S120 nRF51822

Bluetooth® low energy

SoftDevice Specification v0.5

Key Features

- *Bluetooth*® 4.0 compliant low energy single-mode protocol stack
    - Link layer
    - L2CAP, ATT, and SM protocols
    - GATT, GAP, and L2CAP APIs
    - Central and Observer roles - up to 8 simultaneous connections
    - GATT Client and Server
    - SM 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
    - Any RTOS 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

- A4WP wireless charging
- Sports and fitness devices
    - Sports watch
    - Bike computers

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 distributor, please visit 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

Document Status

Status	Description
v0.5	This specification contains target specifications for product development.
v0.7	This specification contains preliminary data; supplementary data may be published from Nordic Semiconductor ASA later.
v1.0	This specification contains final product specifications. Nordic Semiconductor ASA reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

Revision History

Date	Version	Description
November 2013	0.5	Preliminary release.

1 Introduction

The S120 SoftDevice is a *Bluetooth*® low energy (BLE) Central protocol stack solution supporting up to eight simultaneous Central role connections. 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 SoftDevice features and performance.

Note: The SoftDevice features and performance are subject to change between revisions of this document. See ***Section 9.1 "Notification of SoftDevice revision updates"*** on page 29 for more information. To find information on any limitations or omissions, the SoftDevice release notes will contain a detailed summary of the release status.

1.1 Documentation

Document	Description
nRF51 Series Reference Manual	"Appendix A: SoftDevice architecture" in the *nRF51 Series Reference Manual* is essential reading for understanding the resource usage and performance related chapters of this document.
nRF51822 PS	Contains a description of the hardware, modules, and electrical specifications specific to the nRF51822 chip.
Bluetooth Core Specification	The *Bluetooth Core Specification* version 4.0, Volumes 1, 3, 4, and 6 describes *Bluetooth* terminology which is used throughout the SoftDevice Specification.

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 of the function initialize will be written as: *initialize(parameter1, parameter2)*.
- 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 S120 SoftDevice.

2.1 SoftDevice

The S120 SoftDevice is precompiled and linked binary software implementing a *Bluetooth* 4.0 low energy (BLE) protocol stack. The S120 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.

The 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 S120 SoftDevice.

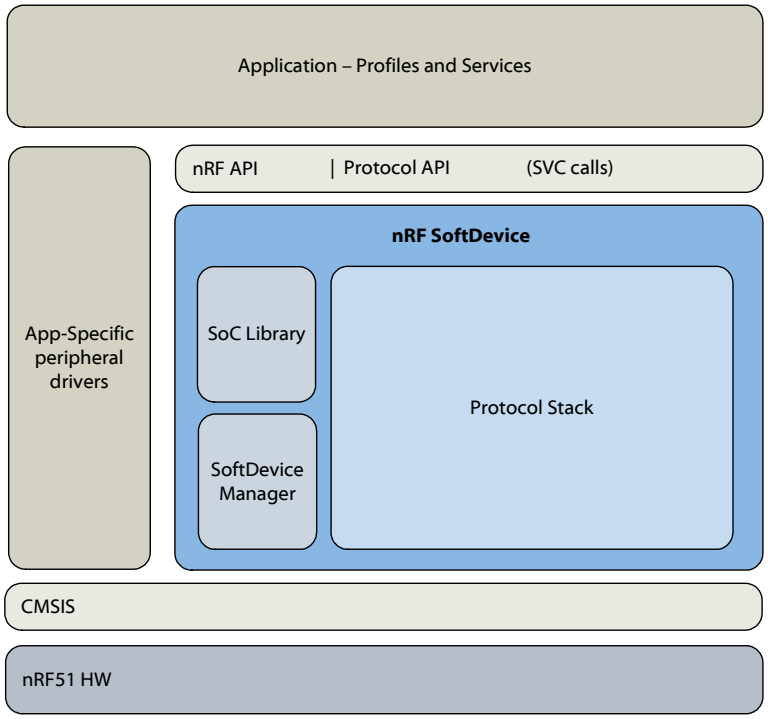

Figure 1 *System on Chip application with the SoftDevice*

The S120 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.

2.2 Multiprotocol support

The S120 SoftDevice supports non-concurrent multiprotocol implementations. This means a proprietary 2.4 GHz protocol can be implemented in the application program area and can access all hardware resources when the SoftDevice is disabled.

3 *Bluetooth* low energy protocol stack

The *Bluetooth* 4.0 compliant low energy (BLE) Host and Controller embedded in the SoftDevice are fully qualified with multi-role support (Central and Observer). The API is defined above the Generic Attribute Protocol (GATT), Generic Access Profile (GAP), and Logical Link Control and Adaptation Protocol (L2CAP). The 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 nRF51xxx integrated circuit (IC).

