S110 nRF51822

Bluetooth® low energy

SoftDevice Specification v1.1

Key Features

- *Bluetooth*® 4.0 compliant low energy single-mode protocol stack
    - Link layer
    - L2CAP, ATT, and SM protocols
    - GATT, GAP, and L2CAP
    - Peripheral and Broadcaster roles
    - GATT Client and Server
    - Full SMP support including MITM and OOB pairing
- Complementary nRF51 SDK including *Bluetooth* profiles and example applications
- Memory isolation between application and protocol stack for robustness and security
- Thread-safe supervisor-call based API
- Asynchronous, event-driven behavior
- No RTOS dependency
    - A RTOS of your choice can be used
- No link-time dependencies
    - Standard ARM® Cortex™-M0 project configuration for application development
- Support for non-concurrent multiprotocol operation
    - Alternate protocol stack running in application space

Applications

- Computer peripherals and I/O devices
    - Mouse
    - Keyboard
    - Multi-touch trackpad
- Interactive entertainment devices
    - Remote control
    - 3D glasses
    - Gaming controller
- Personal Area Networks
    - Health and fitness sensor and monitor devices
    - Medical devices
    - Key fobs and wrist watches
- Remote control toys

Liability disclaimer

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

Life support applications

Nordic Semiconductor's products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

Contact details

For your nearest dealer, please see http://www.nordicsemi.com.
Information regarding product updates, downloads, and technical support can be accessed through your My Page account on our homepage.

Main office: Otto Nielsens veg 12
7052 Trondheim
Norway

Phone: +47 72 89 89 00
Fax: +47 72 89 89 89

Mailing address: Nordic Semiconductor
P.O. Box 2336
7004 Trondheim
Norway

Revision History

Date	Version	Description
March 2013	1.1	Updated for changes made as of S110 v5.0.0; Changed ***section 7.2 "Processor availability"*** on page 20 and ***section 8 "Power profiles"*** on page 23; Changed ***Table 9*** on page 20; Added ***Table 10*** on page 21; Changed ***Table 12*** on page 22; Changed ***Figure 5*** on page 24 and ***Figure 6*** on page 25.
February 2013	1.0	Changed Memory resource requirements in ***Table 2*** on page 15; Added ***section 6.3 "Application signals - software interrupts"*** on page 17; Updated ***chapter 7 "Performance"*** on page 19 and added ***section 7.3 "Data throughput"*** on page 22; Updated diagrams in ***chapter 8 "Power profiles"*** on page 23; Added ***chapter 9 "Radio Notification"*** on page 26; Updated ***chapter 10.1 "S110 SoftDevice identification and revision scheme"*** on page 29.
September 2012	0.6	First release.

Contents

1 Introduction

The S110 SoftDevice is a *Bluetooth*® low energy (BLE) Peripheral protocol stack solution. It integrates a low energy controller and host, and provides a full and flexible API for building *Bluetooth* low energy System on Chip (SoC) solutions.

This document contains information about the S110 SoftDevice features and performance.

Note: The SoftDevice features and performance are subject to change between revisions of this document. See ***section 10.1 "S110 SoftDevice identification and revision scheme"*** on page 29 for more information.

1.1 Required reading

The *nRF51822 Product Specification* and the *nRF51 Series Reference Manual* are essential developer resources for *Bluetooth*® low energy solutions from Nordic Semiconductor. The software architecture chapter in the *nRF51 Series Reference Manual* is essential reading for understanding the resource usage and performance related chapters of this document.

Knowledge of The *Bluetooth* Core Specification, version 4.0, Volumes 1, 3, 4, and 6 is required for using the S110 correctly and for understanding the terminology used within this document.

1.2 Writing conventions

This SoftDevice Specification follows a set of typographic rules to ensure that the document is consistent and easy to read. The following writing conventions are used:

- Command, event names, and bit state conditions are written in `Lucida Console`.
- Pin names and pin signal conditions are written in **`Consolas.`**
- File names and User Interface components are written in **bold**.
- Internal cross references are italicized and written in ***semi-bold.***
- Placeholders for parameters are written in *italic regular font*. For example, a syntax description of SetChannelPeriod will be written as: SetChannelPeriod(*ChannelNumber, MessagingPeriod*).
- Fixed parameters are written in regular text font. For example, a syntax description of SetChannelPeriod will be written as: SetChannelPeriod(0, Period).

2 Product overview

This section provides an overview of the S110 SoftDevice.

2.1 SoftDevice

The S110 SoftDevice is a pre-compiled and linked binary software implementing a *Bluetooth* 4.0 low energy (BLE) protocol stack. The S110 is compatible with selected nRF51 System on Chip (SoC) devices. The Application Programming Interface (API) is a standard C language set of functions and data types that give the application complete compiler and linker independence from the SoftDevice implementation. From an application development point of view, a SoftDevice enables the application programmer to develop their code as a standard ARM® Cortex™-M0 project without needing to integrate with proprietary chip-vendor software frameworks. This means that any ARM® Cortex™-M0 compatible toolchain can be used to develop *Bluetooth* low energy applications with the S110 SoftDevice.

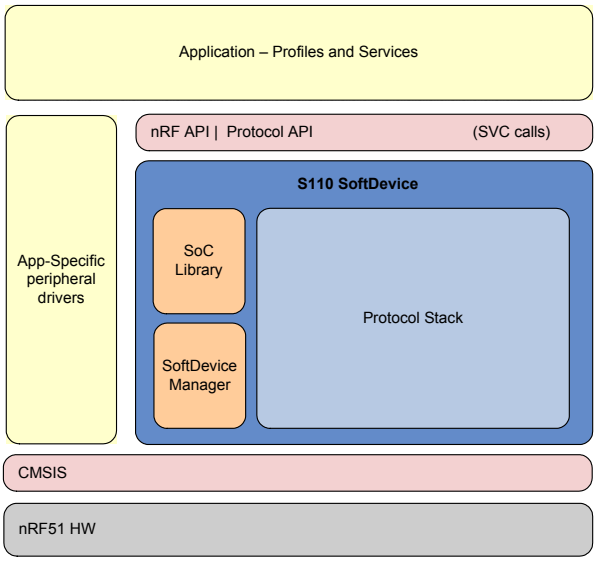

Figure 1 *System on Chip application with the SoftDevice*

The S110 SoftDevice can be programmed onto compatible nRF51 devices during both development and production. This specification outlines the supported features of a production S110 SoftDevice. Alpha and Beta versions may not support all features. To find information on any limitations or omissions, the S110 SoftDevice release notes will contain a detailed summary of the release status.

