

Confidential

Nordic SDK introduction

Nordic *Bluetooth* low energy hands-on training



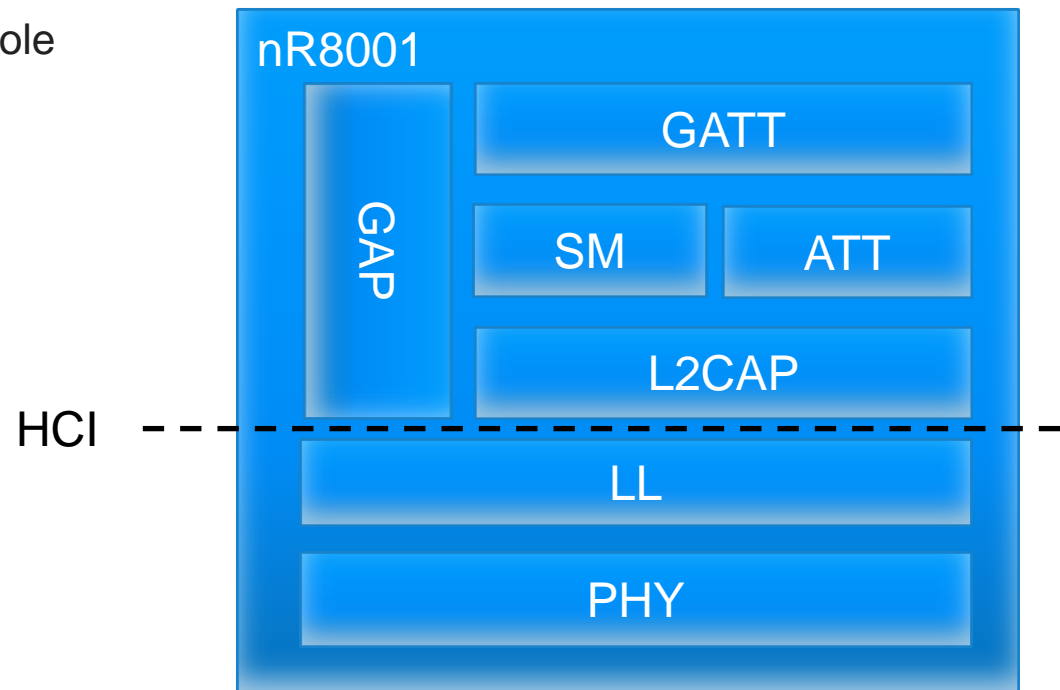
nRF8001

Single-chip *Bluetooth* low energy peripheral device

2

nRF8001

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



The concept



Existing application



(Serial interface)