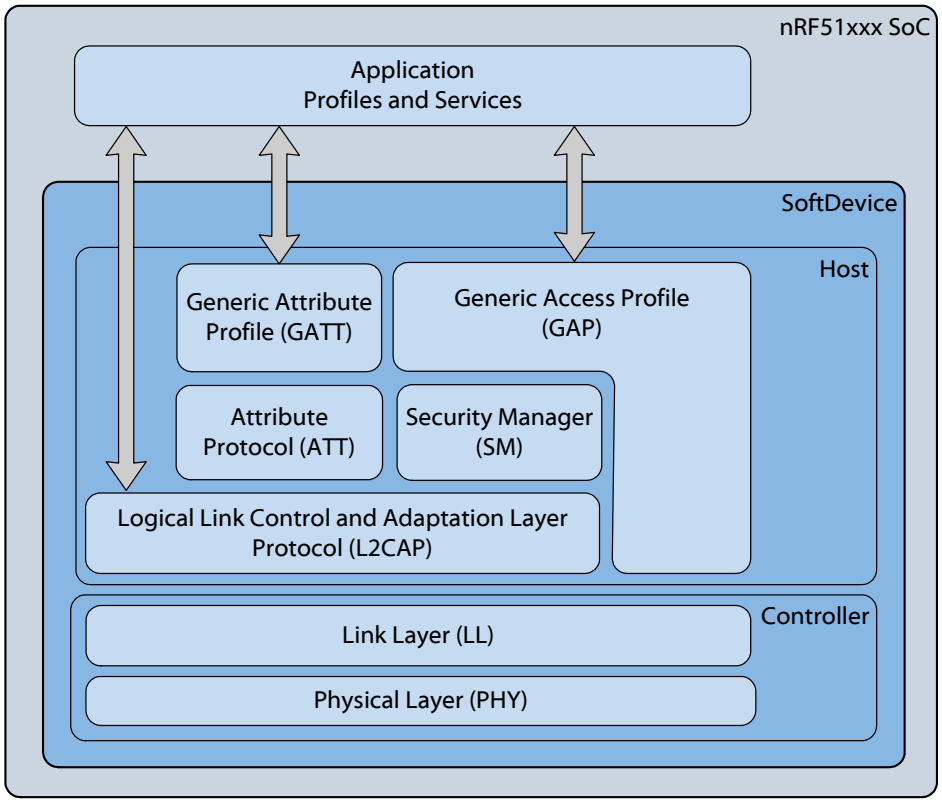

Figure 2 *SoftDevice stack architecture*

3.1 *Bluetooth* low energy features

The BLE protocol stack in the 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 common features have the same interface. The following tables describe the features found in the BLE protocol stack.

API Features	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 multiprotocol	
Packet flow control	Zero-copy buffer management.

***Table 1** API features in the BLE stack*

GAP Features	Description
Multi-role: Central and Broadcaster	
Multiple bond support	Keys and peer information stored in application space. No limitations in stack implementation.
Security mode 1: Levels 1, 2, and 3	No security. Unauthenticated and authenticated pairing with encryption.

***Table 2** GAP features in the BLE stack*

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 Write Long Characteristic Values Write Long Characteristic Descriptors Reliable Writes

Table 3 *GATT features in the BLE stack*

Security Manager Features	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	

Table 4 *Security Manager (SM) features in the BLE stack*

ATT Features	Description
Server protocol	
Client protocol	

Table 5 *Attribute Protocol (ATT) features in the BLE stack*

L2CAP Features	Description
27 byte MTU size	
Dynamically allocated channels	

Table 6 *Logical Link Control and Adaptation Layer Protocol (L2CAP) features in the BLE stack*

Controller, Link Layer Features	Description
Central role Observer/Scanner/Initiator roles	S120 supports concurrent central connections and an additional Observer/ Scanner/Initiator role for new connections. When the maximum number of simultaneous connections are made, the Observer and Scanner roles will be supported for new device discovery though the initiator is not available at that time.
Central connection update	
27 byte MTU	
Encryption	

Table 7 *Controller, Link Layer (LL) features in the BLE stack*

Proprietary Feature	Description
TX Power control	Access for the application to change TX power settings anytime.

Table 8 *Proprietary features in the BLE stack*

3.2 Limitations on procedure concurrency

When there are multiple connections in the Central role, the concurrency of protocol procedures will have some limitations. The Host instantiates both GATT and GAP server instances for each connection, while the SMP server is only instantiated once for all connections. The Link Layer also has concurrent procedure limitations that are handled inside the SoftDevice without requiring management from the application.

The limitations are outlined in ***Table 9*** below.

Protocol procedures	Limitation
GATT	None. All procedures can be executed in parallel.
GAP	None. All procedures can be executed in parallel. Note that some GAP procedures require LL procedures (connection parameter update). In this case, the GAP protocol will queue LL procedures and execute in sequence.
SM	SM procedures cannot be executed in parallel, that is, each SM procedure must run to completion before the next procedure begins across all connections.
LL	LL Disconnect procedure has no limitations and will complete on all links simultaneously. LL connection parameter update can only execute on one link at a time.

Table 9 *Procedure concurrency*

4 SoC library

<Content in this chapter may not be accurate for this release.>

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

Feature	Description
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.
NVIC	Gives the application access to all NVIC features without corrupting SoftDevice configurations.
Rand	Gives access to the random number generator hardware.

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.

