S210 nRF51422

ANT™

SoftDevice Specification v1.2

Key Features

- Embedded ANT stack
  - Simple to complex network topologies:
    - Peer-to-peer
    - Star
    - Tree
    - Star-to-star and more
  - Broadcast, acknowledged, and burst communication modes.
  - Supports up to 8 logical channels each with configurable channel periods (5.2 ms - 2 s).
  - Built-in device search and pairing.
  - Built-in interference handling and radio coexistence management with application radio disable requests and application flash write/erase requests.
  - Enhanced ANT features:
    - Advanced burst transfer modes (up to 60 kbps)
    - Optional single channel encryption (AES-128)
    - Supports up to 8 public, private, and/or managed networks.
    - Advanced power management features to optimize application power consumption including Event Filtering and Selective Data Updates.
    - Asynchronous transmit channel
    - Fast channel initiation
- Asynchronous event-driven behavior
- Low link-time dependencies
- No RTOS dependency - you can choose which RTOS to use
- Safe application and protocol coexistence through memory isolation
- Standard ARM® Cortex™ -M0 project configuration for application development

Applications

- Personal area networks:
  - Sport and fitness sensors and monitoring devices
  - Healthcare and lifestyle sensors
  - Personal convenience devices, key fobs, and remote controls
- Environmental sensor networks/high density wireless networking and monitoring
- Logistics/goods tracking
- Smart RF tags

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
January 2014	1.2	**Updated:** • ***Table 1 "System on Chip features"*** on page 9 **Removed:** • ***Figure 4 "Radio Notification"*** on page 14 • ***Figure 5 "Radio Notification, multiple packet transfers"*** on page 15 • ***Figure 6 "Consecutive Radio Events with Radio Notification"*** on page 17 • ***Table 6 "Radio Notification figure labels"*** on page 16 • ***Table 7 "Radio Notification timing ranges"*** on page 16
December 2013	1.1	**Updated:** • ***Figure 6 "Interrupt latencies due to SoftDevice processing"*** on page 21 • ***Figure 7 "ANT Master Transmit Channel"*** on page 25 • ***Figure 8 "ANT Slave Receive Channel"*** on page 27 • ***Table 7 "S210 Memory resource requirements"*** on page 17 • ***Table 14 "LowerStack interrupt latency numbers"*** on page 23 • ***Table 15 "UpperStack interrupt latency numbers"*** on page 24 **Added:** • ***Chapter 6 "Flash memory API"*** on page 11 • ***Chapter 7 "Radio Notification"*** on page 13 • ***Chapter 8 "Bootloader"*** on page 15
December 2012	1.0	**Updated:** • ***Figure 2 "ANT stack architecture"*** on page 6 • ***Figure 6 "Interrupt latencies due to SoftDevice processing"*** on page 21 • ***Section 3.2 "ANT profile and feature support"*** on page 7 • ***Table 7 "S210 Memory resource requirements"*** on page 17 **Added:** • ***Table 14 "LowerStack interrupt latency numbers"*** on page 23 • ***Table 15 "UpperStack interrupt latency numbers"*** on page 24
October 2012	0.6.1	Updated flash size of the S210 SoftDevice to 40 kb.
September 2012	0.6	Preliminary release.

1 Introduction

The S210 SoftDevice is an ANT protocol stack solution that provides a full, flexible application programming interface (API) for building ANT System on Chip (SoC) solutions for Nordic Semiconductor nRF51422 ICs. The S210 SoftDevice simplifies combining the ANT protocol stack and an application on the same CPU.

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 12.1 "Notification of SoftDevice revision updates"*** on page 30 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.
nRF51422 PS	Contains a description of the hardware, modules, and electrical specifications specific to the nRF51422 chip.
nRF51422 PAN	Contains information on limitations related to the SoftDevice.
ANT Message Protocol and Usage	The *ANT Message Protocol and Usage* document describes the ANT protocol in detail and is the starting point for understanding everything else. It contains the fundamental knowledge you need in order to develop successfully with ANT.

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 S210 SoftDevice.

2.1 SoftDevice

The S210 SoftDevice is precompiled and linked binary software implementing a full ANT protocol stack. S210 is compatible with the nRF51422 SoC device. 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 ANT/ANT+ applications with the S210 SoftDevice.

