Confidential

Nordic SDK introduction

Nordic *Bluetooth* low energy hands-on training

NORDIC SEMICONDUCTOR

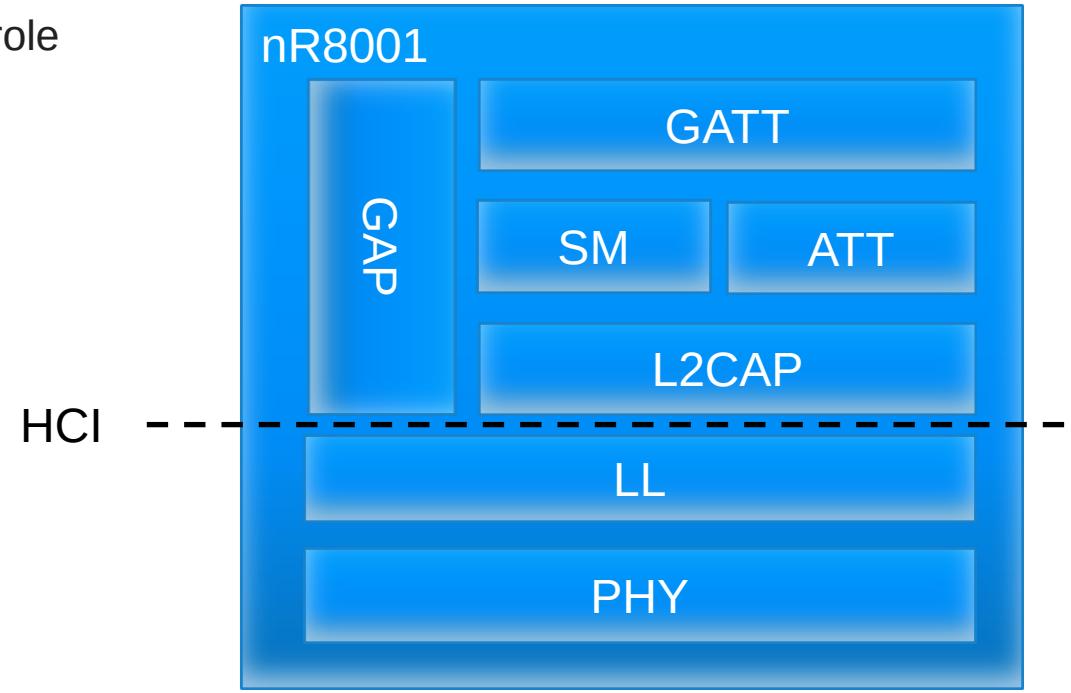
nRF8001

Single-chip *Bluetooth* low energy peripheral device

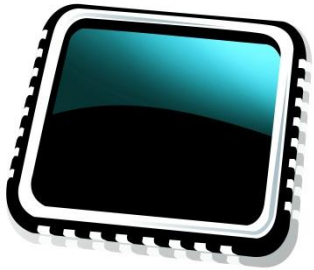
NORDIC SEMICONDUCTOR

nRF8001

- Operates in the peripheral role
- Integrated Host stack
- Integrated Link Layer
- Qualified radio



The concept



Existing application

(Serial interface)

Added Bluetooth connectivity

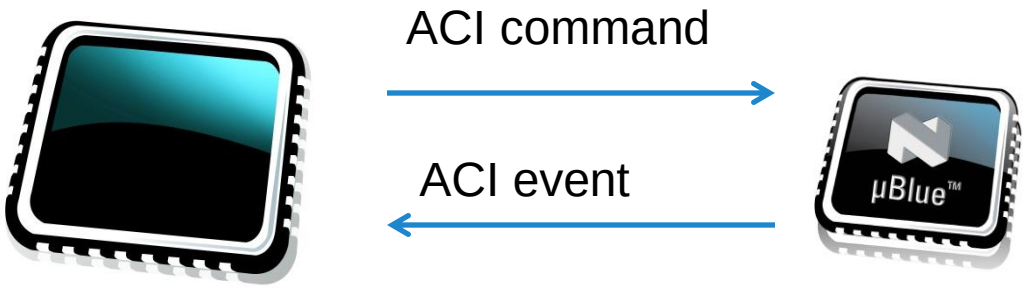
nRF8001 advantages

- Simple to add to an existing platform
- Qualified stack: QDL can be reused
- Low and predictable power consumption