Added Bluetooth
connectivity

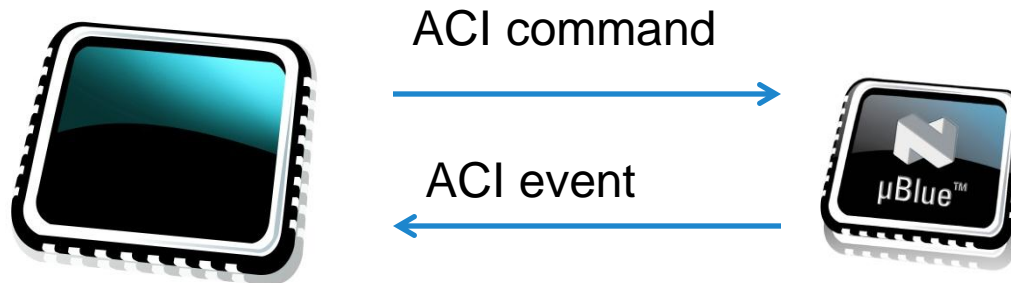
nRF8001 advantages

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

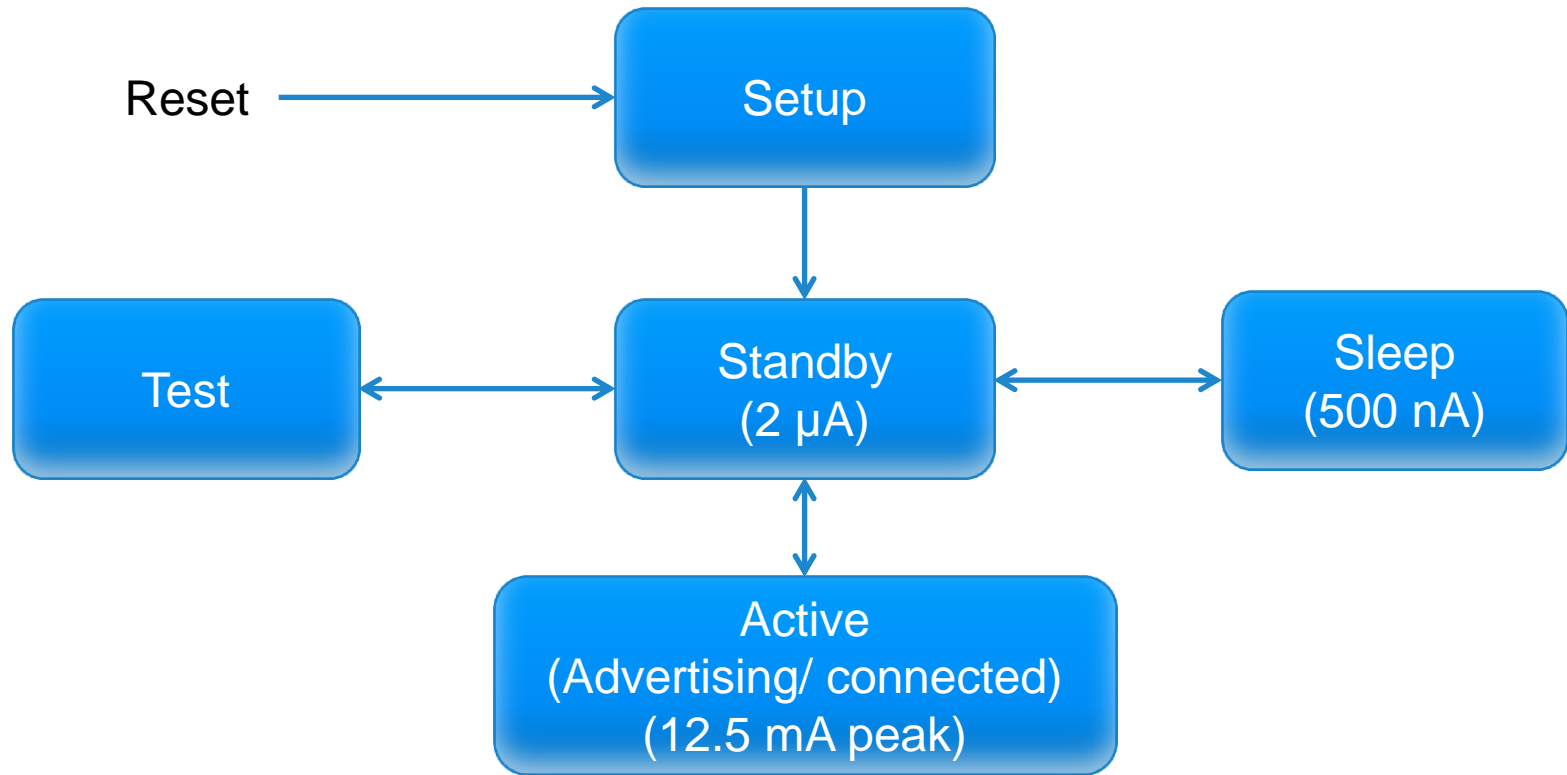
Application Controller Interface (ACI) physical part