Clock	Access to CLOCK block configuration while the SoftDevice is enabled. Allows the HFCLK Crystal Oscillator source to be requested by the application.
Wait for event	Simple power management hook for the application to use to enter a sleep or idle state and wait for an event.
PPI	Configuration interface for PPI channels and groups reserved for an application.
Radio notification	Configure Radio Notification signals on ACTIVE and/or INACTIVE. See ***Section 9.1 "Notification of SoftDevice revision updates"*** on page 29.
Block encrypt (ECB)	Safe use of 128 bit AES encrypt HW accelerator.
Event API	Fetch asynchronous events generated by the SoC library.
Flash memory API	Application access to flash write, erase, and protect. Can also safely be used during active BLE connections.
Temperature	Application access to the temperature sensor.

Table 10 *System on Chip features*

5 SoftDevice Manager

<Content in this chapter may not be accurate for this release.>

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

Feature	Description
SoftDevice control API	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

Table 11 *SoftDevice Manager*

6 SoftDevice resource requirements

The 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 12*** on page 14.

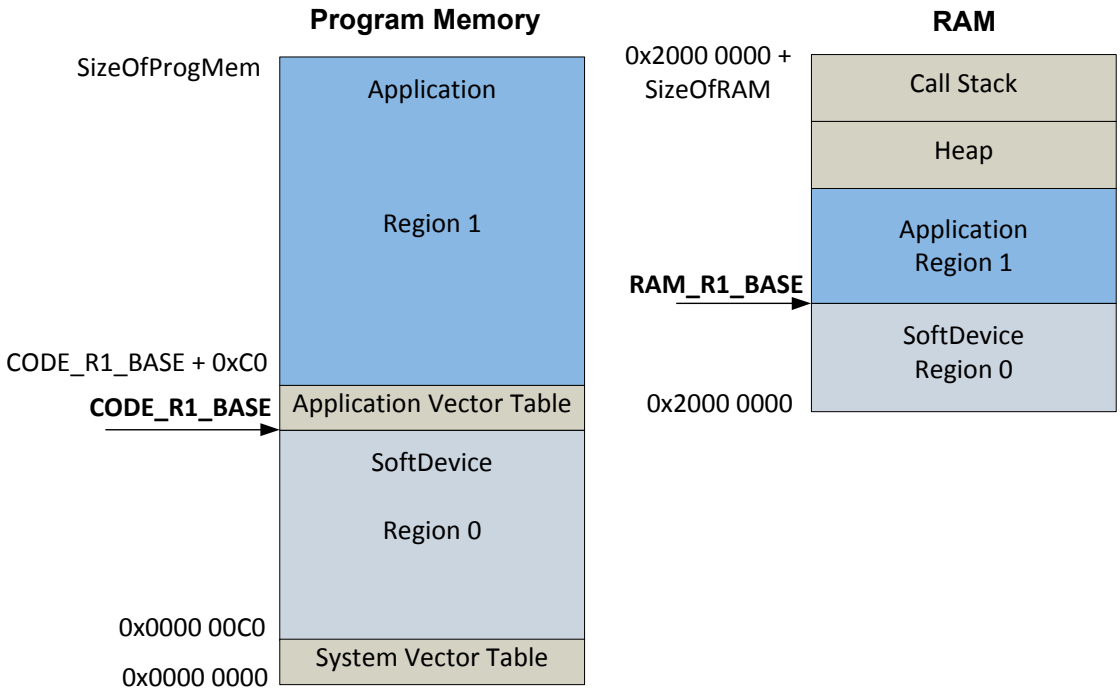

Figure 3 *Memory resource map*

Flash	S120 Enabled	S120 Disabled
Amount	96 kB	96 kB
CODE_R1_BASE	0x0001 8000	0x0001 8000
RAM	**S120 Enabled**	**S120 Disabled**
Amount	10 kB	4 bytes
RAM_R1_BASE	0x2000 2800	0x2000 0004
Call stack[1]	**S120 Enabled**	**S120 Disabled**
Maximum usage	1.5 kB (0x600)	0 bytes (0x00)

Heap	S120 Enabled	S120 Disabled
Maximum allocated bytes	0 bytes (0x00)	0 bytes (0x00)

1. This is only the callstack 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 12 *S120 Memory resource requirements*

6.2 Hardware blocks and interrupt vectors

Table 13 defines access types used to indicate the availability of hardware blocks to the application. ***Table 14*** on page 15 specifies the access the application has, per hardware block, both when the SoftDevice is enabled and disabled.

