key_reply	Reply with an authentication key
0x81	sd_ble_gap_lesc_dhkey_reply	Reply with an LE Secure connections DHKey
0x82	sd_ble_gap_keypress_notify	Notify the peer of a local keypress
0x83	sd_ble_gap_lesc_oob_data_get	Generate a set of OOB data to send to a peer out of band

Operation Code		Description
0x84	sd_ble_gap_lesc_oob_data_set	Provide the OOB data sent/received out of band
0x85	sd_ble_gap_encrypt	Initiate GAP Encryption procedure
0x86	sd_ble_gap_sec_info_reply	Reply with GAP security information
0x87	sd_ble_gap_conn_sec_get	Get the current connection security
0x88	sd_ble_gap_rssi_start	Start reporting the received signal strength to the application
0x89	sd_ble_gap_rssi_stop	Stop reporting the received signal strength
0x8A	sd_ble_gap_scan_start	Start scanning (GAP Discovery procedure, Observer Procedure)
0x8B	sd_ble_gap_scan_stop	Stop scanning (GAP Discovery procedure, Observer Procedure)
0x8C	sd_ble_gap_connect	Create a connection (GAP Link Establishment)
0x8D	sd_ble_gap_connect_cancel	Cancel a connection establishment
0x8E	sd_ble_gap_rssi_get	Get the received signal strength for the last connection event
0x90	sd_ble_gattc_primary_services_discover	Initiate or continue a GATT Primary Service Discovery procedure
0x91	sd_ble_gattc_relationships_discover	Initiate or continue a GATT Relationship Discovery procedure
0x92	sd_ble_gattc_characteristics_discover	Initiate or continue a GATT Characteristic Discovery procedure
0x93	sd_ble_gattc_descriptors_discover	Initiate or continue a GATT Characteristic Descriptor Discovery procedure
0x94	sd_ble_gattc_attr_info_discover	Discovers information about a range of attributes on a GATT server
0x95	sd_ble_gattc_char_value_by_uuid_read	Initiate or continue a GATT Read using Characteristic UUID procedure
0x96	sd_ble_gattc_read	Initiate or continue a GATT Read (Long) Characteristic or Descriptor procedure
0x97	sd_ble_gattc_char_values_read	Initiate a GATT Read Multiple Characteristic Values procedure
0x98	sd_ble_gattc_write	Perform a Write (Characteristic Value or Descriptor, with or without response, signed or not, long or reliable) procedure
0x99	sd_ble_gattc_hv_confirm	Send a Handle Value Confirmation to the GATT Server
0xA0	sd_ble_gatts_service_add	Add a service declaration to the Attribute Table
0xA1	sd_ble_gatts_include_add	Add an include declaration to the Attribute Table
0xA2	sd_ble_gatts_characteristic_add	Add a characteristic declaration, a characteristic value declaration and optional characteristic descriptor declarations to the Attribute Table
0xA3	sd_ble_gatts_descriptor_add	Add a descriptor to the Attribute Table
0xA4	sd_ble_gatts_value_set	Set the value of a given attribute
0xA5	sd_ble_gatts_value_get	Get the value of a given attribute
0xA6	sd_ble_gatts_hvx	Notify or Indicate an attribute value

Operation Code		Description
0xA7	sd_ble_gatts_service_changed	Indicate the Service Changed attribute value
0xA8	sd_ble_gatts_rw_authorize_reply	Respond to a Read/Write authorization request
0xA9	sd_ble_gatts_sys_attr_set	Update persistent system attribute information
0xAA	sd_ble_gatts_sys_attr_get	Retrieve persistent system attribute information from the stack
0xAB	sd_ble_gatts_initial_user_handle_get	Retrieve the first valid user attribute handle
0xAC	sd_ble_gatts_attr_get	Retrieve the attribute UUID and/or metadata
0xB0	sd_ble_l2cap_cid_register	Register a CID with L2CAP
0xB1	sd_ble_l2cap_cid_unregister	Unregister a CID with L2CAP
0xB2	sd_ble_l2cap_tx	Transmit an L2CAP packet