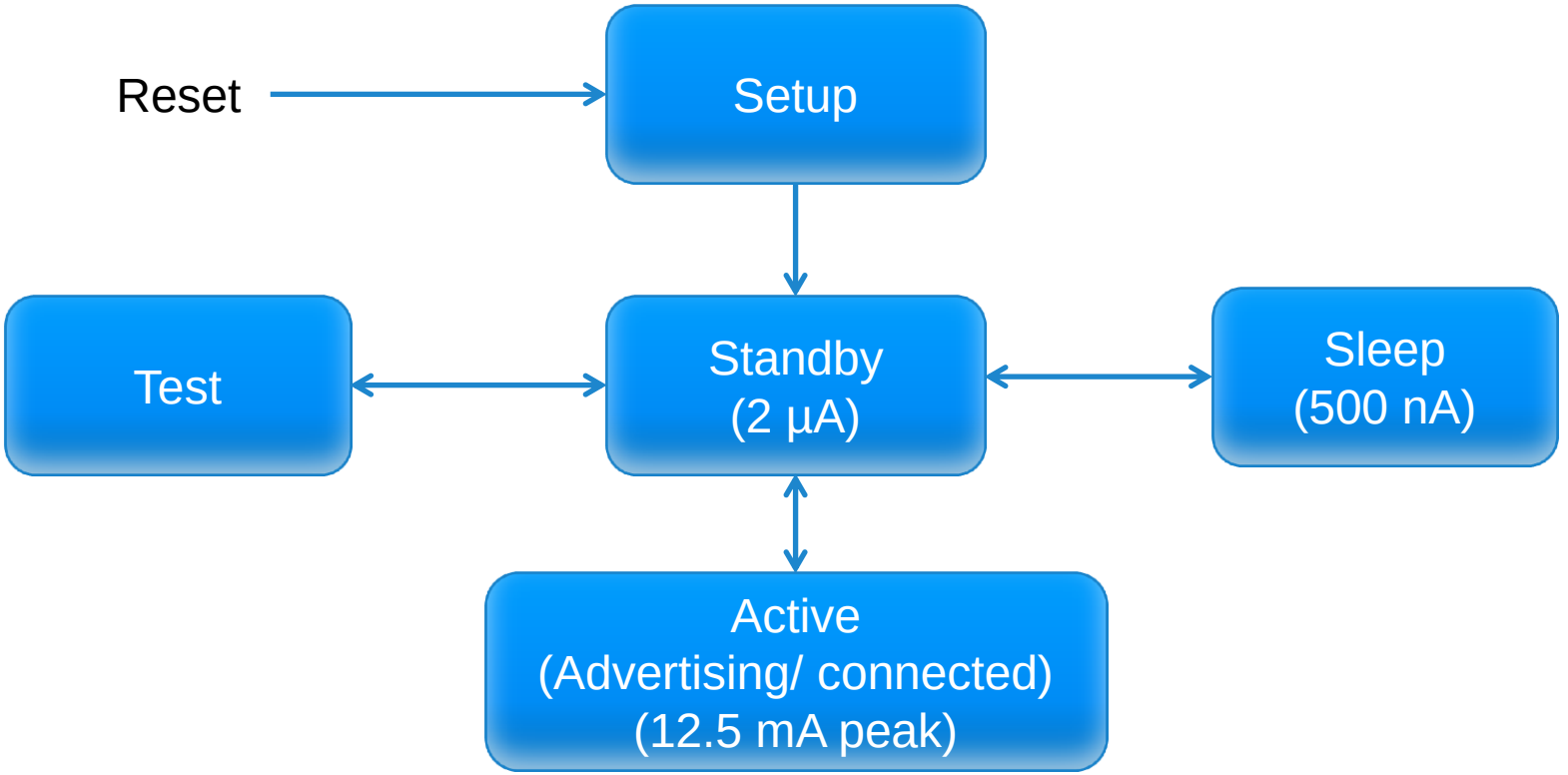
Application Controller Interface (ACI) physical part

- Physical part:
  - 5 IO lines
  - SPI slave: SCK, MISO, MOSI
  - Hand-shake signals: REQN, RDYN
- Operations:
  - Send an ACI command
  - Receive an ACI event
  - The two above combined
- Driver is delivered with the SDK.