2.2 Multiprotocol support

The S110 SoftDevice supports non-concurrent multiprotocol implementations. This means a proprietary 2.4 GHz protocol can be implemented in the application program area. This protocol can access all hardware resources when the S110 SoftDevice is disabled.

3 *Bluetooth* low energy protocol stack

The *Bluetooth* 4.0 compliant low energy (BLE) Host and Controller embedded in the S110 SoftDevice are fully qualified with multi-role support (Peripheral and Broadcaster). The API is defined above the Generic Attribute Protocol (GATT), Generic Access Profile (GAP), and Logical Link Control and Adaptation Protocol (L2CAP). The S110 SoftDevice allows applications to implement standard *Bluetooth* low energy profiles as well as proprietary use case implementations.

The nRF51 Software Development Kit (SDK) completes the BLE protocol stack with Service and Profile implementations. Single-mode System on Chip (SoC) applications are enabled by the full BLE protocol stack and nRF518xx integrated circuit (IC).

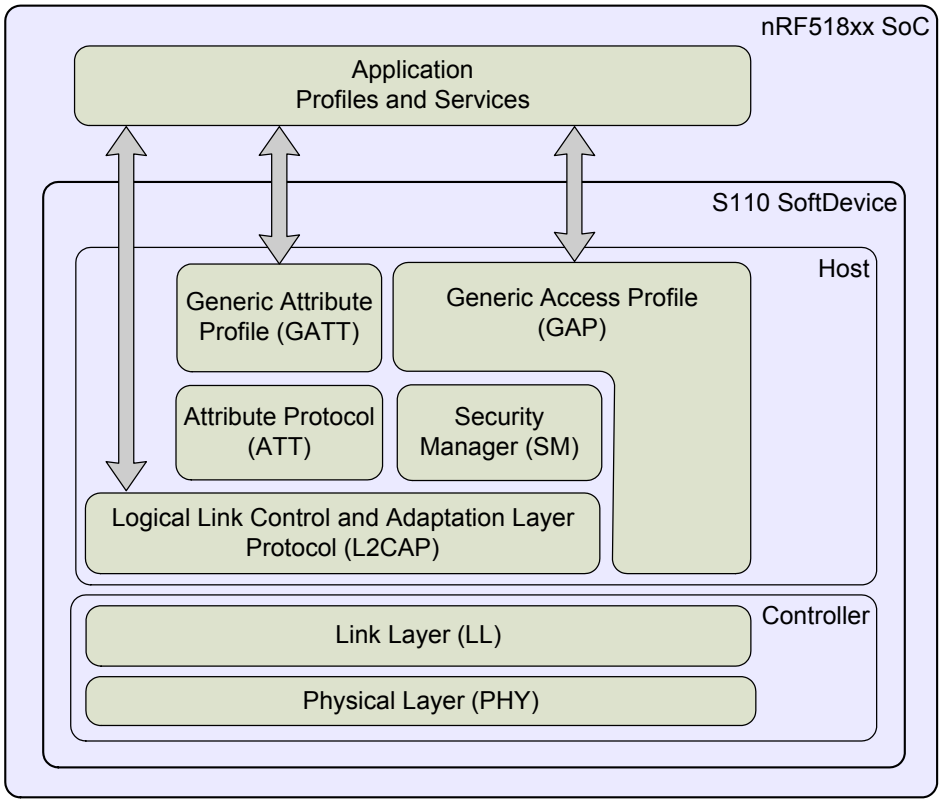

Figure 2 *LE stack architecture*

3.1 Profile and service support

The peripheral behavior for all adopted profiles is supported by the S110 SoftDevice at the time of publishing. The Profiles and corresponding Services supported by the S110 SoftDevice are shown in ***Table 1***.

Adopted Profile	Adopted Services	Supported
HID over GATT	HID Battery Device Information	YES
Heart Rate Monitor	Heart Rate Device Information	YES
Proximity	Link Loss Immediate Alert TX Power	YES
Blood Pressure	Blood pressure	YES
Health Thermometer	Health Thermometer	YES
Glucose	Glucose	YES
Phone Alert Status	Phone Alert Status	YES
Alert Notification	Alert Notification	YES
Time	Current Time Next DST Change Reference Time Update	YES
Find Me	Immediate Alert	YES
Cycling speed and cadence	Cycling speed and cadence Device information	YES
Running speed and cadence	Running speed and cadence Device information	YES

Table 1 *Adopted Profile and Service support*

3.2 *Bluetooth* low energy features

The BLE protocol stack in the S110 SoftDevice has been designed to provide an abstract, but flexible, interface for application development for *Bluetooth* low energy devices. GAP, GATT, SM, and L2CAP are implemented in the SoftDevice and managed through the API. GAP and GATT procedures and modes that are common to most profiles, such as the handling of discoverability, connectability, pairing, and bonding, are implemented in the stack.

The BLE API is consistent across *Bluetooth* role implementations where features in common have the same interface. In the following sections, the features of the BLE stack in the S110 SoftDevice are identified.

3.3 Application Programming Interface (API)

Feature	Description
Interface to: GATT/GAP/L2CAP	Consistency between APIs including shared data formats.
GATT DB population and access	Full flexibility to populate the DB at run time, attribute removal is not supported.
Thread-safe, asynchronous, and event driven	Minimizes exposure to concurrency issues.
Vendor-specific (128 bit) UUIDs for proprietary profiles	Compact, fast, and memory efficient management of 128 bit UUIDs.
Non-concurrent multi-protocol	
Packet flow control	Zero-copy buffer management.