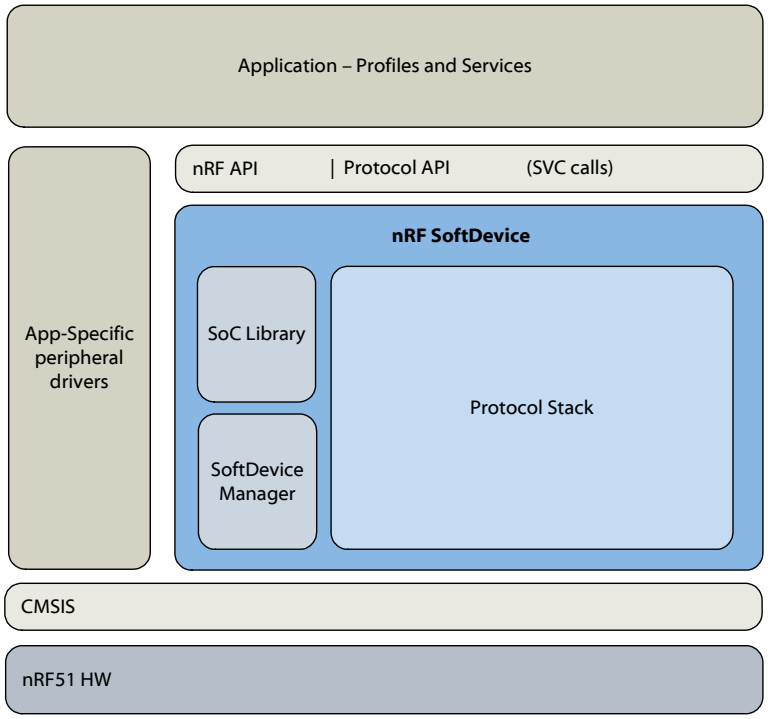

Figure 1 *System on Chip application with the SoftDevice*

The S210 SoftDevice can be programmed onto compatible nRF51 devices during both development and production. This specification outlines the supported features of a Production level SoftDevice. Alpha and beta versions may not support all features.

2.2 Multiprotocol support

The S210 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 ANT protocol Stack

The SoftDevice is a fully qualified ANT compliant stack that supports all ANT core functionality, including ANT+ profiles. See http://thisisant.com for further information regarding ANT.

Note: ANT+ implementations must pass the ANT+ certification process to receive ANT+ Compliance Certificates.

3.1 Overview

The nRF51 Software Development Kit (SDK) supplements the ANT protocol stack with complete peripheral drivers, example applications, and ANT+ profile implementations.

Note: ANT+ network keys are needed to make ANT+ compliant products. The keys can be obtained by registering as an ANT+ adopter at http://thisisant.com.

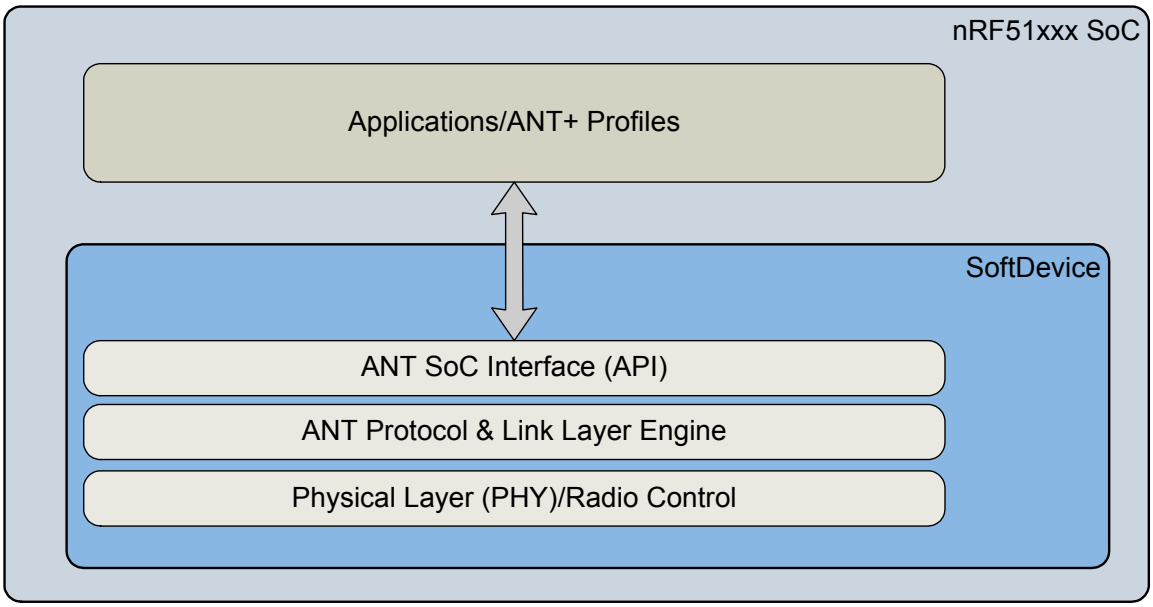

Figure 2 *ANT stack architecture*

3.2 ANT profile and feature support

All ANT nRF24AP1 and nRF24AP2 profiles and features are fully supported and can be found in "Table 4 ANT message summary supported by nRF24AP2" of the *nRF24AP2 Product Specification*. Additional enhanced ANT features are available in the SoftDevice and are described below.