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

Table 13 *Hardware access type definitions*

ID	Base address	Instance	Access (SoftDevice enabled)	Access (SoftDevice disabled)
0	0x40000000	MPU	Restricted	Open
0	0x40000000	POWER	Restricted	Open
0	0x40000000	CLOCK	Restricted	Open
1	0x40001000	RADIO	Blocked	Open
2	0x40002000	UART0	Open	Open
3	0x40003000	SPI0/TWI0	Open	Open
4	0x40004000	SPI1/TWI1/SPIS1	Open	Open
		...		
6	0x40006000	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	Restricted	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
19	0x40013000	LCOMP	Open	Open
20	0x40014000	Software interrupt	Open	Open
21	0x40015000	SoC Radio Notification Events	Restricted	Open
22	0x40016000	ANT/BLE/SoC Events	Restricted	Open
23	0x40017000	Software interrupt	Restricted[1]	Open
24	0x40018000	Software interrupt	Blocked	Open
25	0x40019000	Software interrupt	Blocked	Open
		...		
30	0x4001E000	NVMC	Restricted	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 15*** on page 16 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 14 *Peripheral protection and usage by SoftDevice*

6.3 Application signals - software interrupts

Software interrupts are used by the SoftDevice to signal the application of events. ***Table 15*** 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	LowerStack processing - not user configurable.
5	25	UpperStack signaling - not user configurable.

Table 15 *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 16*** 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	SoftDevice enabled	SoftDevice disabled
Application	Channels 0 - 7	Channels 0 - 15
SoftDevice	Channels 8 - 15	-

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

Table 16 *PPI channel and group availability*

6.5 SVC number ranges

Table 17 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	SoftDevice enabled	SoftDevice disabled
Application	0x00-0x0F	0x00-0x0F
SoftDevice	0x10-0xFF	0x10-0xFF

Table 17 *SVC number allocation*

7 Multi-link Central role scheduling

The S120 stack supports up to eight Central role connections and an Observer or Scanner simultaneously. An Initiator can only be started if there are less than eight Central roles running.

7.1 Connection timing

The link scheduling system in the S120 SoftDevice adds central link events relative to the first connected link. ***Figure 4*** shows a scenario where there are two links established. C0 events are of the first central connection made and C1 events are of the second connection made. C1 events are initially offset from C0 events by "T" milliseconds. C1 events, in this example, have exactly double the connection interval of C0 events (the connection intervals have a common factor which is "connectionInterval 0"), so the events remain forever offset by Tms.

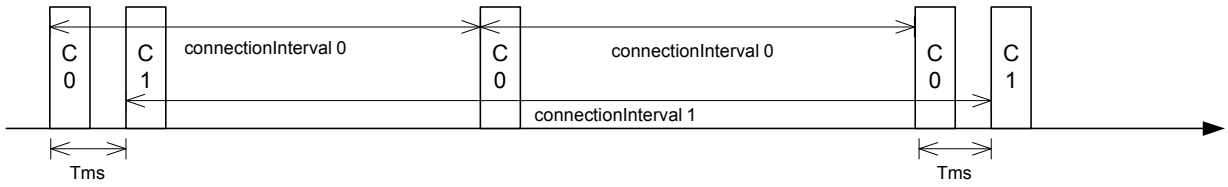

Figure 4 *Multi-link scheduling - one or more connections, factored intervals*

In ***Figure 5*** the connection intervals do not have a common factor. This connection parameter configuration is possible, though this will result in dropped packets when events overlap. In this scenario, the second event shown for C1 is dropped because it overlapped with the C0 event.

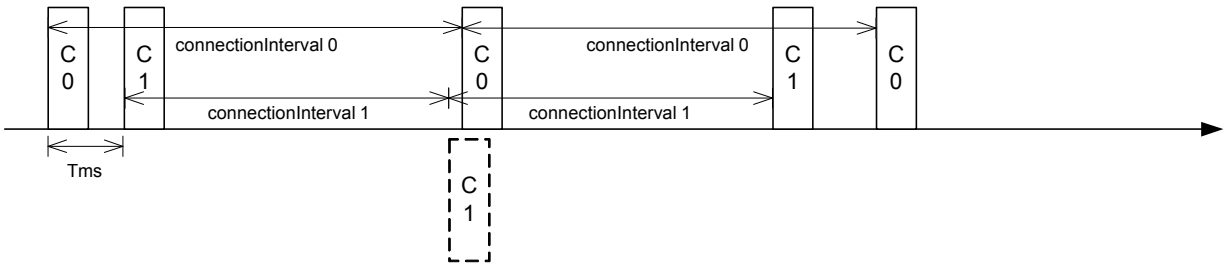

Figure 5 *Multi-link scheduling - one or more connections, unfactored intervals*

Figure 6 shows the maximum number of central links possible at one time (8) with the minimum connection interval possible without having event collisions and dropped packets (20 ms). In this case, all available event time is used for the central links.

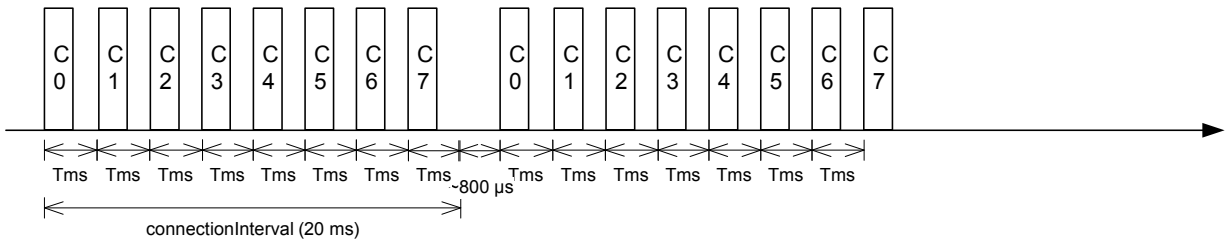

Figure 6 *Multi-link scheduling - max connections, min interval*

Figure 7 shows a scenario where the connInterval is longer than the minimum, and central 2 and 4 have been disconnected or do not have events in this time period. It shows idle event time for each connection interval and the remaining central connections maintain their timing offsets without the other links.