3.4 Generic Access Profile (GAP)

Feature	Description
Multi-role: Peripheral & Broadcaster	
Limited and general discoverable modes	
Multiple bond support	Keys and peer information stored in application space No limitations in stack implementation.
User-defined Advertising data	Full control over advertising and scan response data for the application.
Security mode 1: Levels 1, 2, and 3	

3.5 Generic Attribute Profile (GATT)

Features	Description
Comprehensive GATT Server	
Support for authorization: R/W characteristic value R/W descriptors	Enables control points Enables freshest data Enables GAP authorization
Full GATT Client	Flexible data management options for packet transmission with either fine control or abstract management
Implemented GATT Sub-procedures	Discover all Primary Services Discover Primary Service by Service UUID Find included Services Discover All Characteristics of a Service Discover Characteristics by UUID Discover All Characteristic Descriptors Read Characteristic Value Read using Characteristic UUID Read Long Characteristic Values Write Without Response Write Characteristic Value Notifications Indications Read Characteristic Descriptors Read Long Characteristic Descriptors Write Characteristic Descriptors

3.6 Security Manager (SM)

Feature	Description
Lightweight key storage for reduced NV memory requirements	
Authenticated MITM (Man in the middle) protection	
Pairing methods: Just works, Passkey Entry, and Out of Band	

3.7 Attribute Protocol (ATT)

Feature	Description
Server protocol	
Client protocol	

3.8 Logical Link Control and Adaptation Layer Protocol (L2CAP)

Feature	Description
27 byte MTU size	
Dynamically allocated channels	

3.9 Controller, Link Layer (LL)

Feature	Description
Slave role	
Slave connection update	
27 byte MTU	
Encryption	

3.10 Proprietary features

Feature	Description
TX Power control	Access for the application to change TX power settings anytime.
Application Latency	Enhanced management of power versus response time of an application to data sent from a peer device.
Resolvable address whitelisting based on IRK	Synchronous and low power solution for BLE enhanced privacy.

4 SoC library

The following features ensure the Application and SoftDevice coexist with safe sharing of common SoC resources.

Feature	Description
sd_mutex_*	Atomic mutex API. Disabling global interrupts in the application could cause dropped packets or lost connections. This API safely implements an atomic operation for the application to use.
sd_nvic_*	Gives the application access to all NVIC features without corrupting SoftDevice configurations.
sd_rand_*	Gives access to the random number generator hardware.

sd_power_*	Access to POWER block configuration while the SoftDevice is enabled:
	• Access to RESETREAS register
	• Set power modes
	• Configure power fail comparator
	• Control RAM block power
	• Use general purpose retention register
	• Configure DC/DC converter state
	  • OFF
	  • ON
	  • AUTOMATIC - The SoftDevice will manage the DC/DC converter state by switching it on for all Radio Events and off all other times.

sd_clock_*	Access to CLOCK block configuration while the SoftDevice is enabled. Allows the HFCLK Crystal Oscillator source to be requested by the application.
sd_app_event_wait	Simple power management hook for the application to use to enter a sleep or idle state and wait for an event.
sd_ppi_*	Configuration interface for PPI channels and groups reserved for an application.
sd_radio_notification_cfg_set	Configure Radio Notification signals on ACTIVE and/or INACTIVE. See ***chapter 9 "Radio Notification"*** on page 26.
sd_ecb_block_encrypt	Safe use of 128 bit AES encrypt HW accelerator.
sd_event_get	Fetch asynchronous events generated by the SoC library.
sd_power_dcdc_mode_set	The SoC library allows the DC/DC converter to be enabled for an application with the following two options: • ON - enabled all the time. • Auto managed - Enabled by the SoftDevice for each Radio Event and disabled outside of Radio Events.

5 SoftDevice Manager

The following feature enables the Application to manage the SoftDevice on a top level.

Feature	Description
`sd_softdevice_ *`	Control of SoftDevice state through enable and disable. On enable, the low frequency clock source selects between the following options: • RC oscillator • Synthesized from high frequency clock • Crystal oscillator

6 S110 resource requirements

The S110 SoftDevice uses on-chip resources including memory, system blocks, and peripheral blocks. The use of resources may change depending on if the SoftDevice is enabled or disabled. This chapter outlines how memory and hardware are used by the SoftDevice.

6.1 Memory resource map and usage

The memory map for program memory and RAM at run time with the SoftDevice enabled is illustrated in ***Figure 3*** below. Memory resource requirements, both when the SoftDevice is enabled and disabled, are shown in ***Table 2*** on page 15.

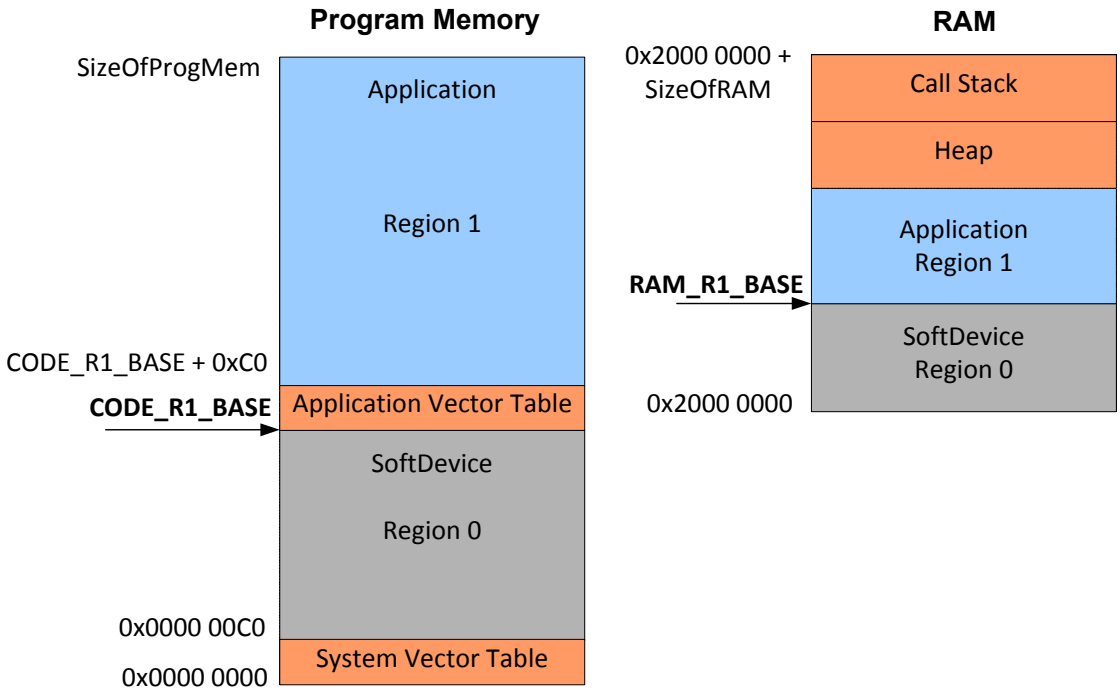

Figure 3 *Memory resource map*

Flash	S110 Enabled	S110 Disabled
Amount	80 kB	80 kB
CODE_R1_BASE	0x0001 4000	0x0001 4000

RAM	S110 Enabled	S110 Disabled
Amount	8 kB	4 bytes
RAM_R1_BASE	0x2000 2000	0x2000 0004

Call stack[1]	S110 Enabled	S110 Disabled
Maximum usage	1.5 kB	0 kB

Heap	S110 Enabled	S110 Disabled
Maximum allocated bytes	0 bytes	0 bytes

1. This is only the call stack used by the SoftDevice at run time. The application call stack memory usage must be added for the total call stack size to be set in the user application.