3.2.1 Advanced Burst Transfer

Advanced Burst Transfer is intended to facilitate and improve the ability of devices to transfer information. This is employed primarily in ANT-FS use cases, with an emphasis on longer file transfers. Advanced Burst Transfer improves the efficiency of transferring a file by increasing the transfer speed (up to 60 kbps) while providing greater immunity to RF interference. Burst stalling and adjustable retry mechanisms are introduced to allow greater flexibility of host data source and sink rates.

Note: The peer device must also support this feature to achieve higher transfer rates. This feature is backward compatible with existing burst transfer methods (up to 20 kbps).

3.2.2 Single channel encryption

The ANT channel encryption engine provides the ability for a single channel to transmit and receive data in a secure manner using AES-128. This feature simplifies applications that require data security such as medical and health devices. Broadcast messaging, acknowledged messaging, and burst transfers/advanced burst transfers are supported data communication modes for encryption.

ANT encryption mode allows broadcast data to be received by multiple receivers. Alternatively, point-to-point encrypted links can be created through the use of inclusion/exclusion encryption ID list.

3.2.3 Multiple network keys

For use cases that benefit from transmission of data on multiple networks, the SoftDevice will support use of up to 8 public, private, and/or managed (ANT+) network keys.

3.2.4 Event filtering

Event filtering is provided for the application to avoid or reduce processing of ANT events, in an effort to conserve power consumption and/or resources. The application can select which event messages to block from the ANT protocol layer to the application.

3.2.5 Selective Data Updates

Selective Data Updates is another feature intended to aid in managing power consumption of the application. The facility may be used when an application needs to process received messages only if the specified data has changed. This could apply to devices such as display units that update the display only when the displayed data has actually changed and are otherwise asleep. This feature can be enabled for Broadcast messages only or Broadcast and Acknowledged messages. This feature does not apply to Burst Transfers.

3.2.6 Asynchronous Transmission channel

Asynchronous Transmission mode introduces the ability for users to initiate transmissions that are not bound by any specified periods or intervals. Asynchronous channels do not need to be “opened” and are simply initiated by sending data to the channel. Asynchronous channels may be used when user generated input on a device needs to cause data transmission to a receiver with minimal delay (for example remote control applications). Receivers must employ scanning channels to effectively use this feature. Broadcast messaging, acknowledged messaging, and burst transfers are supported in this mode.

3.2.7 Fast Channel Initiation

Enabling the Fast Channel Initiation feature allows ANT synchronous transmission channels to start as soon as possible. The latency from the channel opening to transmission is reduced. This feature should only be used in scenarios where a device opens for a brief period of time and requires low latency.

4 SoC library

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 disable API	Requests short periods of guaranteed radio inactivity for application activity.
Radio notification	Configure Radio Notification signals on ACTIVE and/or INACTIVE. See ***Section 12.1 "Notification of SoftDevice revision updates"*** on page 30.
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 ANT activities.
Temperature	Application access to the temperature sensor.

Table 1 *System on Chip features*

5 SoftDevice Manager

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 clock • Synthesized from high frequency clock • Crystal oscillator

***Table 2** SoftDevice Manager*

6 Flash memory API

Asynchronous flash memory operations are performed using the SoC library API and provide the application with flash write, flash erase, and flash protect support through the SoftDevice. This interface can safely be used during ANT radio activities.

The flash memory access is scheduled in between the protocol radio events. The time required for memory access may be larger than is allowed for certain ANT operations.

- ANT Burst Transfers - In this case, flash operations may be denied or delayed until the ANT burst transfer is finished.
- ANT Transmit / Receive Keep Alive – These critical ANT transmit/receive radio operations may delay flash operation requests.

ANT activity	Flash write
ANT RX Search/Scanning ANT TX/RX Broadcast ANT TX/RX Acknowledged	Typically allows full write size (256 words) attempts. May generate flash timeout event: `NRF_EVT_FLASH_OPERATION_ERROR` if critical radio activities needs to occur. In this case, retry flash write.
ANT TX/RX Burst Transfer	• Maximum flash write size attempt for concurrent operation: 48 words. Up to 30% reduction of burst transfer rate during continuous flash write activity. May generate flash timeout event: `NRF_EVT_FLASH_OPERATION_ERROR` if critical radio activities needs to occur. In this case, retry flash write. • Larger flash write size attempts, in general, will consistently generate flash timeout event: `NRF_EVT_FLASH_OPERATION_ERROR` until the burst transfer has ended. In this case, retrying flash writes will only succeed after the burst activity has finished.
ANT activity	**Flash erase**
ANT RX Search/Scanning ANT TX/RX Broadcast ANT TX/RX Acknowledged	Typically allows flash erase attempts. May generate flash timeout event: `NRF_EVT_FLASH_OPERATION_ERROR` if critical radio activities needs to occur. In this case, retry flash erase.
ANT TX/RX Burst Transfer	Flash erase attempts, in general, will consistently generate flash timeout event: `NRF_EVT_FLASH_OPERATION_ERROR` until the burst transfer has ended. In this case, retrying flash erases will only succeed after the burst activity has finished.