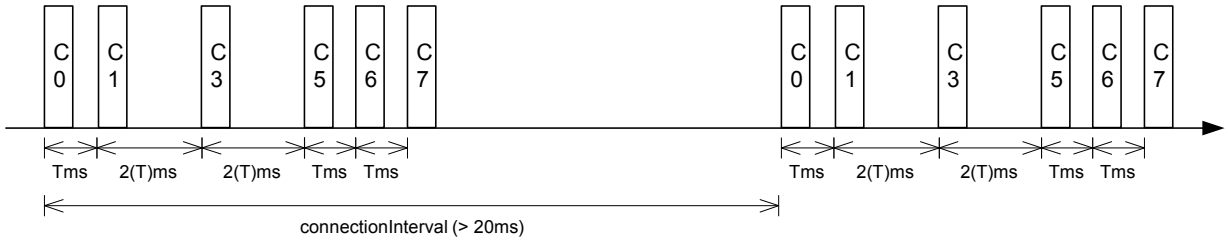

Figure 7 *Multi-link scheduling - max connections, interval > min*

7.2 Scanner timing

Figure 8 shows that when scanning for advertisers with no active connections, the scan interval and window can be any value within the *Bluetooth Core Specification*.

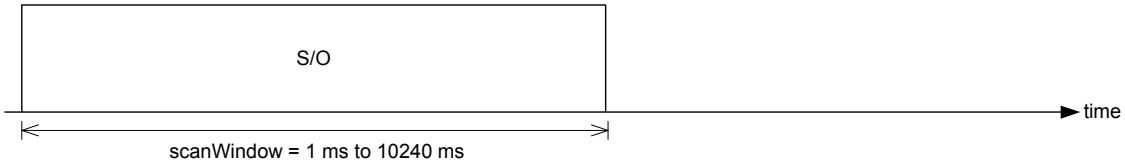

Figure 8 *Scanner timing - no active connections*

Figure 9 shows that when there is an active connection, the scanner or observer role will be started synchronously with the first connected central link at a distance of 8(T)ms. With scanInterval equal to the connectionInterval and a scanWindow ≤ connectionInterval + (8(T)+0.8)ms, scanning will proceed without packet loss.

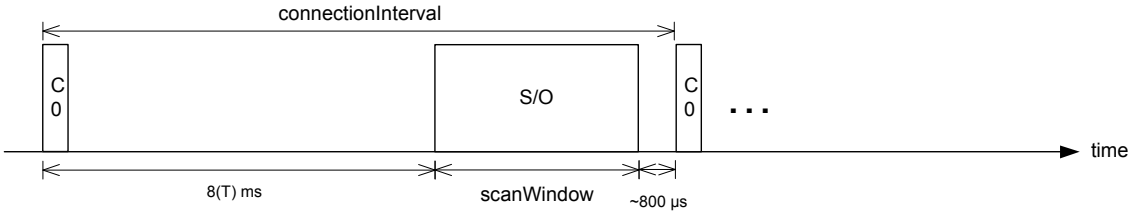

Figure 9 *Scanner timing - one connection*

Figure 10 shows a scanner with a long scanWindow which will cause some connection events to be dropped.

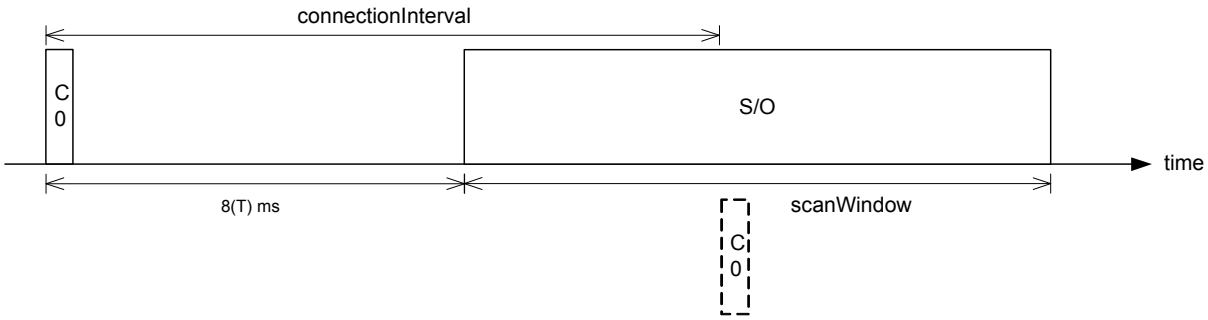

Figure 10 *Scanner timing - one connection, long window*

If all links have a short connection interval (20 ms) and the scanner is started, the scanner events will collide with central link events causing packets on connections to be dropped as shown in ***Figure 11***.