- 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



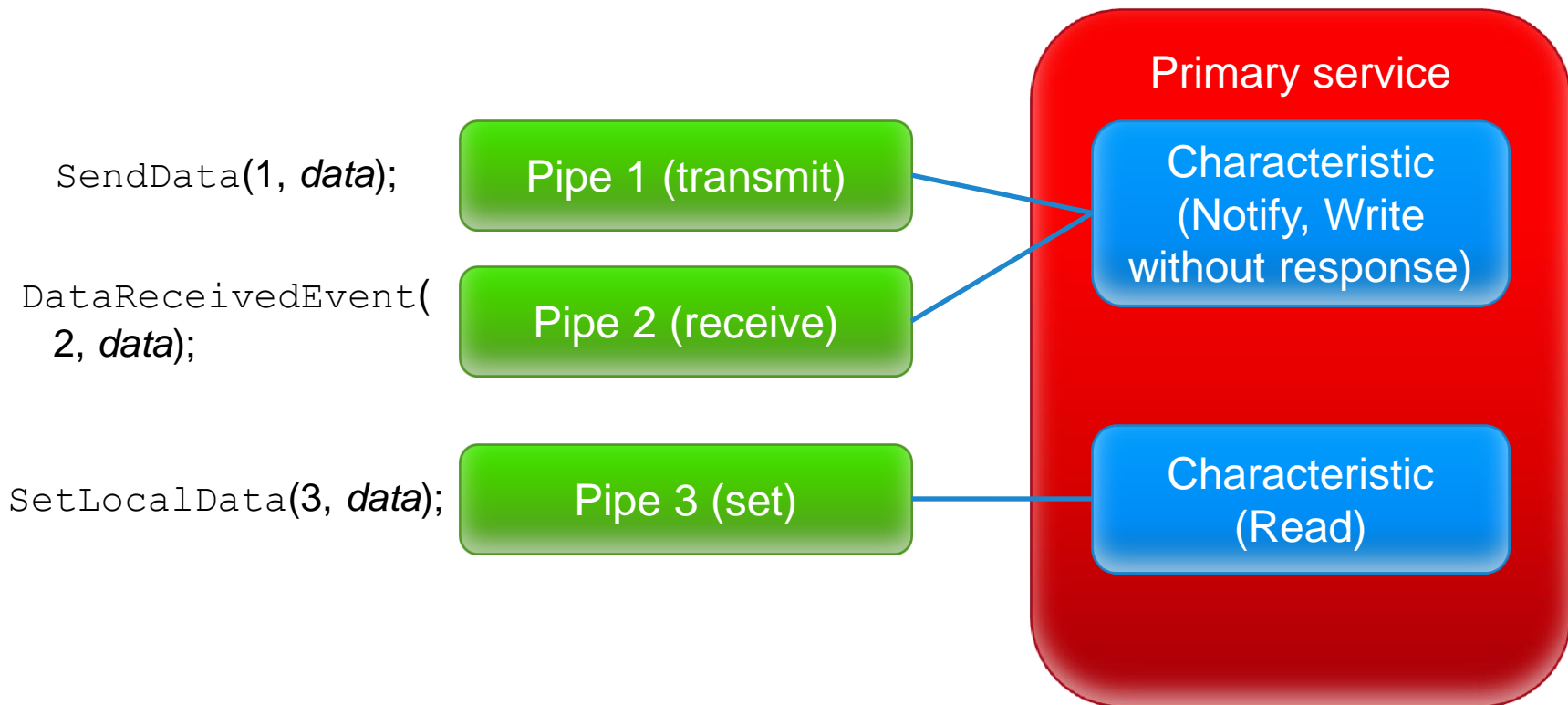
System commands and events

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 μ A).
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

14

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

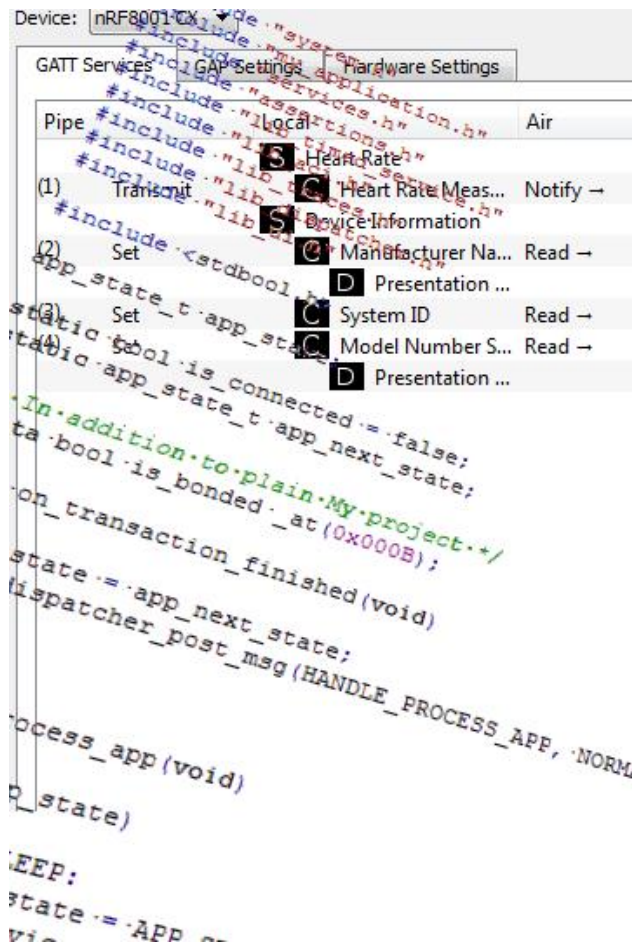
- Source code
 - Libraries
 - Examples
 - Documentation
- nRFgo Studio
 - nRF8001 configuration
 - Flash programming
- Master Control Panel
 - GUI for the master emulator