***Table 2** S110 Memory resource requirements*

6.2 Hardware blocks and interrupt vectors

Table 3 defines access types used to indicate the availability of hardware blocks to the application. ***Table 4*** on page 16 specifies the access the application has, per hardware block, both when the SoftDevice is enabled and disabled.

Access	Definition
Restricted	Used by the SoftDevice. Application has limited access through the SoftDevice API.
Blocked	Used by the SoftDevice. Application has no access.
Open	Not used by the SoftDevice. Application has full access.

***Table 3** Hardware access type definitions*

ID	Base address	Instance	Access (S110 enabled)	Access (S110 disabled)
0	0x40000000	POWER	Restricted	Open
0	0x40000000	CLOCK	Restricted	Open
1	0x40001000	RADIO	Blocked	Open
2	0x40002000	UART0	Open	Open
3	0x40003000	SPIM0 / 2W0	Open	Open
4	0x40004000	SPIM1 / 2W1	Open	Open
		…		
6	0x40006000	Port 0 GPIOTE	Open	Open
7	0x40007000	ADC	Open	Open
8	0x40008000	TIMER0	Blocked	Open
9	0x40009000	TIMER1	Open	Open
10	0x4000A000	TIMER2	Open	Open
11	0x4000B000	RTC0	Blocked	Open
12	0x4000C000	TEMP	Open	Open
13	0x4000D000	RNG	Restricted	Open
14	0x4000E000	ECB	Restricted	Open
15	0x4000F000	CCM	Blocked	Open
15	0x4000F000	AAR	Blocked	Open
16	0x40010000	WDT	Open	Open
17	0x40011000	RTC1	Open	Open
18	0x40012000	QDEC	Open	Open
		…		
20	0x40014000	Software interrupt	Open	Open
21	0x40015000	Software interrupt	Open or blocked[1]	Open
22	0x40016000	Software interrupt	Open or blocked[1]	Open
23	0x40017000	Software interrupt	Blocked	Open
24	0x40018000	Software interrupt	Blocked	Open
25	0x40019000	Software interrupt	Blocked	Open
		…		
30	0x4001E000	NVMC	Open	Open
31	0x4001F000	PPI	Restricted	Open
NA	0x50000000	GPIO P0	Open	Open
NA	0xE000E100	NVIC	Restricted[2]	Open

1. Blocked only when signals are configured. See ***Table 6*** on page 17 for software interrupt allocation.
2. Not protected. For robust system function, the application program must comply with the restriction and use the NVIC API for configuration when the SoftDevice is enabled.

Table 4 *Peripheral protection and usage by SoftDevice*

6.3 Application signals - software interrupts

Software interrupts are used by the S110 SoftDevice to signal the application of events. ***Table 5*** shows the allocation of software interrupt vectors to SoftDevice signals.

Software interrupt (SWI)	Peripheral ID	SoftDevice Signal
0	20	Unused by the SoftDevice and available to the application.
1	21	Radio Notification - optionally configured through API.
2	22	SoftDevice Event Notification.
3	23	Reserved.
4	24	Lower stack processing - not user configurable.
5	25	Upper stack signaling - not user configurable.

Table 5 *Software interrupt allocation*

6.4 Programmable Peripheral Interconnect (PPI)

When the SoftDevice is enabled, the PPI is restricted with only some PPI channels and groups available to the application. ***Table 6*** shows how channels and groups are assigned between the application and SoftDevice.

Note: All PPI channels are available to the application when the SoftDevice is disabled.

PPI channel allocation	S110 enabled	S110 disabled
Application	Channels 0 - 7	Channels 0 - 15
SoftDevice	Channels 8 - 15	-

PPI group allocation	S110 enabled	S110 disabled
Application	Groups 0 - 1	Groups 0 - 3
SoftDevice	Groups 2 - 3	-

Table 6 *PPI channel and group availability*

6.5 SVC number ranges

Table 7 shows which SVC numbers an application program can use and which numbers are used by the SoftDevice.

Note: The SVC number allocation does not change with the state of the SoftDevice (enabled or disabled).

SVC number allocation	S110 enabled	S110 disabled
Application	0x00-0x0F	0x00-0x0F
SoftDevice	0x10-0xFF	0x10-0xFF

Table 7 *SVC number allocation*

7 Performance

This chapter documents key SoftDevice performance parameters for interrupt latency, processor availability, and data throughput.

7.1 Interrupt latency

Latency, additional to ARM® Cortex™-M0 hardware architecture latency, is introduced by SoftDevice logic to manage interrupt events. This latency occurs when an interrupt is forwarded to the application from the SoftDevice and is part of the minimum latency for each application interrupt. The maximum application interrupt latency is dependent on protocol stack activity as described in ***section 7.2 "Processor availability"*** on page 20.

Interrupt	CPU cycles	Latency at 16 MHz
Open peripheral interrupt	50	3.2 µs
Blocked or restricted peripheral interrupt (only forwarded when SoftDevice disabled)	63	4 µs
Application SVC interrupt	14	1 µs

Table 8 *Additional latency due to SoftDevice processing*

See ***Table 4*** on page 16 for open, blocked, and restricted peripherals.

7.2 Processor availability

The SoftDevice architecture chapter of the *nRF51 Reference Manual* describes interrupt management in SoftDevices and is required knowledge for understanding this section.

Illustrated in ***Figure 4*** are the parameter values from ***Table 9***. The parameters are defined around lower stack and upper stack interrupts. These interrupts service real time protocol events and API calls (or deferred internal SoftDevice tasks) respectively. Lower stack interrupts are extended by a "CPU Suspend" state during radio activity to improve link integrity. This means lower stack interrupts will block application and upper stack processing during a Radio Event for a time proportional to the number of packets transferred in the event. See ***Table 10*** on page 21 for more information.