Application Controller Interface (ACI) protocol



nRF8001 behavior



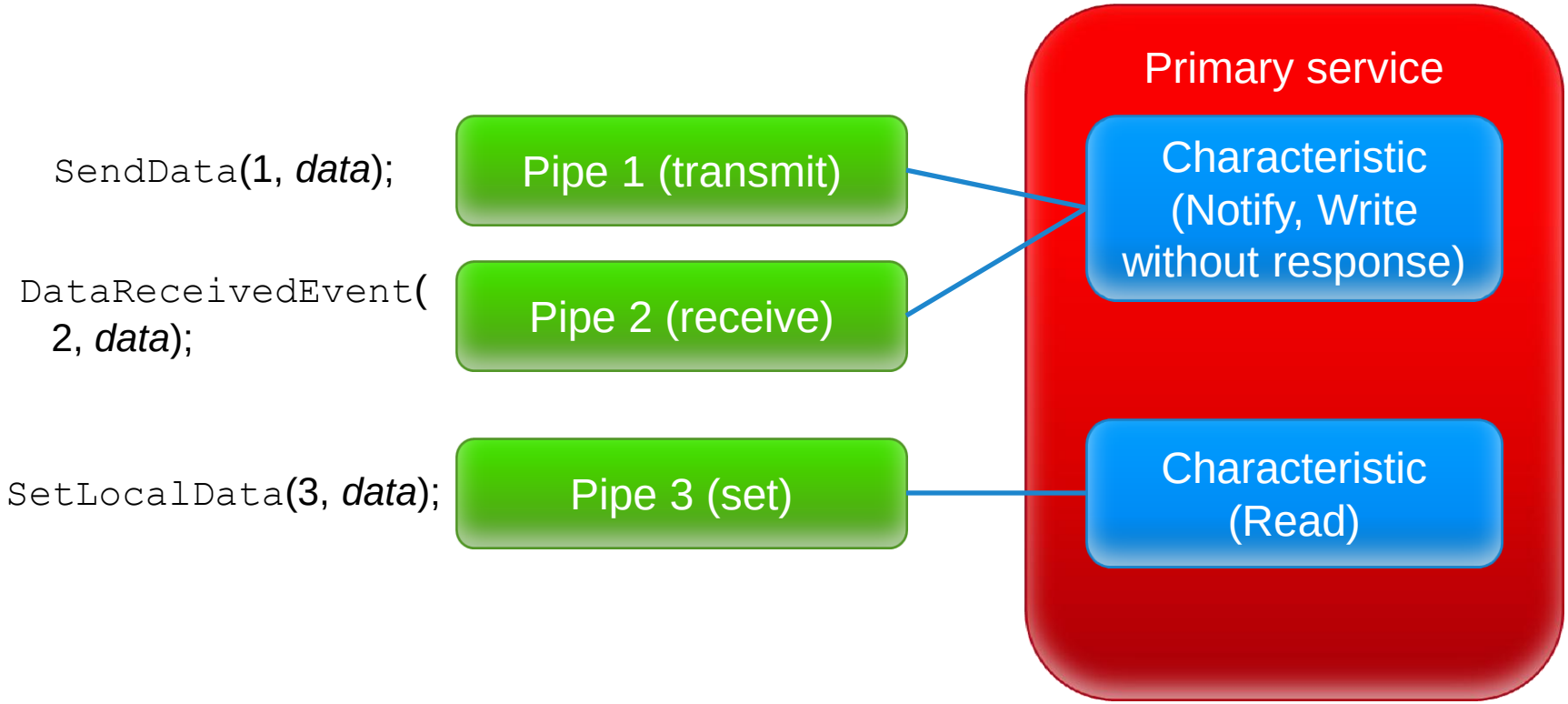
System commands and events

Packet	Usage
DeviceStartedEvent	Received when the device has started and is ready for use.
CommandResponseEvent	Received in response to system commands.
Sleep	Sets the radio in Sleep state (0.5 µA).
Wakeup	Wakes up the IC from Sleep state.
Setup	Writes configuration data to the IC.
RadioReset	Resets the radio.
HardwareErrorEvent	Received in case of failure.

Connection commands and events

Packet	Usage
Connect	Requests a connection
ConnectedEvent	Received when a connection is established.
Bond	Initiates the bonding procedure.
BondStatusEvent	Received when the bonding procedure succeeded.
Disconnect	Disconnect from the peer device.
DisconnectedEvent	Received when disconnected.
TimingEvent	Received when timing parameters are updated.
OpenRemotePipe	Open a pipe that maps to a remote characteristic.
PipeStatusEvent	Received when the pipe availability has changed.

Service pipes

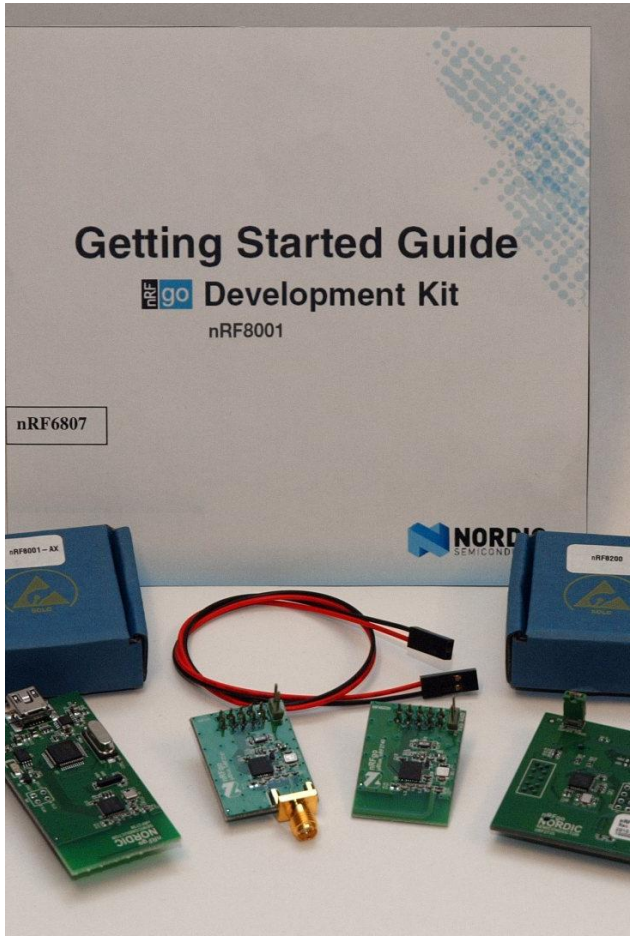


Data transfer commands and events

Packet	Usage
SendData	Sends data over a pipe.
RequestData	Requests data from a pipe.
SendDataAck	ACKs received data.
SetLocalData	Sets the characteristic value in the local attribute server.
DataReceivedEvent	Data received from a peer device.
DataCreditEvent	New credit available.
DataAckEvent	ACK received from the peer device.
PipeErrorEvent	Received if there was a transmission error.