3.3. Event List

Event Code		Description
0x01	BLE_EVT_TX_COMPLETE	Common BLE Event base. Transmission Complete
0x02	BLE_EVT_USER_MEM_REQUEST	User Memory request
0x03	BLE_EVT_USER_MEM_RELEASE	User Memory release
0x10	BLE_GAP_EVT_CONNECTED	GAP BLE Event base. Connection established
0x11	BLE_GAP_EVT_DISCONNECTED	Disconnected from peer
0x12	BLE_GAP_EVT_CONN_PARAM_UPDATE	Connection Parameters updated
0x13	BLE_GAP_EVT_SEC_PARAMS_REQUEST	Request to provide security parameters
0x14	BLE_GAP_EVT_SEC_INFO_REQUEST	Request to provide security information
0x15	BLE_GAP_EVT_PASSKEY_DISPLAY	Request to display a passkey to the user
0x16	BLE_GAP_EVT_KEY_PRESSED	Notification of a keypress on the remote device
0x17	BLE_GAP_EVT_AUTH_KEY_REQUEST	Request to provide an authentication key
0x18	BLE_GAP_EVT_LESC_DHKEY_REQUEST	Request to calculate an LE Secure Connections DHKey
0x19	BLE_GAP_EVT_AUTH_STATUS	Authentication procedure completed with status
0x1A	BLE_GAP_EVT_CONN_SEC_UPDATE	Connection security updated
0x1B	BLE_GAP_EVT_TIMEOUT	Timeout expired
0x1C	BLE_GAP_EVT_RSSI_CHANGED	RSSI report
0x1D	BLE_GAP_EVT_ADV_REPORT	Advertising report
0x1E	BLE_GAP_EVT_SEC_REQUEST	Security Request
0x1F	BLE_GAP_EVT_CONN_PARAM_UPDATE_REQUEST	Connection Parameter Update Request
0x20	BLE_GAP_EVT_SCAN_REQ_REPORT	Scan request report
0x30	BLE_GATTC_EVT_PRIM_SRVC_DISC_RSP	GATTC BLE Event base. Primary Service Discovery Response event
0x31	BLE_GATTC_EVT_REL_DISC_RSP	Relationship Discovery Response event
0x32	BLE_GATTC_EVT_CHAR_DISC_RSP	Characteristic Discovery Response event
0x33	BLE_GATTC_EVT_DESC_DISC_RSP	Descriptor Discovery Response event
0x34	BLE_GATTC_EVT_ATTR_INFO_DISC_RSP	Attribute Information Response event
0x35	BLE_GATTC_EVT_CHAR_VAL_BY_UUID_READ_RSP	Read by UUID Response event
0x36	BLE_GATTC_EVT_READ_RSP	Read Response event

Event Code		Description
0x37	BLE_GATTC_EVT_CHAR_VALS_READ_RSP	Read multiple Response event
0x38	BLE_GATTC_EVT_WRITE_RSP	Write Response event
0x39	BLE_GATTC_EVT_HVX	Handle Value Notification or Indication event
0x3A	BLE_GATTC_EVT_TIMEOUT	Timeout event
0x50	BLE_GATTS_EVT_WRITE	GATTS BLE Event base. Write operation performed
0x51	BLE_GATTS_EVT_RW_AUTHORIZE_REQUEST	Read/Write Authorization request
0x52	BLE_GATTS_EVT_SYS_ATTR_MISSING	A persistent system attribute access is pending
0x53	BLE_GATTS_EVT_HVC	Handle Value Confirmation
0x54	BLE_GATTS_EVT_SC_CONFIRM	Service Changed Confirmation. No additional event structure applies
0x55	BLE_GATTS_EVT_TIMEOUT	Peer failed to respond to an ATT request in time
0x70	BLE_L2CAP_EVT_RX	L2CAP BLE Event base. L2CAP packet received