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



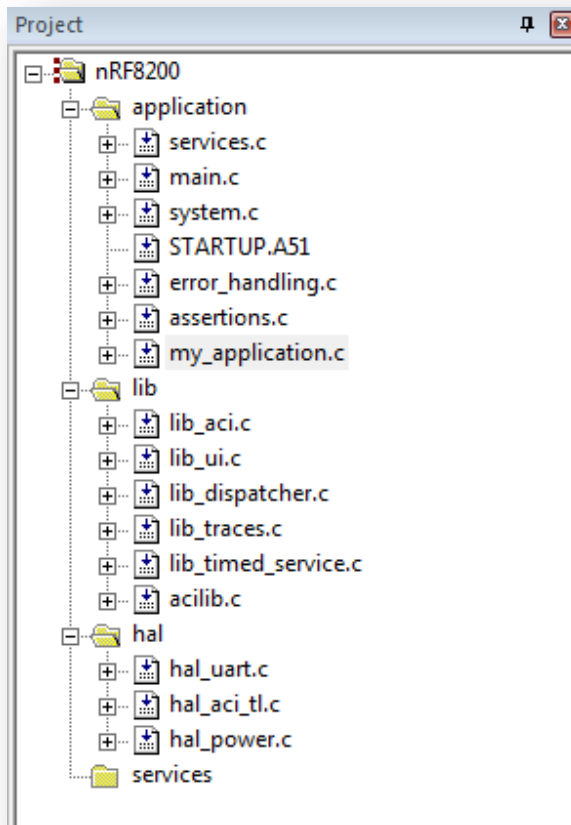
Development

How to develop applications and implement profiles

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

Device: **nRF8001 CX** ← IC code and build number Setup ID: 0

GATT Services | GAP Settings | Hardware Settings

Pipe	Local	Air	Remote
	S Heart Rate		
(1) Transmit	C Heart Rate Measurement	Notify →	
	S Device Information		
(2) Set	C Manufacturer Name String	Read →	
	D Presentation Format		
(3) Set	C System ID	Read →	
(4) Set	C Model Number String	Read →	
	D Presentation Format		

Services Add new service
 Drag and drop services from this list to the pane on the left

- Network Availability
- Immediate Alert
- Link Loss Alert
- TX Power
- Battery
- Health Thermometer
- Device Information
- Heart Rate

Defined services

ACI pipes

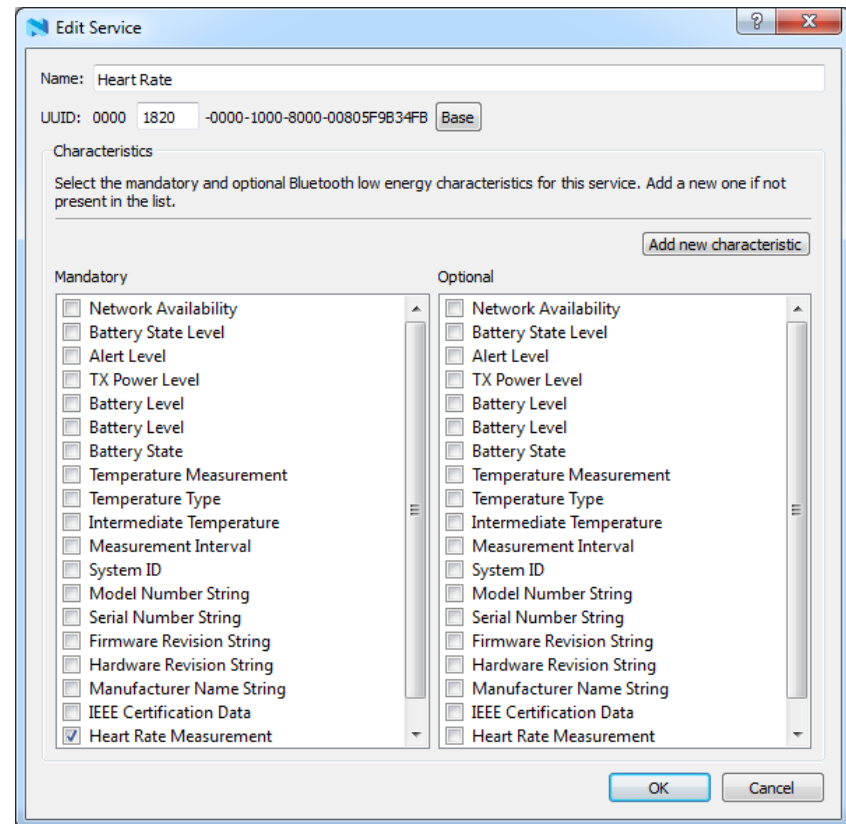
Heart Rate Monitor services

GATT procedure

Collector services (none)