Test commands and events

Packet	Usage
Test	Enter and exit test mode
Echo	Sends data and returns them in an EchoEvent. Useful for testing the ACI driver.
EchoEvent	Response from an Echo command.
DtmCommand	Send DTM packets. Used in Test mode.



nRF8001 Development Kit

Short overview

NORDIC SEMICONDUCTOR

Kit contents

Kit contents

- **nRF8001 modules: PCB antenna (1) and SMA connector (1)**
- **Carrier board with nRF8200 application processor**
- **Bluetooth low energy master emulator**
- Cable
- Samples (5 + 5)

nRFgo Starter Kit

- nRFgo Starter Kit is a generic platform for development kits by Nordic Semiconductor.
- Needed by the nRF8001 DK.
- The essential part: nRFgo Motherboard:
  - Needed for programming the nRF8200
  - Debugger for nRF8200
  - Power supply with configurable voltage

nRF8001 Software Development Kit

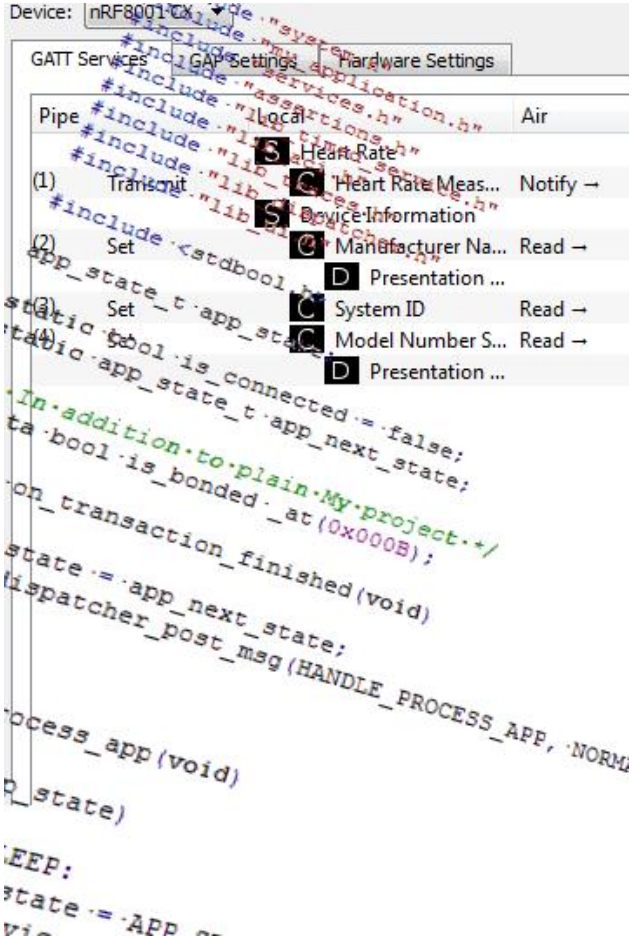
- Source code
  - Libraries
  - Examples
  - Documentation
- nRFgo Studio
  - nRF8001 configuration
  - Flash programming
- Master Control Panel
  - GUI for the master emulator

Compiler tools

- Sample code is based on Keil PK51:
  - Not included with the kit
  - PK51 is a tool for 8051 processors
- Other compilers can be used:
  - Source code is written in C
  - Precompiled HEX files are available in the kit

Demo: check the hardware

- nRF8001 SDK, nRFgo Studio, Master Control Panel are installed
- Setup necessary hardware
- Program Proximity Application
- Run



Development

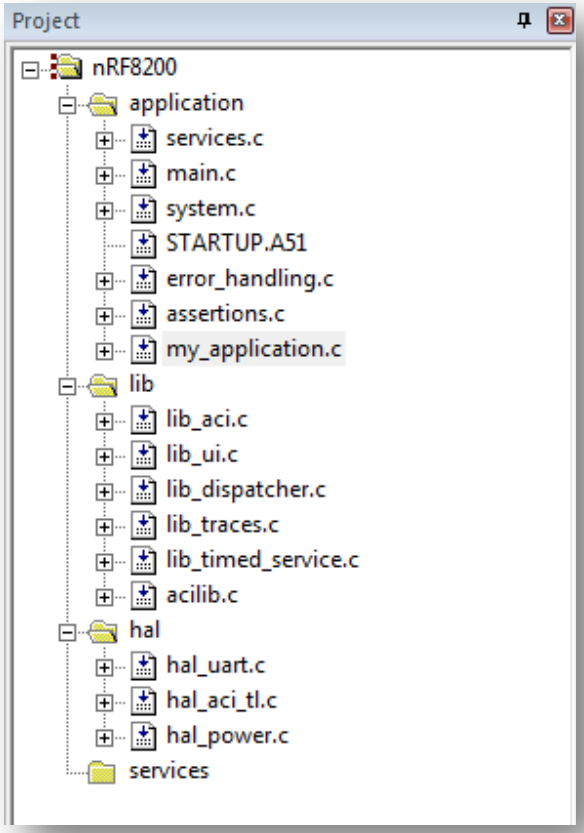
How to develop applications and implement profiles

NORDIC SEMICONDUCTOR

What is the product?