Table 3 *ANT flash write/erase*

6.1 Radio disable API

The SoC library API provides a set of interfaces in which the application can request short periods of time where radio activity will not occur. This is useful in particular situations where the application may need to reserve guaranteed time for activities and cannot run/rely on signals from SoC Radio Notification.

Similar to Flash Memory API, if the radio disable request is larger than allowed for certain ANT operations, then the operation may be delayed or denied. If the radio disable session is denied, the application is notified of this occurrence via NRF_EVT_RADIO_BLOCKED or NRF_EVT_RADIO_CANCELED event. The application may choose to retry the operation after receiving these events. Judicious use of this feature is required as this can impact overall ANT radio protocol performance.

ANT activity	Radio disable
ANT RX Search	Typically allows up to 20 ms radio disable sessions. May generate session blocked event NRF_EVT_RADIO_BLOCKED or NRF_EVT_RADIO_CANCELED if critical radio activities needs to occur. Frequent radio disable sessions can impact search performance.
ANT RX Scanning	Will generally not allow any radio disable sessions.
ANT TX/RX Broadcast ANT TX/RX Acknowledged	Typically allows up to full duration (100ms) radio disable sessions. May generate session blocked event NRF_EVT_RADIO_BLOCKED or NRF_EVT_RADIO_CANCELED if critical radio activities needs to occur. Frequent radio disable sessions can impact broadcast/acknowledged message throughput.
ANT TX/RX Burst Transfer	• Maximum radio disable session size for concurrent operation: 1.25 ms. Up to 50% reduction of burst transfer rate during continuous radio disable requested sessions. May generate session blocked event NRF_EVT_RADIO_BLOCKED or NRF_EVT_RADIO_CANCELED if critical radio activities need to occur. • Larger requested radio disable session time, in general, will consistently generate session blocked event NRF_EVT_RADIO_BLOCKED or NRF_EVT_RADIO_CANCELED until the burst transfer has ended. In this case, radio disable session can only occur outside of ANT burst transfers.

Table 4 *ANT radio disable*

6.2 ANT Radio coexistence configuration

ANT provides set of APIs to help manage ANT radio coexistence scheduling with respect to Flash Write and Radio Disable Requests from the application. By default, these settings should not be changed unless very specific coexistence behavior is required.

7 Radio Notification

The Radio Notification is a configurable feature which enables ACTIVE and INACTIVE (nACTIVE) signals from the SoftDevice to the application to notify when the Radio will be in use. The signal is sent using software interrupt, as specified in ***Table 10*** on page 19.

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 manage peak current drawn during periods where the radio is on.

The previously supported feature, ANT RFActive Notification, is still available for use through the ANT API set. However, it is recommended that the SoC Radio Notification be used instead due to the following reasons:

- SoC radio notification uses dedicated software interrupt for immediate notification to the application. ANT RFActive Notification uses the ANT event queue for notification and could be subjected to application event processing delay.
- Notifications in concurrent multi-protocol radio solutions are properly handled by the SoC Radio Notification feature. ANT RFActive Notification only applies to ANT radio events.

Figure 3 shows the active signal in relation to ANT radio activities where t_{ndist} is the time between ACTIVE and the first TX or RX activity of an ANT event.