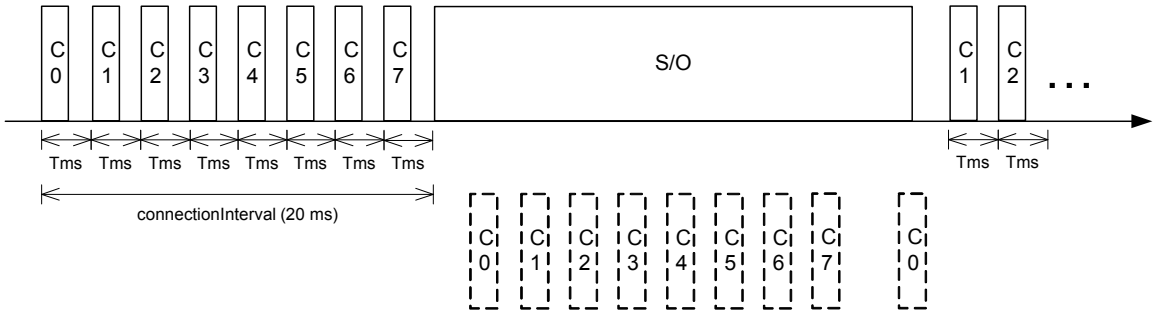

Figure 11 *Scanner timing - minimum connection interval*

7.3 Initiator timing

When making the first connection to a peer, the initiator will make the connection in the minimum time and allocate the first central link connection event 1.25 ms after the connect request was sent as shown in ***Figure 12***.

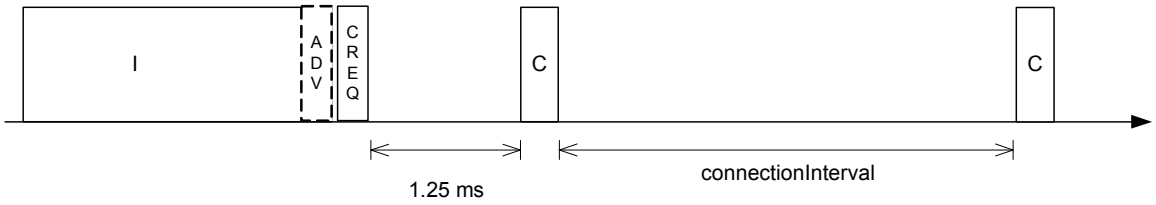

Figure 12 *Initiator - first connection*

When making a new connection with other connections already made, the initiator will start asynchronously to the connected link events and position the new central connection's first event in a free slot between existing events. ***Figure 13*** illustrates this when all existing connections have the same connection interval and the initiator starts around the same time as the 7th Central connection (C6) event in the schedule.

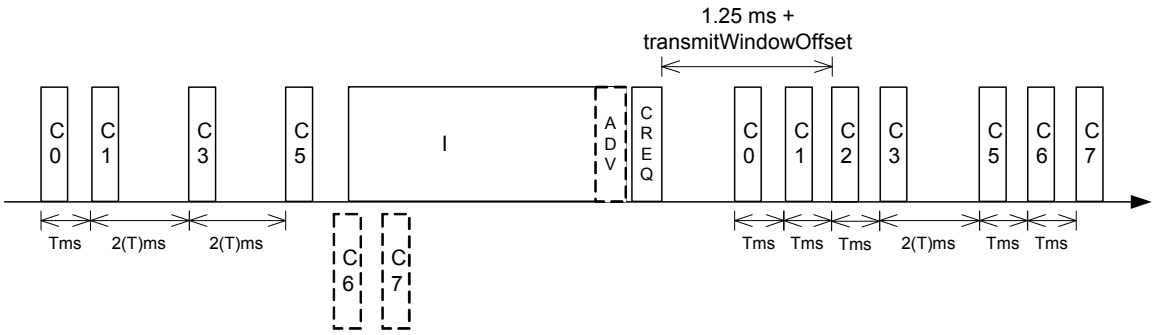

Figure 13 *Initiator - one or more connections*

When making connections to newly discovered devices, the scanner may be used for discovery followed by the initiator. In ***Figure 14***, the initiator is started directly after discovering a new device to connect as fast as possible to that device. The result is some connection events being dropped while the initiator runs. Central events scheduled in the transmit window offset will not be dropped (C3). In this case the 5th peer connection schedule is available (C4), and is allocated for the new connection.