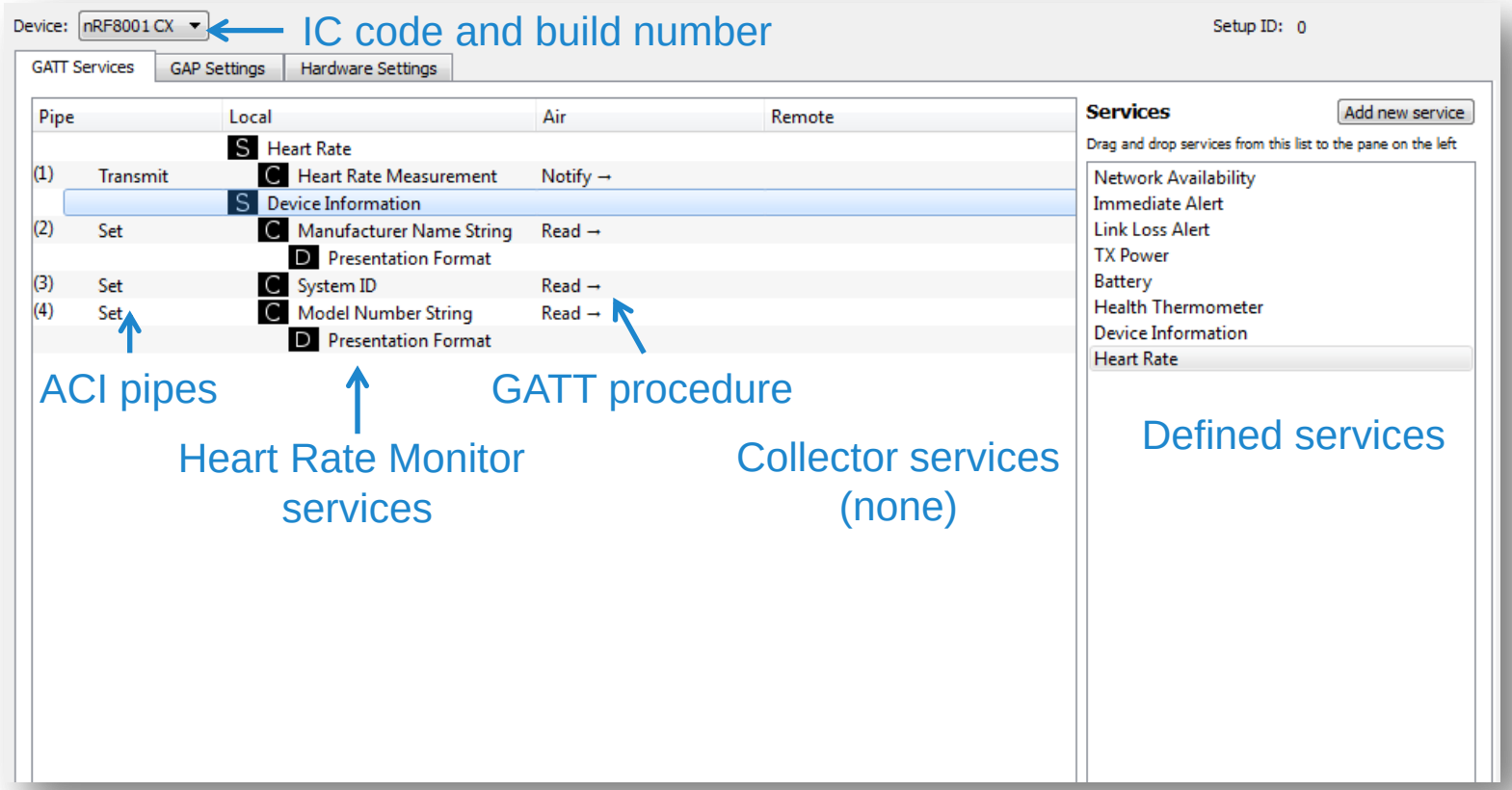
- Example: Heart Rate Profile
  - Defines roles: Heart Rate Sensor and Collector
  - Defines services that shall be used
  - Defines GAP requirements
- With nRF8001 you can design a Heart Rate Sensor:
  - “The Heart Rate Sensor shall use the GAP peripheral role.”
  - nRF8001 will be the GATT server
  - Profile behavior is on the Collector side

Start from an existing project



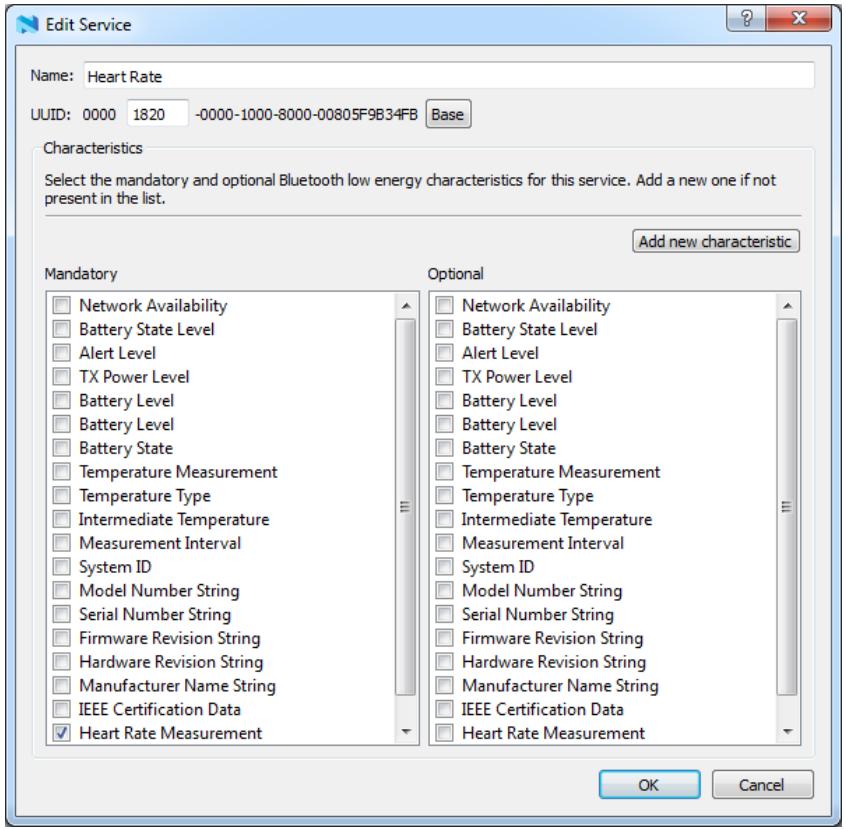
- "My project":
  - Handles the nRF8001 states
  - No services
  - Based on the dispatcher library
  - Includes timer library
  - Includes trace library
- What is needed to make it a Heart Rate Sensor project?
  - Rename files and folders
  - Add services
  - Add and change the code