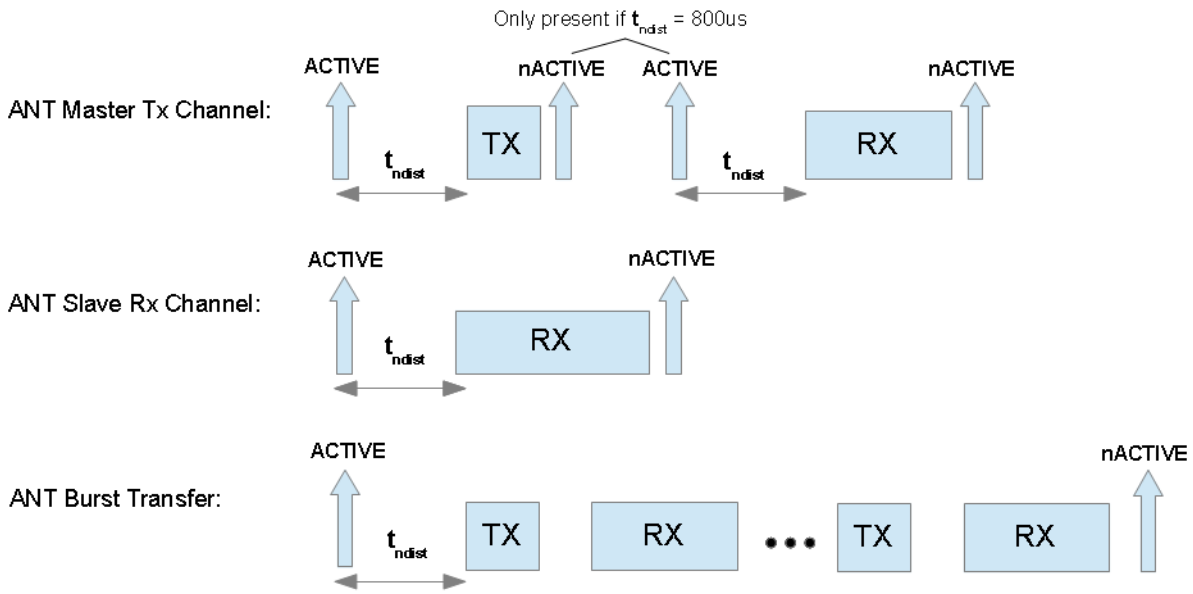

Figure 3 *Radio Notifications*

Note:
- Flash write/erase operations scheduled through the Flash Memory API will generate radio notifications if enabled.
- Radio disable sessions scheduled through the Radio Disable API will generate radio notifications if enabled.

Table 5 lists the acceptable/recommended t_{ndist} values for ANT activities in combination with other scheduler requested activities (for example, flash write/flash erase/radio disable session).

ANT activity	Concurrent operations			
	None	**Flash write and/or erase session**	**Radio disable session**	**Flash write and/or erase session + Radio disable session**
ANT RXSearch/ Scanning	t_{ndist} = 800, 1740, 2680, 3620, 4560, 5500	t_{ndist} = 800	t_{ndist} = 800	t_{ndist} = 800
ANT TX/RX Broadcast	t_{ndist} = 800, 1740, 2680, 3620, 4560, 5500	t_{ndist} = 800	t_{ndist} = 800	t_{ndist} = 800
ANT TX/RX Acknowledged	t_{ndist} = 800, 1740, 2680, 3620, 4560, 5500	t_{ndist} = 800	t_{ndist} = 800	t_{ndist} = 800
ANT TX/RX Burst Transfer	t_{ndist} = 800, 1740, 2680, 3620, 4560, 5500	t_{ndist} = 800	t_{ndist} = 800	t_{ndist} = 800

Table 5 *Acceptable radio notification configuration distances, t_{ndist}*

Note: Extremely fast ANT channel intervals (for example > 200 Hz) may not generate any radio notifications and are treated as one continuous radio event.

8 Bootloader

The SoftDevice supports the use of a bootloader. A bootloader has access to the full SoftDevice API and can be implemented just as any other application that uses a SoftDevice. In particular, the bootloader can make use of the SoftDevice API to enable protocol stack interaction.

The use of a bootloader is supported in the SoftDevice architecture by dividing the application code space region (R1) into two separate regions. The lower region, from CLENR0 and upwards, contains the application, while the upper region contains the bootloader. The start of the upper region, the bootloader's base address, is set by the UICR.BOOTADDR register.

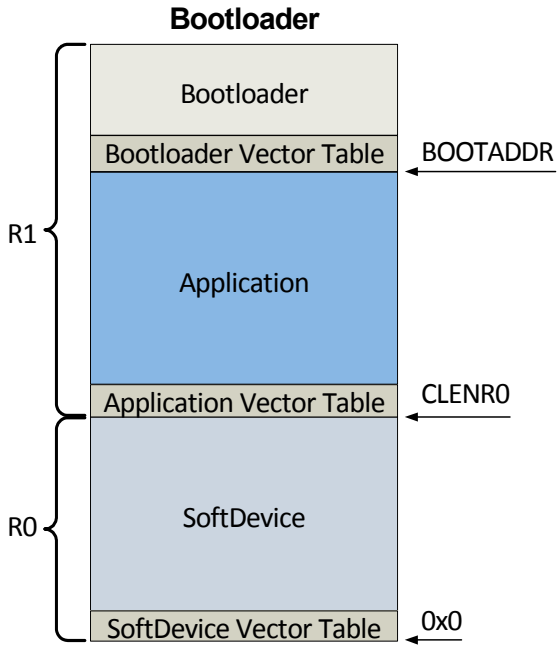

Figure 4 *R0 (SoftDevice) and R1 (Application + Bootloader).*

At reset, the SoftDevice checks the UICR.BOOTADDR register. If this register is blank (0xFFFFFFFF), the SoftDevice assumes that no bootloader is present. It then forwards interrupts to the application and executes the application as usual. If the BOOTADDR register is set to an address different from 0xFFFFFFFF, the SoftDevice assumes that the bootloader vector table is located at this address. Interrupts are then forwarded to the bootloader at this address and execution will be started at the bootloader reset handler.

For a bootloader to transfer execution from itself to the application, the bootloader should first call the `sd_softdevice_forward_to_application()` SoC function to forward interrupts to the application instead of to the bootloader. The bootloader should then branch to the application's reset handler after reading the address of the handler from the Application Vector Table.

UICR.BOOTADDR contents	Interpretation
0xFFFFFFFF	No bootloader present or enabled.
<ADDR>	Bootloader present with base address <ADDR>.