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

The screenshot shows the 'Edit Characteristic' dialog box with the following fields and values:

- Characteristic Name: Heart Rate Measurement
- Characteristic UUID: 0000 2a40 -0000-1000-8000-00805F9B34FB (Base)
- Initial value: 003C (Hex)
- Max data length: 2 bytes
- Use fixed length: ☒
- Use Characteristic User Description: ☐
- Use Characteristic Presentation Format: ☐
- Format: boolean
- Exponent: 0
- Unit: 0000
- Name Space: 01
- Descriptor: 0000
- Profile Specific Characteristic Descriptors: (Empty list with 'Add new descriptor' button)
- Buttons: OK, Cancel

Defining GAP behavior

Device: nRF8001 CX

GATT Services GAP Settings Hardware Settings

Device identification:

Bluetooth Device Name: Nordic HRM

Characters in shortened name: 6

Appearance (16-bit UUID in HEX): 0000

Security:

Device security: Authentication and encryption

Minimum encryption key size: 7

Maximum encryption key size: 16

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)

Advertising:

General Bond

Local Name: Use shortened Local Name

TX Power Level: ☐ Advertise TX Power Level

Local Services: ☒ Advertise list of local services

Service Solicitation: ☐ Advertise list of solicited services

Slave Connection Interval Range: ☐ Advertise SCIR

TX Power offset: 0 dBm

Select the local services to advertise:

Service UUIDs: S Heart Rate
S Device Information

Select the remote services to solicit:

Service Solicitation:

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!

Timing parameters:

Maximum connection interval:	1500,00 ms	<input type="checkbox"/> No specific maximum
Minimum connection interval:	250,00 ms	<input type="checkbox"/> No specific minimum
Slave latency:	0	
Timeout multiplier:	100 ms	<input checked="" type="checkbox"/> No specific value

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

Advertiser packet	
1	Length: 2
2	AD type = Flags
3	Bonded Mode
4	Length: 3
5	More 16-bit UUIDs available
6	20
7	18
8	Length: 7
9	AD type = Shortened local name
10	'N'
11	'o'
12	'r'
13	'd'
14	'i'
15	'c'
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	

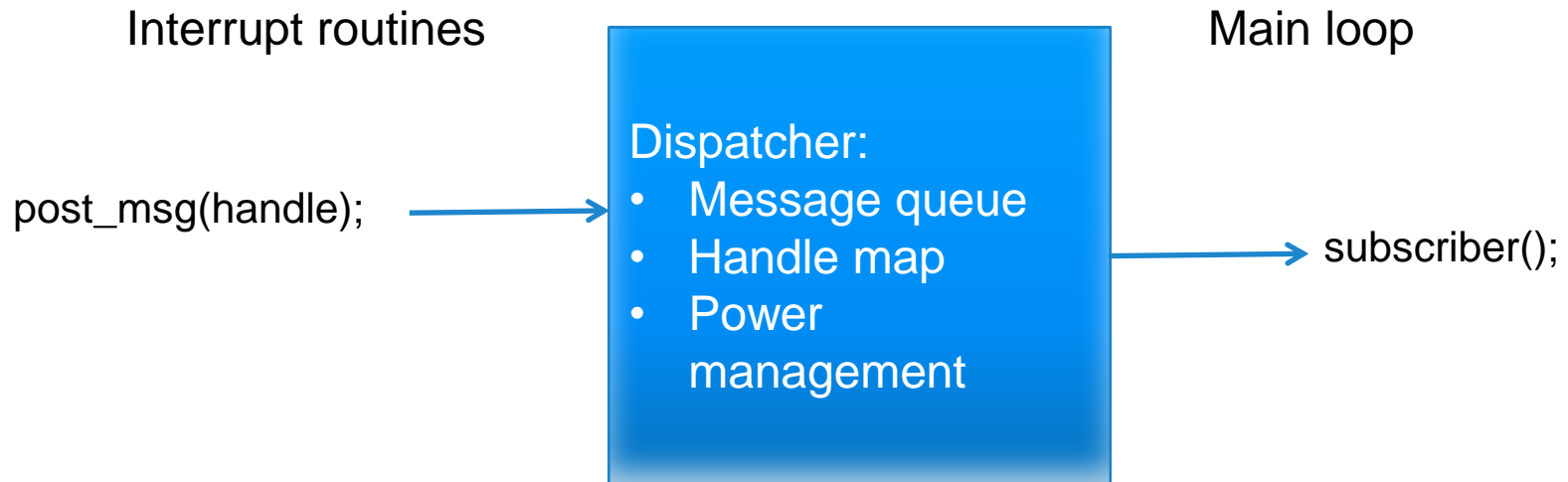
Generate setup files

- 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”