Configuring services and characteristics



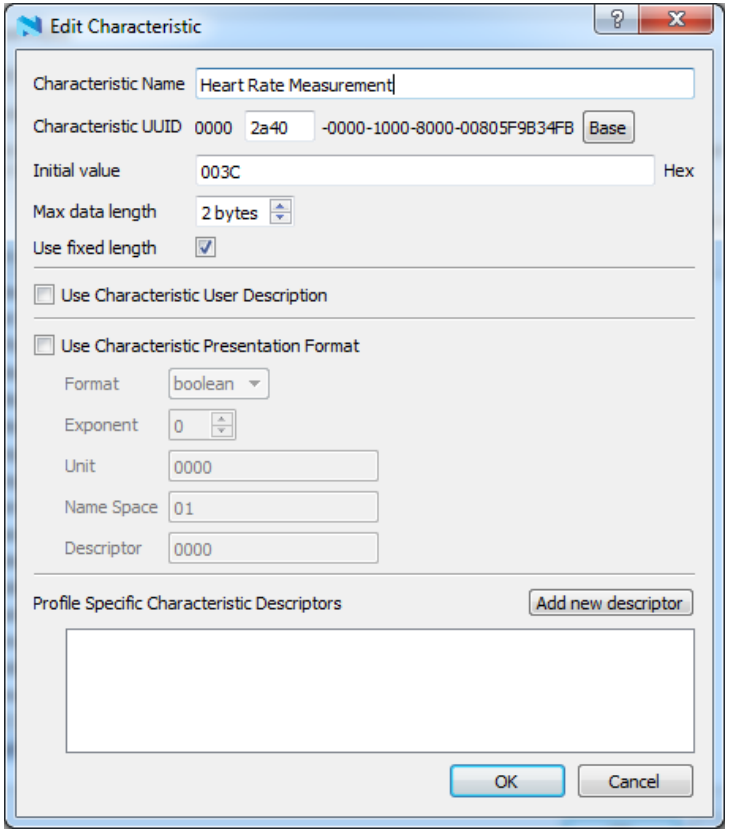
Defining GATT services

- Click **Add new service**:
  - Enter the name of the service.
  - Enter the service UUID.
  - Select mandatory and optional characteristics.
- Click Add new characteristic if the needed characteristic is not in the list.

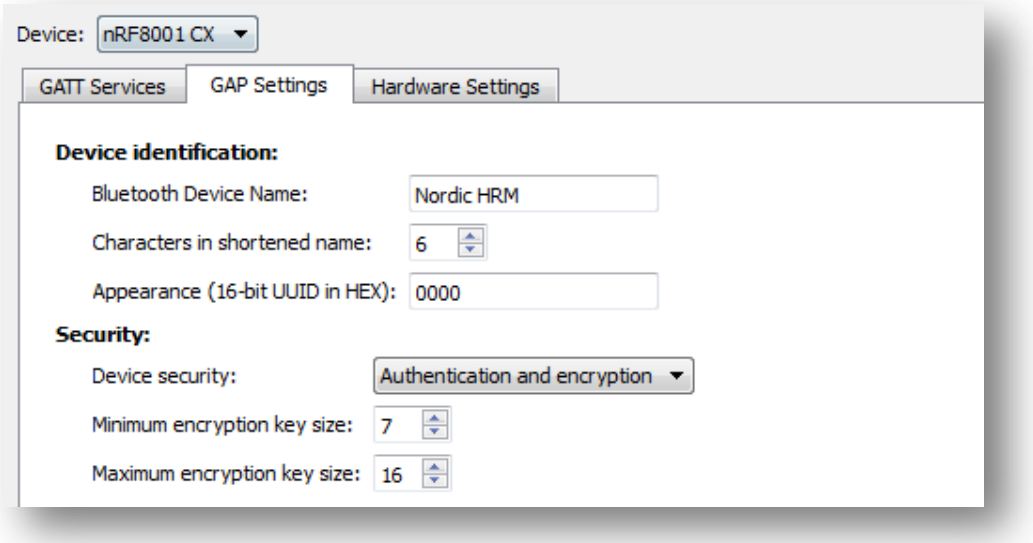


Defining characteristics

- Enter the name and UUID of the characteristic
- Enter initial value and data length
- Choose if the characteristic should include a Characteristic User Description
- Choose if the characteristic should include a Characteristic User Presentation format