Table 6 *UICR.BOOTADDR register contents*

9 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.

9.1 Memory resource map and usage

The memory map for program memory and RAM at run time with the SoftDevice enabled is illustrated in ***Figure 5*** below. Memory resource requirements, both when the SoftDevice is enabled and disabled, are shown in ***Table 7*** on page 17.

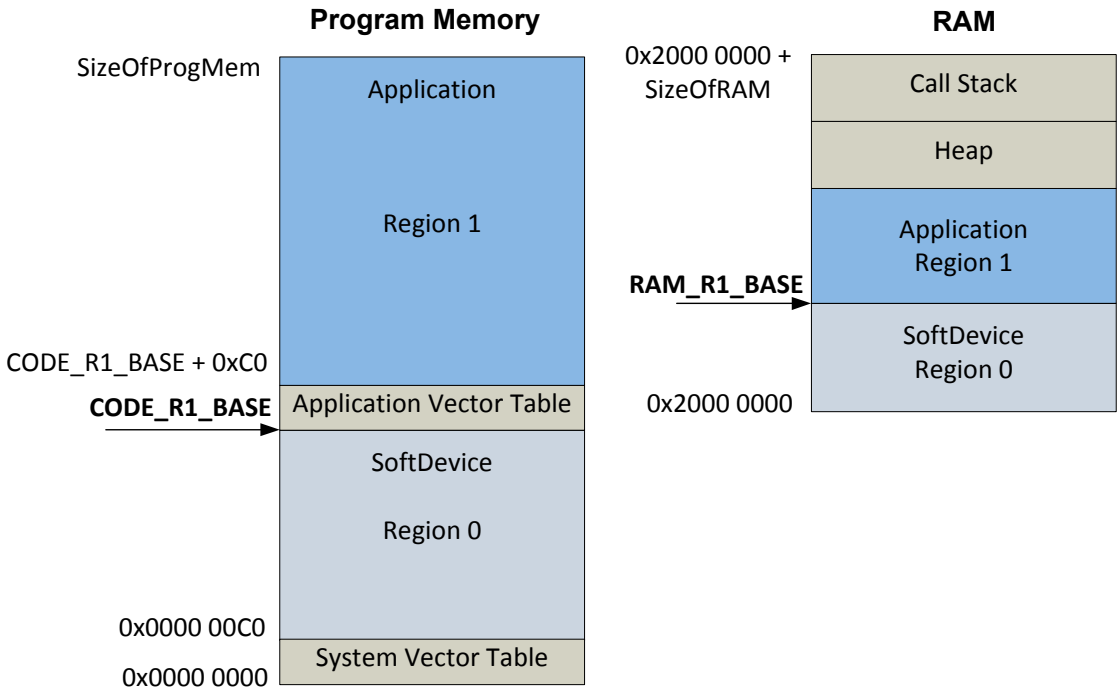

Figure 5 *Memory resource map*

Flash	S210 Enabled	S210 Disabled
Amount	48 kB	48 kB
CODE_R1_BASE	0x0000 C000	0x0000 C000

RAM	S210 Enabled	S210 Disabled
Amount	2.25 kB	4 bytes
RAM_R1_BASE	0x2000 0900	0x2000 0004

Callstack[1]	S210 Enabled	S210 Disabled
Maximum usage	1 kb	0 bytes

Heap	S210 Enabled	S210 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 7 *S210 Memory resource requirements*

9.2 Hardware blocks and interrupt vectors

Table 8 defines access types used to indicate the availability of hardware blocks to the application. ***Table 9*** on page 18 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 8 *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	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	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	00x4001300	LCOMP	Open	Open
20	0x40014000	Software interrupt	Open	Open
21	0x40015000	SoC Radio Notification Events	Blocked	Open
22	0x40016000	ANT/SoC Events	Blocked	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	Open	Open
NA	0xE000E100	NVIC	Restricted[2]	Open

1. Blocked only when signals are configured. See ***Table 10*** on page 19 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 9 *Peripheral protection and usage by SoftDevice*

9.3 Application signals - software interrupts

Software interrupts are used by the SoftDevice to signal the application of events. ***Table 10*** 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 10 *Software interrupt allocation*

9.4 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 11*** 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 11 *PPI channel and group availability*

9.5 SVC number ranges

Table 12 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 12 *SVC number allocation*

10 ANT performance

This section documents key SoftDevice performance parameters for interrupt latency, processor availability, and data throughput with respect to the ANT stack only.

10.1 Interrupt latency

Latency, additional to ARM Cortex™-M0 hardware architecture latency, is introduced by SoftDevice logic to manage interrupt events.

Interrupt	Additional latency in CPU cycles
Open peripheral interrupt	50
Blocked or restricted peripheral interrupt (only forwarded when SoftDevice disabled)	63
Application SVC interrupt	14

10.2 Processor availability

Figure 6 illustrates the parameter values in ***Table 15*** on page 24. The "SoftDevice architecture" chapter of the *nRF51 Reference Manual* describes exception (interrupt) management in SoftDevices and is required knowledge for this section.