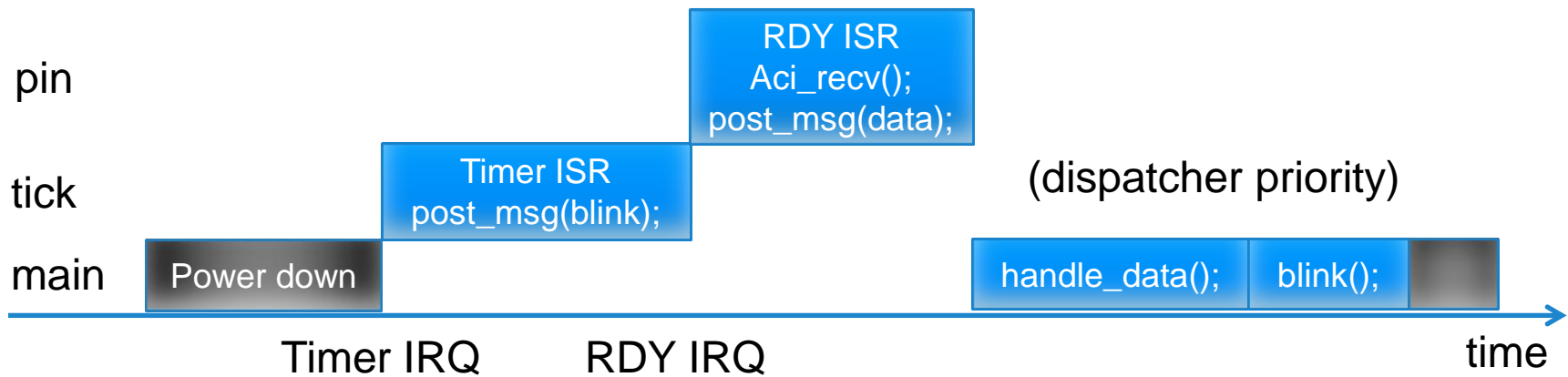
- The SDK examples are based on the dispatcher library:
 - Scheduler for the main vector.
 - Designed for low power.
 - Can be replaced by an RTOS.
- Offered support:
 - C implementation for nRF8200 (8051).
 - Setup tool in nRFgo Studio.
 - Documentation.

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	<code>HANDLE_EVT_RADIO_STARTED_STANDBY</code>	<code>on_radio_started_stdby(void)</code>	NULL	NULL
1	<code>HANDLE_EVT_RADIO_STARTED_SETUP</code>	<code>on_radio_started_setup(void)</code>	NULL	NULL
2	<code>HANDLE_EVT_RADIO_PIPES_CONNECTED</code>	<code>on_connect_pipe_ready_event(void)</code>	NULL	NULL
3	<code>HANDLE_EVT_RADIO_PIPES_DISCONNECTED</code>	<code>on_disconnect_event(void)</code>	NULL	NULL
4	<code>HANDLE_EVT_ERROR</code>	<code>radio_evt_error(void)</code>	NULL	NULL
5	<code>HANDLE_EVT_RADIO_DATA_RECEIVED</code>	<code>on_pipe_updated(void)</code>	NULL	NULL
6	<code>HANDLE_EVT_RADIO_ADVERTISE_TIMEOUT</code>	<code>on_advertise_timeout(void)</code>	NULL	NULL
7	<code>HANDLE_EVT_RADIO_DEVICE_VERSION</code>	<code>on_device_version(void)</code>	NULL	NULL
8	<code>HANDLE_EVT_RADIO_CMD_RESPONSE</code>	NULL	NULL	NULL
9	<code>HANDLE_TRANSACTION_FINISHED</code>	<code>on_transaction_finished(void)</code>	NULL	NULL
10	<code>HANDLE_EVT_RADIO_DEBUG_INFO</code>	<code>on_radio_evt_debug_info(void)</code>	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