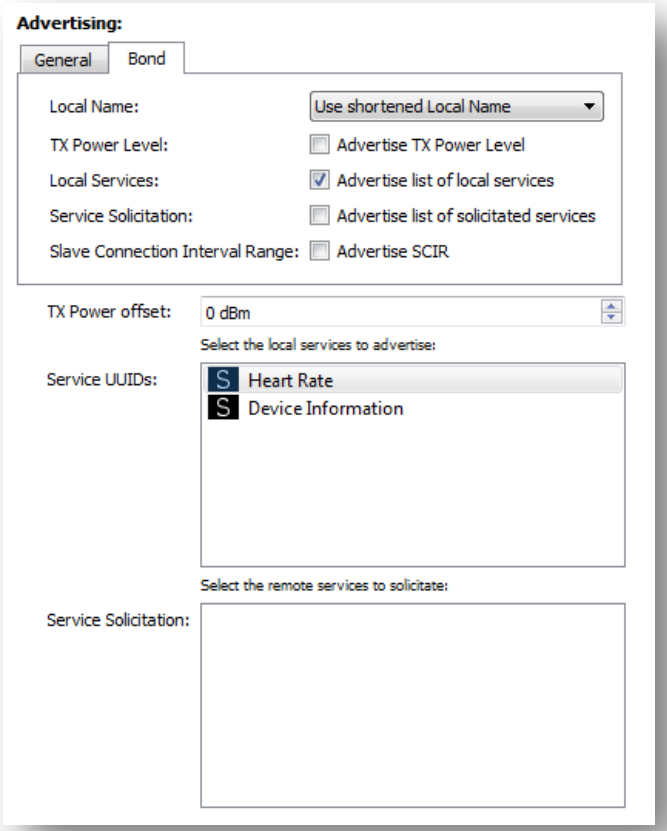


Defining GAP behavior



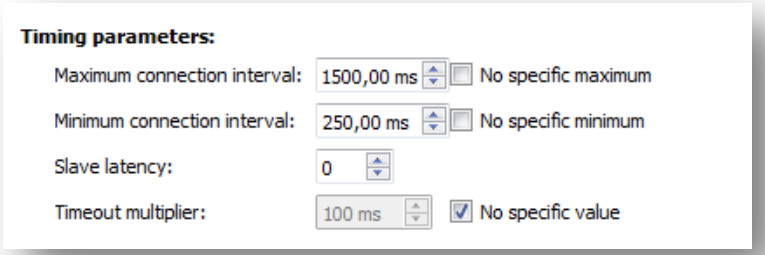
Defining GAP behavior

- Advertising
  - Select which AD types to advertise.
  - The data is reused, for example Device Name, available services, TX power.
  - Independent settings for general mode and bond mode (Limited connectable mode in GAP)



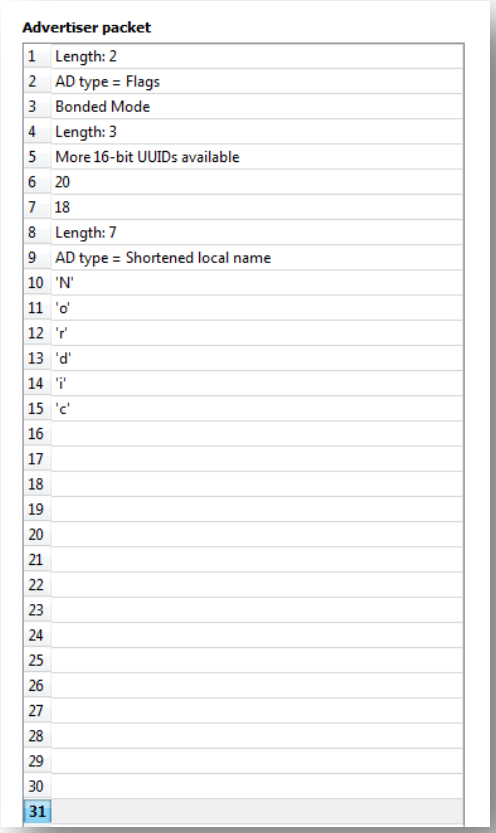
Defining GAP behavior

- Set the Preferred Peripheral Connection Parameter (GAP characteristic)
- Remember: These are just preferences, the master will set the connection interval.
- The application can reissue the request and reject a connection based on the connection interval
- Battery life!



Defining GAP behavior

- The data in the advertiser packet is displayed to the right
- View depends on the which of the General or Bond tabs is selected