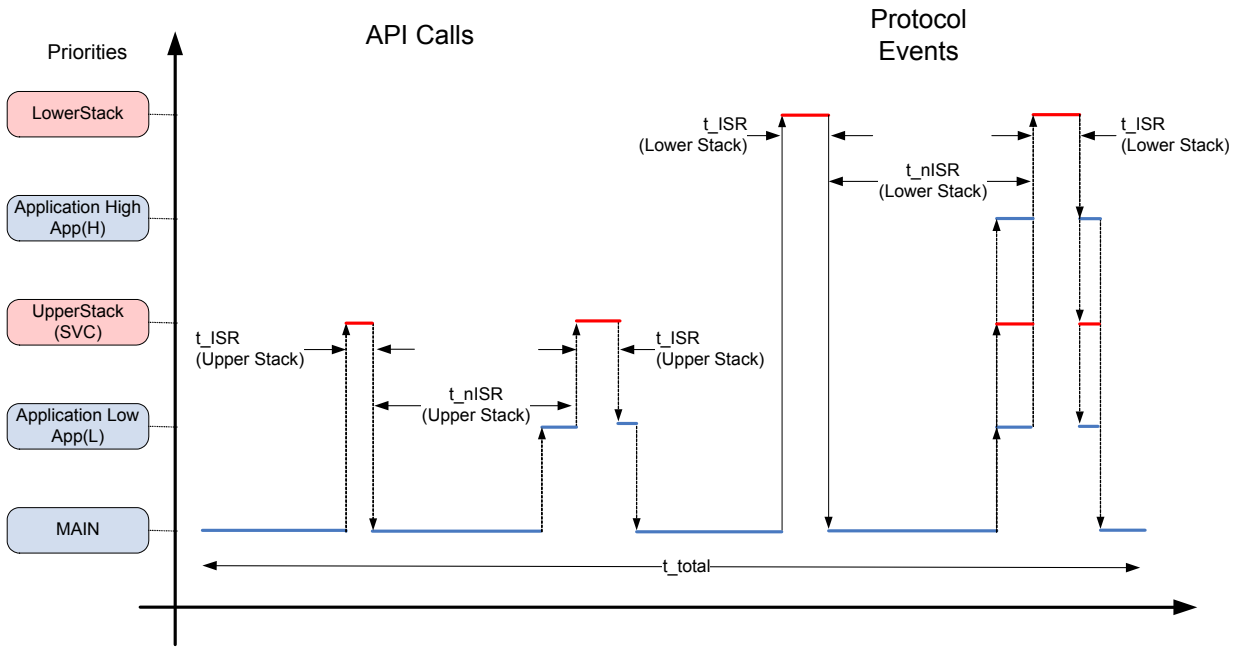

Figure 4 *Interrupt latencies due to SoftDevice processing*

		Upper stack		
Parameter	**Description**	**Min**	**Nom**	**Max**
$t_{ISR(upper\ stack)}$	Maximum interrupt processing time	-	-	250 µs
$t_{nISR(upper\ stack)}$	Minimum time between interrupts	Application dependent.[1]		

1. Calls to the SoftDevice API trigger the upper stack interrupt.

Table 9 *S110 interrupt latency - upper stack*

Parameter	Description	Packets	Nominal
$t_{ISR(lower\ stack)1}$	Maximum interrupt latency during Radio Event. Includes the time the CPU is used by the lower stack for processing and the time the CPU suspended during radio activity. In each case, maximum encrypted packet length in both RX and TX are assumed.	1	1180 µs
$t_{ISR(lower\ stack)\ 2}$		2	2136 µs
$t_{ISR(lower\ stack)\ 3}$		3	3092 µs
$t_{ISR(lower\ stack)\ 4}$		4	4048 µs
$t_{ISR(lower\ stack)\ 5}$		5	5004 µs
$t_{ISR(lower\ stack)\ 6}$		6	5960 µs
$tn_{ISR(lower\ stack)}$	Minimum time between lower stack interrupts.	n/a	1390 µs

Table 10 *S110 interrupt latency lower stack*

Application Low, App(L), can be blocked by the SoftDevice for a maximum of:

$$App(L)_{latency_{max}} = t_{ISR_{max(Upper\ Stack)}} + t_{ISR_{max(Lower\ Stack)}}$$

Application High, App(H), can be blocked by the SoftDevice for a maximum of:

$$App(H)_{latency\_max} = t_{ISR_{max(Lower\ Stack)}}$$

Table 11 shows expected CPU utilization percentages for the upper stack and lower stack given a set of typical stack connection parameters.

Note: Upper stack utilization is based only on the processing required to update the database and transfer data to and from the application when the data is transferred.

BLE connection configuration	Lower stack	Upper stack	CPU suspend	Remaining
Connection interval 4 s No data transfer	0.01%	0.01%	0.03%	~99%
Connection interval 100 ms 1 packet transfer per event	0.4%	0.6%	0.7%	~98%
Connection interval 7.5 ms 4 packet transfer per event	11%	27%	42%	~20%

Table 11 *Processor usage and remaining availability for example BLE connection configurations*

7.3 Data throughput

The maximum data throughput limits in ***Table 12*** apply to encrypted packet transfers. To achieve maximum data throughput, the application must exchange data at a rate that matches on-air packet transmissions and use the maximum data payload per packet.

Protocol	Role	Method	Maximum data throughput
L2CAP		Receive	140 kbps
		Send	140 kbps
		Simultaneous send and receive	110 kbps
GATT	Client	Receive Notification	120 kbps
		Send Write command	120 kbps
		Send Write request	10 kbps
		Simultaneous receive Notification and send Write command	100 kbps
GATT	Server	Send Notification	120 kbps
		Receive Write command	110 kbps
		Receive Write request	10 kbps
		Simultaneous send Notification and receive Write command	80 kbps

Table 12 *L2CAP and GATT maximum data throughput*

8 Power profiles

This chapter provides power profiles for MCU activity during *Bluetooth* low energy Radio Events implemented in the S110 SoftDevice. These profiles give a detailed overview of the stages of a Radio Event, the approximate timing of stages within the event, and how to calculate the peak current at each stage using data from the nRF51822 Product Specification. The lower stack CPU profile (including CPU activity and CPU suspend) during the event is shown separately. These profiles are based on events with empty packets.

8.1 Connection event

Stage	Description	Current Calculations[1]
(A)	Preprocessing	ION + I_{RTC} + I_{X32k} + $I_{CPU,Flash}$
(B)	Standby + XO ramp	ION + I_{RTC} + I_{X32k} + $I_{START,X16M}$
(C)	Standby	ION + I_{RTC} + I_{X32k} + I_{X16M}
(D)	Radio Start	ION + I_{RTC} + I_{X32k} + I_{X16M} + $\int(I_{START,RX})$
(E)	Radio RX	ION + I_{RTC} + I_{X32k} + I_{X16M} + I_{RX} + I_{CRYPTO}
(F)	Radio turn-around	ION + I_{RTC} + I_{X32k} + I_{X16M} + $\int(I_{START,TX})$
(G)	Radio TX	ION + I_{RTC} + I_{X32k} + I_{X16M} + $I_{TX,0dBM}$ + I_{CRYPTO}
(H)	Post-processing	ION + I_{RTC} + I_{X32k} + $I_{CPU,Flash}$
(I)	Idle - connected	ION + I_{RTC} + I_{X32k}

1. See the nRF51822 Product Specification for the symbol values.

Note: When using the 32.768 kHz RC oscillator, I_{RC32k} must be used instead of I_{X32k}.