The parameters of interest are defined around LowerStack and UpperStack exceptions. These exceptions process real time protocol events and API calls (or deferred internal SoftDevice tasks) respectively. The following figure applies only to ANT SVC and ANT protocol events in the SoftDevice.

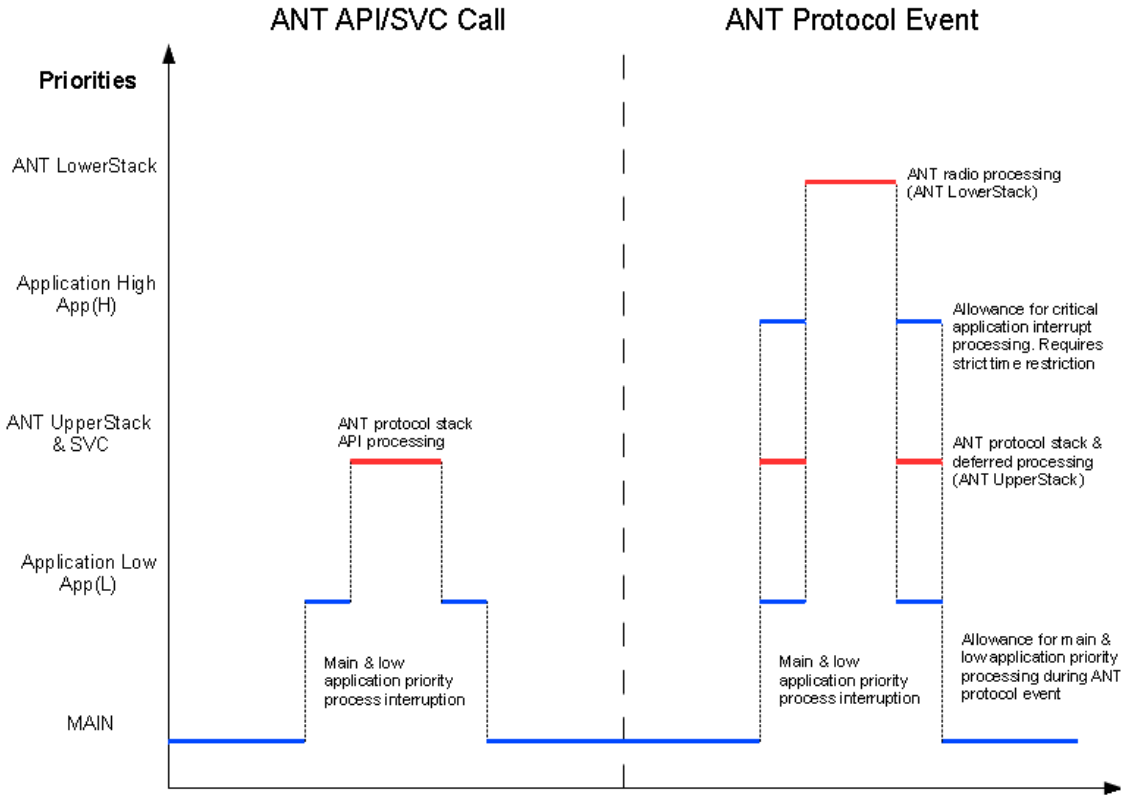

Figure 6 *Interrupt latencies due to SoftDevice processing*

10.3 Processor availability - ANT Protocol Event

Applies to both ANT TX and RX event activities.