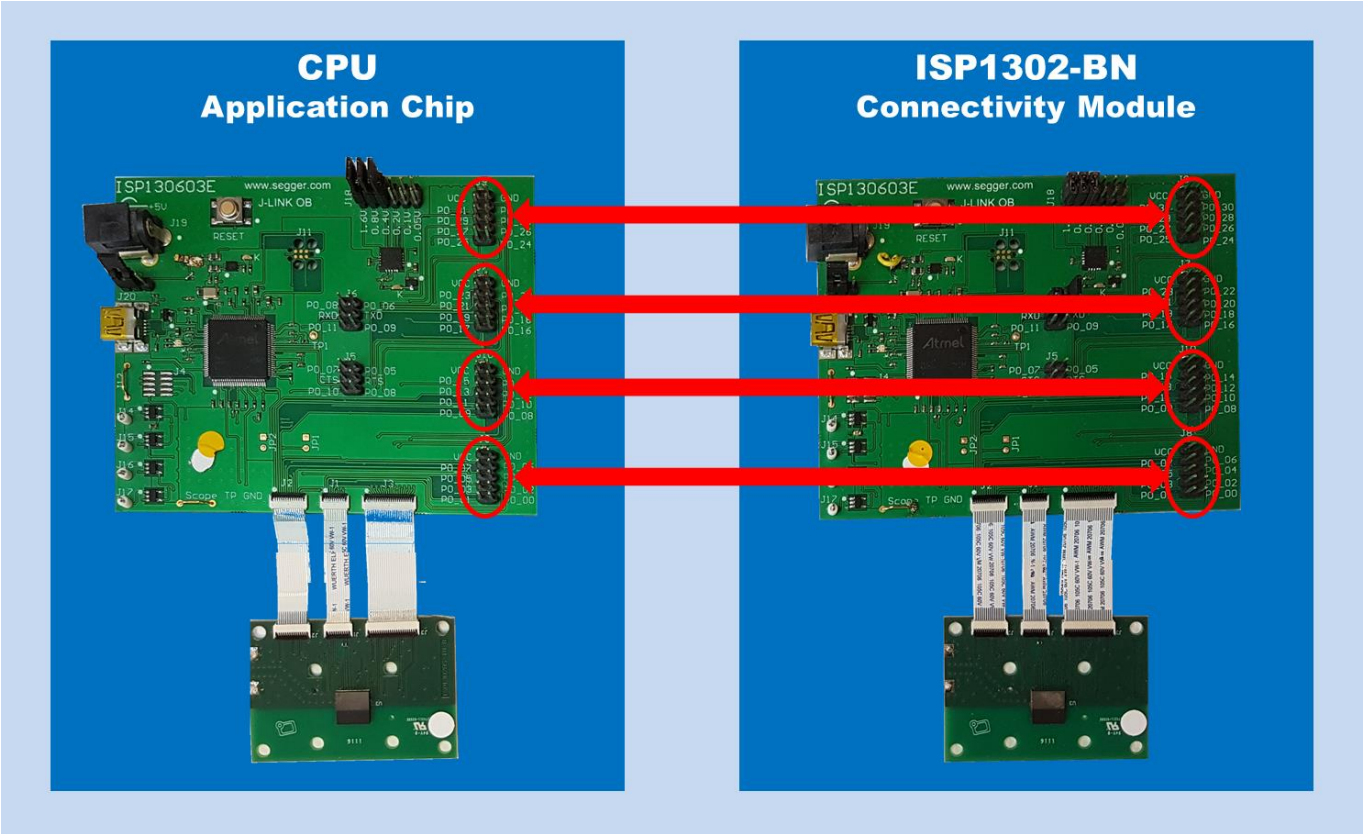
Insight SiP– Green Side – 400 avenue Roumanille – BP 309 – 06906 Sophia-Antipolis Cedex – France –www.insightsip.com

4. Serialization Example

4.1. Hardware Setup

The serialization setup supports two physical transport interfaces for BLE: UART and SPI.

The following figure illustrates how to connect two test boards (with their interface board) as an application board and a connectivity board supplying an SPI or UART connection.



For SPI interface, connect all the following pins on the highlighted connectors:
- GND, P0_00, P0_02, P0_03, P0_06, P0_12 and P0_20
- P0_15 to VCC

For UART interface, connect all the following pins on the highlighted connectors:
- GND, P0_21, P0_22, P0_23 and P0_24
- P0_15 to GND

4.2. Example with SPI Interface

Software Compliance
The following example uses the nRF5 SDK v11.0.0 and the SoftDevice S130 v2.0.0.

Connectivity Module Side
Apply VCC to P0_15 pin SPI/nUART to select SPI mode.

Application Module Side
The application module does not need a SoftDevice. Prepare the application module by performing the following steps:

1. Connect application module by performing the following steps.

2. In Keil, open one of the serialized example projects. The serialized version is located in the ser_s130_spi folder. Choose the example project for the same physical transport layer as on the connectivity board.

Example	Physical transport layers
Alert Notification Application	UART, SPI,
Beacon Transmitter Sample Application	UART, SPI
Blood Pressure Application	UART, SPI
Cycling Speed and Cadence Application	UART, SPI
Glucose Application	SPI
HID Keyboard Application	SPI
Heart Rate Application	SPI
Health Thermometer Application	UART, SPI
Power Profiling Application	UART, SPI
Running Speed and Cadence Application	UART, SPI
Apple Notification Center Service (ANCS) Client Application	SPI
Direct Test Mode	UART, SPI
BLE Heart Rate Collector Example	SPI
BLE Multi-link Example	UART, SPI

In this example, we choose the Heart Rate Application with SPI.

3. In this example the application board is an ISP1302 test board so we need to modify the board definition file to change the pinout.

```
#define SER_APP_SPIM0_SCK_PIN     12    // SPI clock GPIO pin number.
#define SER_APP_SPIM0_MOSI_PIN    06    // SPI Master Out Slave In GPIO pin number
#define SER_APP_SPIM0_MISO_PIN    03    // SPI Master In Slave Out GPIO pin number
#define SER_APP_SPIM0_SS_PIN      20    // SPI Slave Select GPIO pin number
#define SER_APP_SPIM0_RDY_PIN     02    // SPI READY GPIO pin number
#define SER_APP_SPIM0_REQ_PIN     00    // SPI REQUEST GPIO pin number
```

   Note: Normally the reset pin should also be configured (SER_CONN_CHIP_RESET_PIN). This pin should be connected to the SWDIO-nRESET pin of the connectivity module. We don't use it in this example.