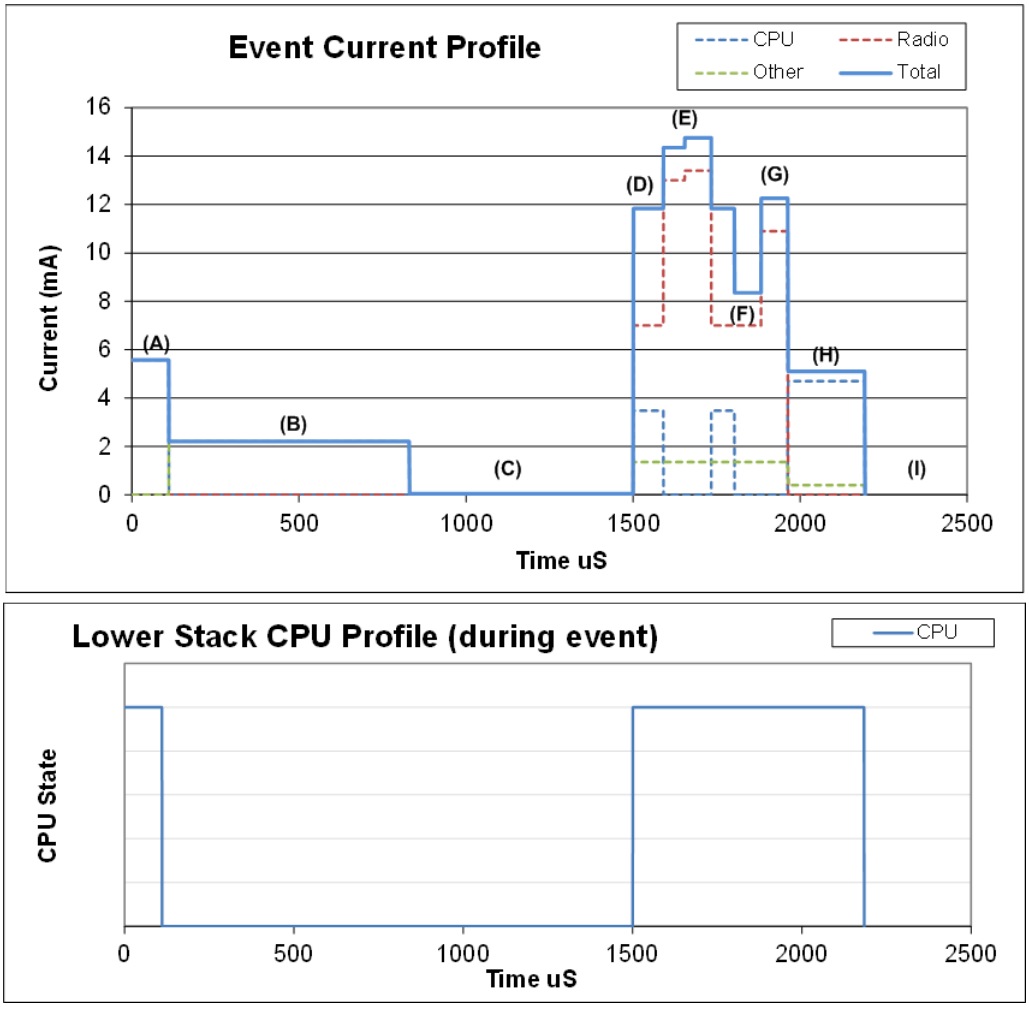

Figure 5 *Connection event*

8.2 Advertising event

Stage	Description	Current Calculation[1]
(A)	Pre-processing	ION + I_{RTC} + I_{X32k} + $I_{CPU,Flash}$
(B)	Standby + XO ramp	ION + I_{RTC} + I_{X32k} + $I_{START,X16M}$
(C)	Standby	ION + I_{RTC} + I_{X32k} + I_{X16M}
(D)	Radio start/switch	ION + I_{RTC} + I_{X32k} + I_{X16M} + ∫ ($I_{START,TX}$)
(E)	Radio TX	ION + I_{RTC} + I_{X32k} + I_{X16M} + $I_{TX,0dBM}$
(F)	Radio turn-around	ION + I_{RTC} + I_{X32k} + I_{X16M} + ∫ ($I_{START,RX}$)
(G)	Radio RX	ION + I_{RTC} + I_{X32k} + I_{X16M} + I_{RX}
(H)	Post-processing	ION + I_{RTC} + I_{X32k} + $I_{CPU,Flash}$
(I)	Idle	ION + I_{RTC} + I_{X32k}