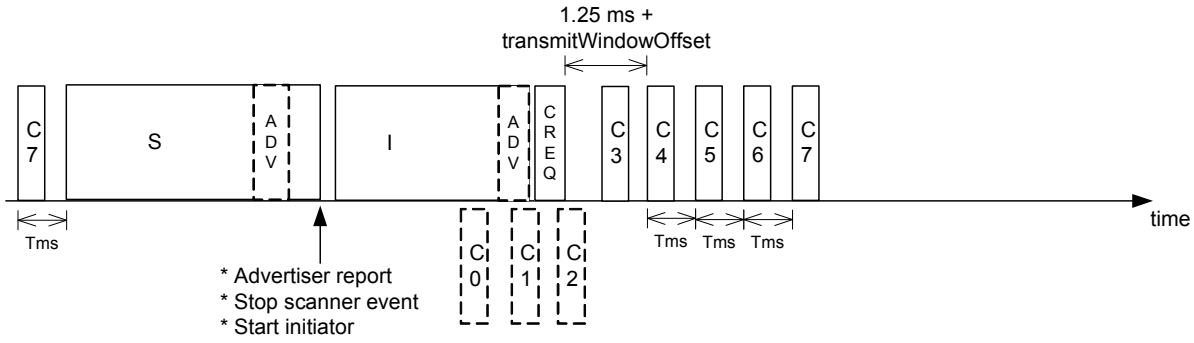

Figure 14 *Initiator - fast connection*

Note: It is not currently possible to schedule the initiator synchronously with connection events. In a future version of the SoftDevice, the initiator may have an option to start either synchronously, like the scanner, or asynchronously to prioritize a new connection.

7.4 Suggested intervals and windows

The value of "T" in the figures shown in previous sections is always 2.4 ms. This allows each central link to send and receive one full length BLE packet before another event starts. There is an additional 800 µs before the first event of the first central connection event in each case as shown in ***Figure 6*** on page 19. Therefore eight link events can complete in a minimum time of 8(2.4)+0.8 = 20 ms.

The minimum connection interval recommended for eight connections is 20 ms. Note that this does not leave time in the schedule for scanning or initiating new connections (when the number of connections already made is less than eight). Scanner, Observer, and Initiator events will therefore cause connection packets to be dropped as in ***Figure 11*** on page 21.

It is recommended that all connections have an interval equal to or some multiple of each other. In the case of using the minimum connection interval, 20 ms, all connections would have an interval of 20 ms or a multiple of 20 ms like 40 ms, 60 ms, 80 ms, etc.

If short connections intervals are not essential to the application an there is a need to have a scanner and/or initiator running at the same time as connections (an initiator will have to be started to make new connections), then it is possible to avoid dropping packets on any connection by having a connection interval of 50 ms or a multiple of 50 ms. In this case, eight connection events and a 30 ms scanner/initiator window can complete within each connection interval as in ***Figure 13*** on page 22.

To summarize, a recommended configuration for operation without dropped packets:

1. The minimum Central connectionInterval should be ≤ 20 ms + scanWindow
2. All Central connections should have a connectionInterval which can be factored by the smallest connection interval. For example [50 ms, 100 ms, 150 ms, 200 ms…] or [75 ms, 150 ms, 225 ms] etc.
3. Scanner, Observer, and Initiator roles should have intervals which can be factored by the smallest connection interval (as with the Central role) and the window should be ≤ connectionInterval – 20 ms.

If the application configures connection intervals between 7.5 ms and 20 ms and/or scan windows larger than recommended, the application should tolerate dropped packets by setting the supervision timeout for connections long enough to avoid loss of connection when packets are dropped because of Scanner, Observer, or Initiator events.

8 BLE performance

This chapter documents key SoftDevice performance parameters for interrupt latency, processor availability, and data throughput with regards to the BLE stack.

Note: The performance of the S120 SoftDevice has been estimated from preliminary test results at this time; however, these values are subject to change between the alpha and production versions of the SoftDevice. The values here should serve as an indication of performance.

8.1 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 8.2 "Processor availability"*** on page 25.

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 18 *Additional latency due to SoftDevice processing*

See ***Table 14*** on page 15 for open, blocked, and restricted peripherals.

8.2 Processor availability

"Appendix A: SoftDevice architecture" in the *nRF51 Reference Manual* describes interrupt management in SoftDevices and is required knowledge for understanding this section.

Shown in ***Figure 15*** are the parameter values from ***Table 19***. The parameters are defined around LowerStack and UpperStack interrupts. These interrupts service real time protocol events and API calls (or deferred internal SoftDevice tasks) respectively. LowerStack interrupts are extended by a CPU Suspend state during radio activity to improve link integrity. This means LowerStack interrupts will block application and UpperStack processing during a Radio Event for a time proportional to the number of packets transferred in the event. See ***Table 20*** on page 26 for more information.