4. In this example the application module is an ISP1302 so we need to modify the serialization library in order to set low frequency source to RC (it uses by default the XTAL clock source). In the file ser_app_hal_nrf51.c, modify the function ser_app_hal_hw_init in order to set the clock source (see below).

```
uint32_t ser_app_hal_hw_init(ser_app_hal_flash_op_done_handler_t handler)
{
  nrf_gpio_cfg_output(CONN_CHIP_RESET_PIN_NO);

  NRF_CLOCK->LFCLKSRC = (CLOCK_LFCLKSRC_SRC_RC << CLOCK_LFCLKSRC_SRC_Pos);
  NRF_CLOCK->EVENTS_LFCLKSTARTED = 0;
  NRF_CLOCK->TASKS_LFCLKSTART    = 1;

  while (NRF_CLOCK->EVENTS_LFCLKSTARTED == 0)
  {
    //No implementation needed.
  }


  NRF_CLOCK->EVENTS_LFCLKSTARTED = 0;
  m_flash_op_handler = handler;
  return NRF_SUCCESS;
}
```

5. Compile the application and download the created.hex file to the application module.

Verification

Power-up both modules at the same time. Check if you can connect with your smartphone using nRF Toolbox or nRF Connect) to the device called Nordic_HRM.

4.3. Example with UART Interface

Software Compliance
The following example uses the nRF5 SDK v11.0.0 and the SoftDevice S130 v2.0.0.

Connectivity Module Side
Select UART mode by connecting P0_15 pin SPI/nUART to GND.

Application Module Side
The application module does not need a SoftDevice. Prepare the application module by performing the following steps:

1. Connect application module by performing the following steps.

2. In Keil, open one of the serialized example projects. The serialized version is located in the ser_s130_uart folder. Choose the example project for the same physical transport layer as on the connectivity board.

Example	Physical transport layers
Alert Notification Application	UART, SPI
Beacon Transmitter Sample Application	UART, SPI
Blood Pressure Application	UART, SPI
Cycling Speed and Cadence Application	UART, SPI
Glucose Application	SPI
HID Keyboard Application	SPI
Heart Rate Application	SPI
Health Thermometer Application	UART, SPI
Power Profiling Application	UART, SPI
Running Speed and Cadence Application	UART, SPI
Apple Notification Center Service (ANCS) Client Application	SPI
Direct Test Mode	UART, SPI
BLE Heart Rate Collector Example	SPI
BLE Multi-link Example	UART, SPI

In this example, we choose the Heart Rate Application with UART.

3. In this example the application board is an ISP1302 test board so we need to modify the board definition file to change the pinout.

```
#define SER_APP_RX_PIN          21    // UART RX pin number.
#define SER_APP_TX_PIN          22    // UART TX pin number.
#define SER_APP_CTS_PIN         23     // UART Clear To Send pin number.
#define SER_APP_RTS_PIN         24    // UART Request To Send pin number.
```

   Note: Normally the reset pin should also be configured (SER_CONN_CHIP_RESET_PIN). This pin should be connected to the SWDIO-nRESET pin of the connectivity module. We don't use it in this example.

4. In this example the application module is an ISP1302 so we need to modify the serialization library in order to set low frequency source to RC (it uses by default the XTAL clock source). In the file ser_app_hal_nrf51.c, modify the function ser_app_hal_hw_init in order to set the clock source (see below).

```
uint32_t ser_app_hal_hw_init(ser_app_hal_flash_op_done_handler_t handler)
{
   nrf_gpio_cfg_output(CONN_CHIP_RESET_PIN_NO);

   NRF_CLOCK->LFCLKSRC = (CLOCK_LFCLKSRC_SRC_RC << CLOCK_LFCLKSRC_SRC_Pos);
   NRF_CLOCK->EVENTS_LFCLKSTARTED = 0;
   NRF_CLOCK->TASKS_LFCLKSTART    = 1;

   while (NRF_CLOCK->EVENTS_LFCLKSTARTED == 0)
   {
      //No implementation needed.
   }


   NRF_CLOCK->EVENTS_LFCLKSTARTED = 0;
   m_flash_op_handler = handler;
   return NRF_SUCCESS;
}
```

5. Compile the application and download the created.hex file to the application module.

Verification

Power-up both modules at the same time. Check if you can connect with your smartphone using nRF Toolbox or nRF Connect) to the device called Nordic_HRM.