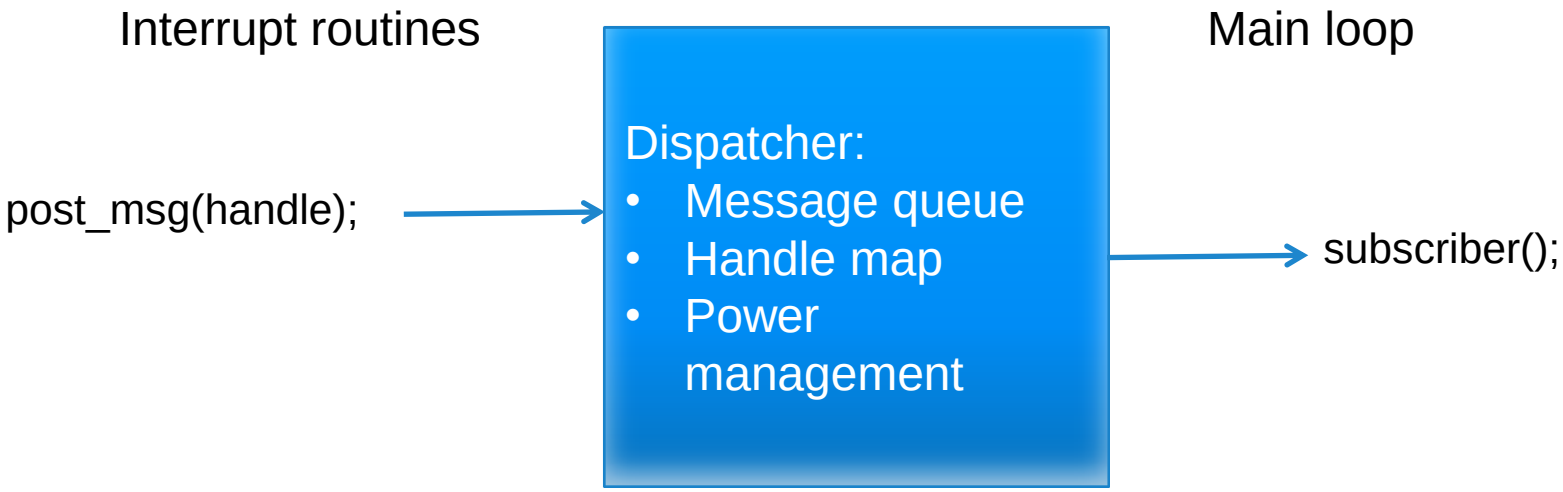
Generate setup files

- GATT Setup → **Generate Source Files**
- Generates three files:
  - services.h: Header files with setup codes and function declarations.
  - services.c: Auto-generated helper functions for the characteristics.
  - services_lock.h: Can replace services.h to make the setup *permanent*!

The “dispatcher”

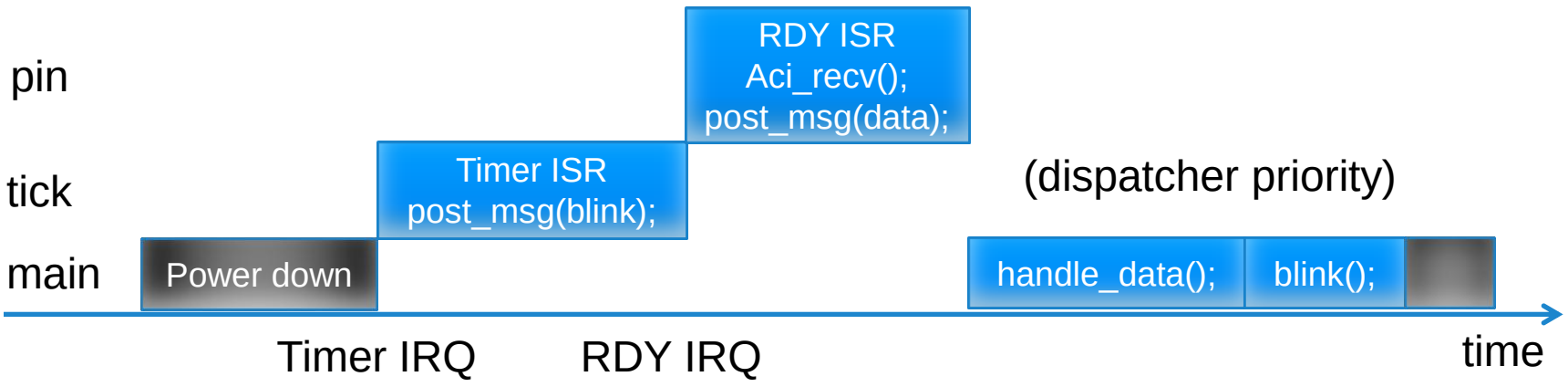
- The SDK examples are based on the dispatcher library:
  - Scheduler for the main vector.
  - Designed for low power.
  - Can be replaced by an RTOS.
- Offered support:
  - C implementation for nRF8200 (8051).
  - Setup tool in nRFgo Studio.
  - Documentation.

Dispatcher (1)



Dispatcher (2)

Vectors:



Setup the Dispatcher

- Can load from and save to dispatcher_config.h.
- Add handles and subscribers for new functionality.
- Some handles (HANDLE_EVT*) are already defined in lib_aci.
- dispatcher_config.h is included by lib_dispatcher.

Dispatcher tool

Subscriber with parameters

	Handle	Subscriber 0	Subscriber 1	Subscriber 2
0	HANDLE_EVT_RADIO_STARTED_STANDBY	on_radio_started_stdby(void)	NULL	NULL
1	HANDLE_EVT_RADIO_STARTED_SETUP	on_radio_started_setup(void)	NULL	NULL
2	HANDLE_EVT_RADIO_PIPES_CONNECTED	on_connect_pipe_ready_event(void)	NULL	NULL
3	HANDLE_EVT_RADIO_PIPES_DISCONNECTED	on_disconnect_event(void)	NULL	NULL
4	HANDLE_EVT_ERROR	radio_evt_error(void)	NULL	NULL
5	HANDLE_EVT_RADIO_DATA_RECEIVED	on_pipe_updated(void)	NULL	NULL
6	HANDLE_EVT_RADIO_ADVERTISE_TIMEDOUT	on_advertise_timedout(void)	NULL	NULL
7	HANDLE_EVT_RADIO_DEVICE_VERSION	on_device_version(void)	NULL	NULL
8	HANDLE_EVT_RADIO_CMD_RESPONSE	NULL	NULL	NULL
9	HANDLE_TRANSACTION_FINISHED	on_transaction_finished(void)	NULL	NULL
10	HANDLE_EVT_RADIO_DEBUG_INFO	on_radio_evt_debug_info(void)	NULL	NULL

Code!

- Start from an example:
  - My project (my_project) provides a clean start.
- Add code:
  - Add subscriber functions for new handles.
  - Add application functionality.
- Change code:
  - Add the Bond command if using authentication, for example.

Traces

- Library for sending 3-byte codes over UART
- Decoded in nRFgo Studio
- Activated and deactivated using compile options