1. See the *nRF51822 Product Specification* for the symbol values.

Note: IRC32k should be substituted for I_{X32k} when using the 32.768k R_{COSC}

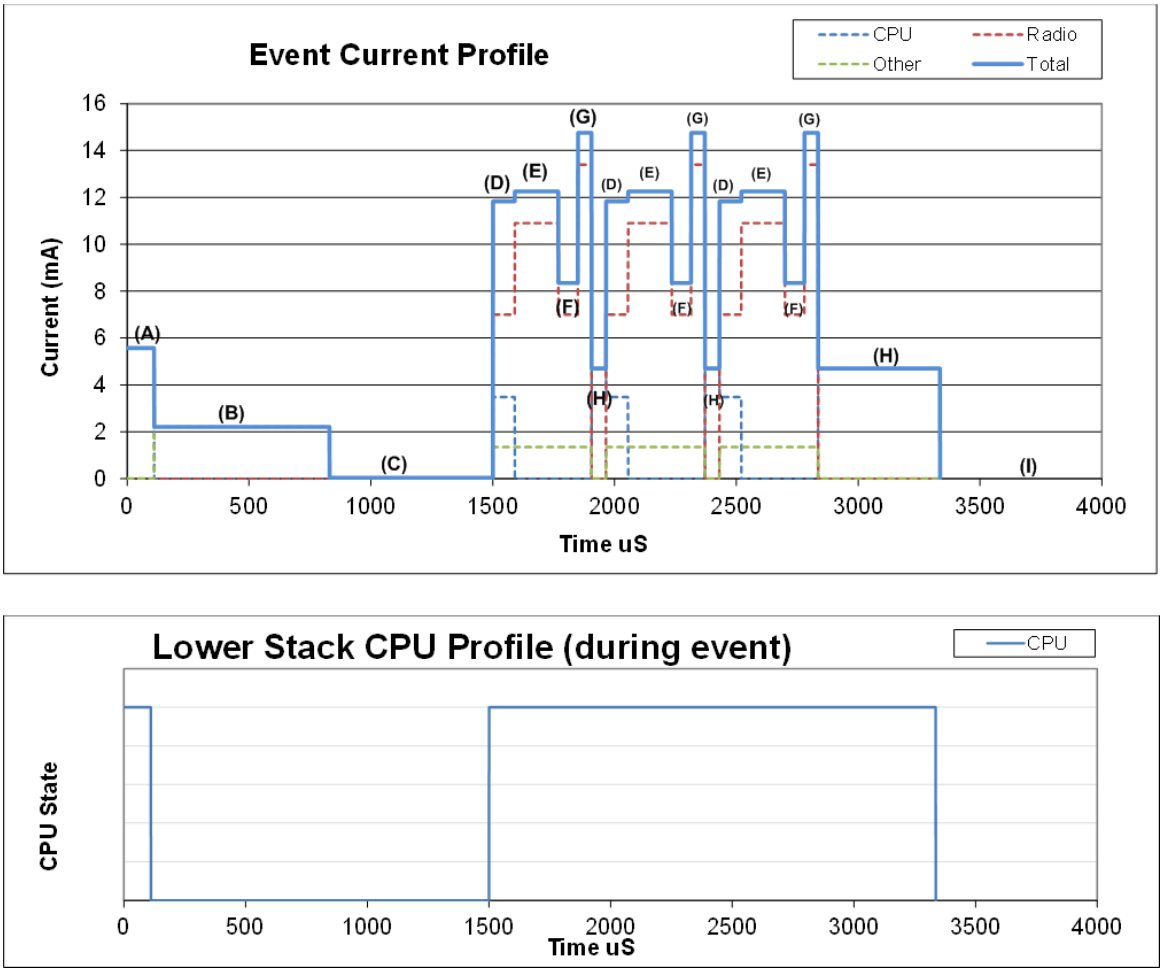

Figure 6 *Advertising event*

9 Radio Notification

The Radio Notification is a configurable feature which enables ACTIVE and INACTIVE (nACTIVE) signals from the S110 SoftDevice to the application to notify when the Radio will be in use. The signal is sent using software interrupt, as specified in ***section 6.3 "Application signals - software interrupts"*** on page 17.

The ACTIVE signal, if enabled, is sent before the Radio Event starts. The INACTIVE signal is sent at the end of the Radio Event. These signals can be used by the application programmer to synchronize application logic with Radio activity and packet transfers. For example, the ACTIVE signal can be used to shut off external devices to manage peak current drawn during periods when the radio is on; or to trigger sensor data collection for transmission in the Radio Event.

Figure 7 shows the active signal in relation to the Radio Event.

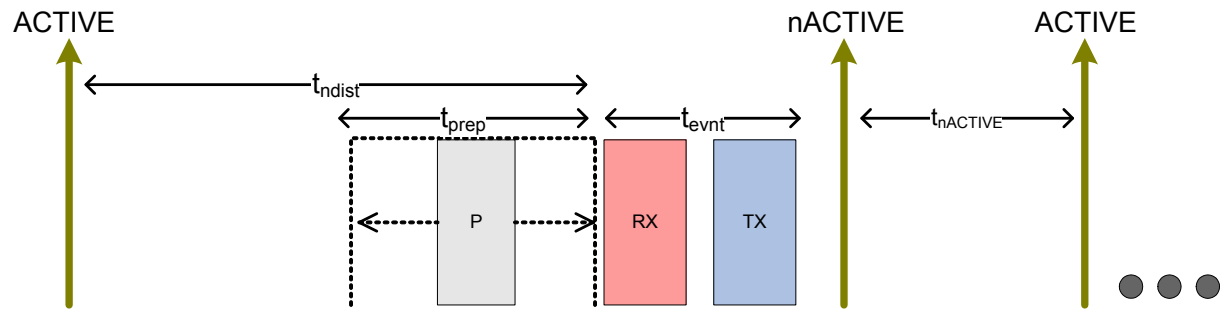

Figure 7 *Radio Notification*

Many packets can be sent and received in one Radio Event. Radio Notification events will be as shown in ***Figure 8***.

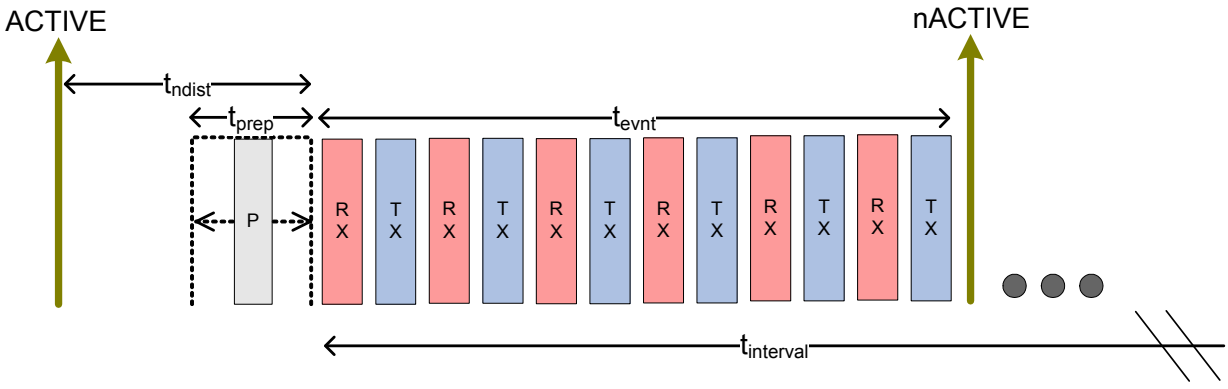

Figure 8 *Radio Notification, multiple packet transfers*

Table 13 describes the notation used in ***Figure 7*** and ***Figure 8*** on page 26.