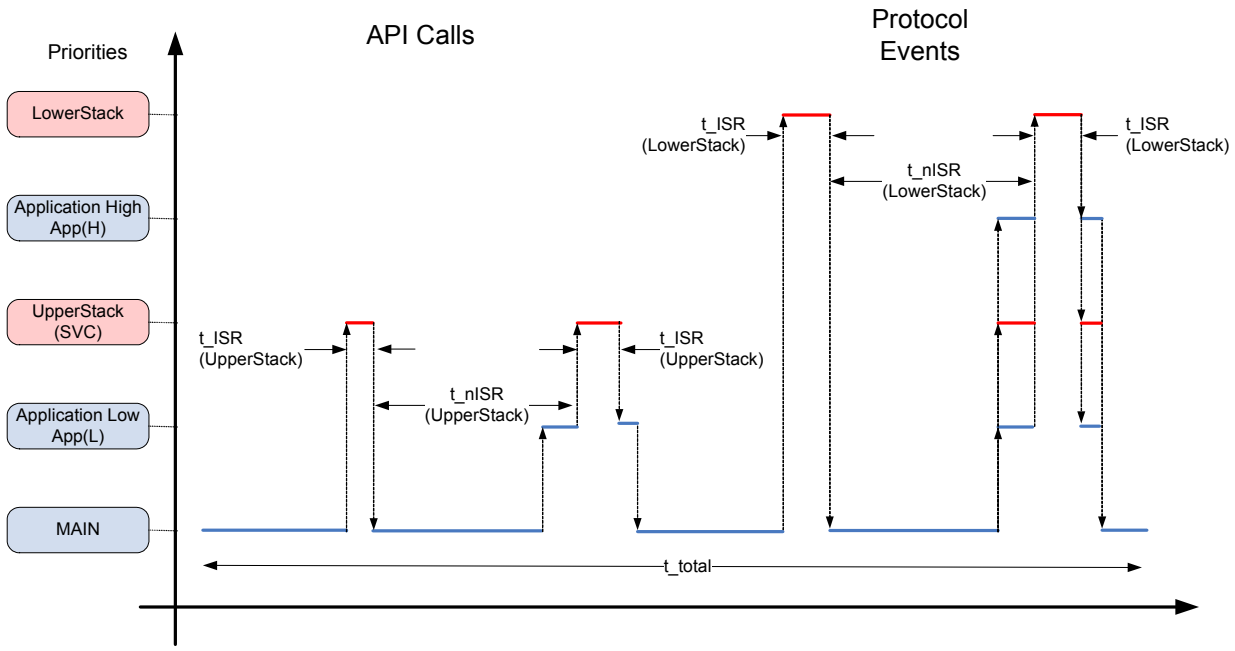

Figure 15 *Interrupt latencies due to SoftDevice processing*

		UpperStack		
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 19 *SoftDevice interrupt latency - UpperStack*

Parameter	Description	Packets	Nominal
$t_{ISR(lower\ stack)1}$	Maximum interrupt latency during Radio Event. Includes the time the CPU is used by the LowerStack 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
$tn_{ISR(lower\ stack)}$	Minimum time between LowerStack interrupts.	n/a	800 µs

Table 20 *SoftDevice interrupt latency LowerStack*

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 21 shows expected CPU utilization percentages for the UpperStack and LowerStack given a set of typical stack connection parameters.

Note: UpperStack 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	LowerStack	UpperStack	CPU suspend	Remaining
Connection interval 4 s 8 connections No data transfer	TBD%	TBD%	TBD%	TBD%
Connection interval 20 ms 8 connections 1 packet transfer per event (bidirectional)	20%	15%	30%	~35%
Connection interval 100 ms 1 packet transfer per event (bidirectional)	5%	5%	10%	~80%

Table 21 *Processor usage and remaining availability for example BLE connection configurations*

8.3 Data throughput

The maximum data throughput limits in ***Table 22*** 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	TBD kbps
		Send	TBD kbps
		Simultaneous send and receive	TBD kbps (each direction)
GATT	Client	Receive Notification	10 kbps
		Send Write command	10 kbps
		Send Write request	1 kbps
		Simultaneous receive Notification and send Write command	10 kbps (each direction)
GATT	Server	Send Notification	10 kbps
		Receive Write command	10 kbps
		Receive Write request	1 kbps
		Simultaneous send Notification and receive Write command	10 kbps (each direction)

Table 22 *L2CAP and GATT maximum data throughput*

9 SoftDevice identification and revision scheme

The SoftDevices will be identified by the SoftDevice part code, a qualified IC partcode (for example, nRF51822), and a version string.

For revisions of the SoftDevice which are production qualified, the version string consists of major, minor, and revision numbers only, as described in ***Table 23***.

For revisions of the SoftDevice which are not production qualified, a build number and a test qualification level (alpha/beta) are appended to the version string.

For example: S110_nRF51822_1.2.3-4.alpha, where major = 1, minor = 2, revision = 3, build number = 4 and test qualification level is alpha. Additional SoftDevice revision examples are given in ***Table 24***.

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.
Build number increment (if present)	New build of non-production version.

Table 23 *Revision scheme*

Sequence number	Description
s110_nrf51822_1.2.3-1.alpha	Revision 1.2.3, first build, qualified at alpha level
s110_nrf51822_1.2.3-2.alpha	Revision 1.2.3, second build, qualified at alpha level
s110_nrf51822_1.2.3-5.beta	Revision 1.2.3, fifth build, qualified at beta level
s110_nrf51822_1.2.3	Revision 1.2.3, qualified at production level

Table 24 *SoftDevice revision examples*

The test qualification levels are outlined in ***Table 25***.

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 25 *Test qualification levels*

9.1 Notification 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 SoftDevice will be updated with additional features and/or fixed issues if needed. Supported production versions of the 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.