Label	Description	Notes
ACTIVE	The ACTIVE signal prior to a Radio Event.	
nACTIVE	The INACTIVE signal after a Radio Event.	Because both ACTIVE and INACTIVE use the same software interrupt, it is up to the application to manage them. If both ACTIVE and INACTIVE are configured ON by the application, there will always be an ACTIVE signal before an INACTIVE signal.
t_{ndist}	The notification distance - the time between ACTIVE and first RX/TX in a Radio Event.	This time is configurable by the application developer and can be changed in between Radio Events.
$t_{interval}$	Time between Radio Events as per the protocol.	
t_{evnt}	The time used in a Radio Event.	
t_{prep}	The time before first RX/TX to prepare and configure the radio.	The application will be interrupted by the lower stack during t_{prep}. **Note**: All packet data to send in an event should be sent to the stack t_{prep} before the Radio starts.
P	CPU processing in the lower stack interrupt between ACTIVE and RX.	The CPU processing may occur anytime, up to t_{prep} before RX.

Table 13 *Radio Notification figure labels*

Table 14 shows the ranges of the timing symbols in ***Figure 7*** on page 26.

Value	Range (µs)
t_{ndist}	800, 1740, 2680, 3620, 4560, 5500
t_{evnt}	550 to 1300 Advertiser - 0 to 31 bytes payload, 3 channels) 900 to 5400 Slave - 1 to 6 packets RX and TX unencrypted data when connected) 1000 to 5800 Slave - 1 to 6 packets RX and TX encrypted data when connected)
t_{prep}	200 to 1500
P	<=150

Table 14 *Radio Notification timing ranges*

Using the numbers from ***Table 14***, the amount of CPU time available between ACTIVE and a Radio Event is:

$$t_{ndist} - P$$

Shown here is the amount of time before stack interrupts begin. Data packets must be transferred to the stack using the API within this time from the ACTIVE signal if they are to be sent in the next Radio Event.

$$t_{ndist} - t_{prep(maximum)}$$

Note: t_{prep} may be larger than t_{ndist} when $t_{ndist} = 800$. If time is required to handle packets or manage peripherals before interrupts are generated by the stack, t_{ndist} should be set larger than 1500.

To ensure the notification signal is available to the application in the configured time, the following rule is applied:

$$t_{ndist} + t_{evnt} < t_{interval}$$

To ensure this rule is true, the stack may limit the length of a Radio Event, t_{evnt}, thereby reducing the maximum throughput and effecting maximum data rate. ***Figure 9*** shows consecutive Radio Events with Radio Notification and illustrates the limitation in t_{evnt} which may be required to ensure t_{ndist} is preserved.

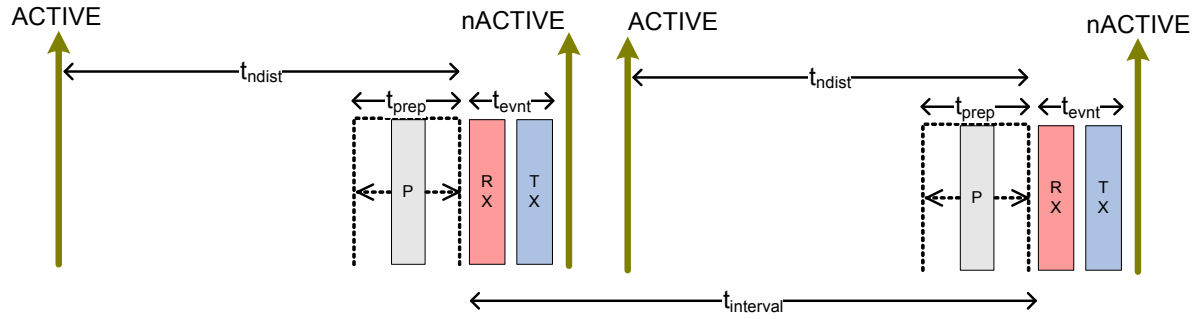

Figure 9 *Consecutive Radio Events with Radio Notification*

Table 15 shows the limitation on the maximum number of packets which can be transferred per Radio Event given a t_{ndist} and $t_{interval}$ combination.

t_{ndist}	$t_{interval}$		
	7.5 ms	**10 ms**	**≥ 15 ms**
800	6	6	6
1740	5	6	6
2680	4	6	6
3620	3	5	6
4560	2	4	6
5500	1	3	6

Table 15 *Maximum packet transfer per Radio Event relative to t_{ndist} and $t_{interval}$.*

10 SoftDevice compatibility and selection

10.1 S110 SoftDevice identification and revision scheme

S110 SoftDevices will be identified by the S110 part code, a qualified IC partcode (for example, nRF51822) and a version number consisting of Major, Minor, and Revision numbers.

For example: S110_nRF51822_1.2.3, where major = 1, minor = 2 and, revision = 3.

Table 16 outlines how version numbers are incremented.

Revision	Description
Major increments	Modifications to the API or the function or behavior of the implementation or part of it have changed. Changes as per Minor Increment may have been made. Application code will not be compatible without some modification.
Minor increments	Additional features and/or API calls are available. Changes as per Revision Increment may have been made. Application code may have to be modified to take advantage of new features.
Revision increments	Issues have been resolved or improvements to performance implemented. Existing application code will not require any modification.

Table 16 *Revision scheme*

Additionally, for revisions of the SoftDevice which are not production qualified, the qualification level, alpha or beta, and sequence number will be appended.

Qualification level	Sequence number
Alpha	s110_nrf51822_1.2.3-1.alpha
Beta	s110_nrf51822_1.2.3-1.beta
Production	s110_nrf51822_1.2.3

The test qualification levels are outlined in ***Table 17.***

Qualification	Description
Alpha	Development release suitable for prototype application development. Hardware integration testing is not complete. Known issues may not be fixed between alpha releases. Incomplete and subject to change.
Beta	Development release suitable for application development. In addition to alpha qualification: Hardware integration testing is complete but may not be feature complete and may contain known issues. Protocol implementations are tested for conformance and interoperability.
Production	Qualified release suitable for product integration. In addition to beta qualification: Hardware integration tested over supported range of operating conditions. Stable and complete with no known issues. Protocol implementations conform to standards.

Table 17 *Test qualification levels*

10.2 Communication of SoftDevice revision updates

When new versions of a SoftDevice become available or the qualification status of a given revision of a SoftDevice is changed, product update notifications will be automatically forwarded, by email, to all users who have a profile configured to receive notifications from the Nordic Semiconductor website.

The S110 SoftDevice will be updated with additional features and/or fixed issues if needed. Supported production versions of the S110 SoftDevice will remain available after updates, so products do not need requalification on release of updates if the previous version is sufficiently feature complete for your product.