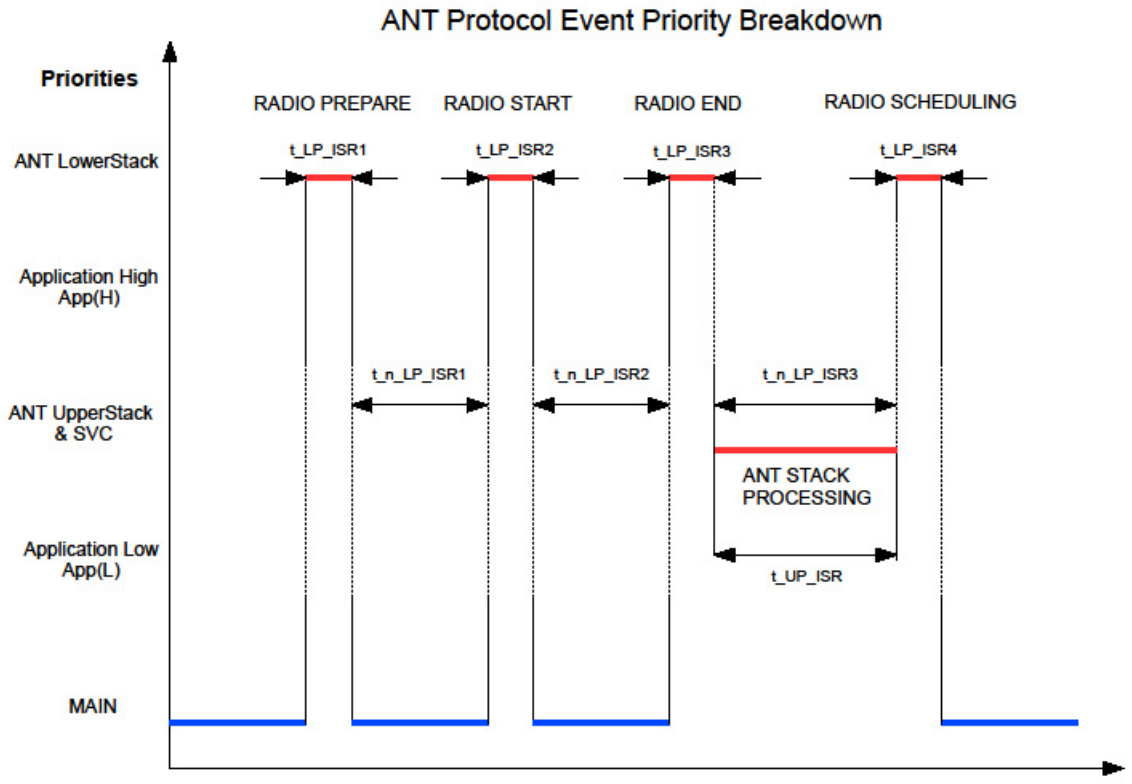

Table 13 *ANT protocol event*

10.3.1 LowerStack

Table 14 summarizes ANT radio and stack processing times for all supported ANT activity types in typical use cases with no high application priority interruption. High priority application interrupts will directly affect ANT STACK PROCESSING overhead time (t_UP_ISR).

Parameter	Description	Min.	Typ.	Max.	Comment
$t_{\_LP\_ISR1}$	Maximum ANT_RADIO_PREPARE interrupt processing time.	78 µs	[1]	163 µs	Range of maximums for all ANT activities.
$t_{\_n\_LP\_ISR1}$	Minimum time between RADIO_PREPARE to RADIO_START.	80 µs	[1]	173 µs	Range of minimums for all ANT activities.
$t_{\_LP\_ISR2}$	Maximum ANT_RADIO_START interrupt processing time.	30 µs	30 µs	30 µs	
$t_{\_n\_LP\_ISR2}$	Minimum time between RADIO_START and RADIO_END.	251 µs	[1]	251 µs	Range of minimums for all ANT activities.
$t_{\_LP\_ISR3}$	Maximum ANT_RADIO_END interrupt processing time.	48 µs	[1]	65 µs	Range of maximums for all ANT activities.
$t_{\_n\_LP\_ISR3}$	Minimum time between RADIO_END to RADIO_SCHEDULING.	185 µs	[1]	368 µs	Range of minimums for all ANT activities.
$t_{\_LP\_ISR4}$	Maximum ANT_RADIO_SCHEDULING interrupt processing time.	88 µs	88 µs	88 µs	

1. This data depends on ANT channel type and activity.

Table 14 *LowerStack interrupt latency numbers*

10.3.2 UpperStack

Parameter	Description	Min.	Typ.	Max.	Comment
$t_{\_UP\_ISR}$	Maximum ANT stack interrupt processing time	233 µs	1	310 µs	Range of maximums for all.
$t_{\_SVC\_ISR}$	Maximum ANT SVC call interrupt processing time		2		
$t_{\_n\_SVC\_ISR}$	Minimum time between ANT SVC call interrupts		3		

1. This data depends on ANT channel type and activity.
2. This data depends on an SVC call.
3. This data depends on the Application.

Table 15 *UpperStack interrupt latency numbers*

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

$$App(L)_{latency_{\max}} = t_{\_UP\_ISR} + t_{\_LP\_ISR3} + t_{\_LP\_ISR4}$$

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

$$App(H)_{latency_{\max}} = \max(t_{\_LP\_ISRx}) \; where \; x = 1,2,3,4$$

Restrictions:

- During ANT STACK PROCESSING, high application interrupts (App H) may be run to service critical operations (for example UART byte interrupts). The service operation should not exceed 100 µs every 500 µs interval or else ANT performance will be impacted.
- If high application interrupts are not needed, all application processes should be run in APP(L) or MAIN level.
- While ANT allows application processing to occur during ANT RF events (to maximize application CPU time), internal non-volatile memory/flash writes and erases are not allowed to be done freely. These operations must be scheduled outside of ANT events and use of the provided flash interface API by the SoftDevice is required, see ***Chapter 6 "Flash memory API"*** on page 11.

11 ANT power profiles

These profiles give a detailed overview of the stages of a radio event, the approximate timing of stages within an event, and how to calculate the peak current at each stage. Currently only the master transmit channel and slave receive channel profiles are shown. Similar calculations can be extended to other ANT activity modes.

11.1 Master Transmit Channel

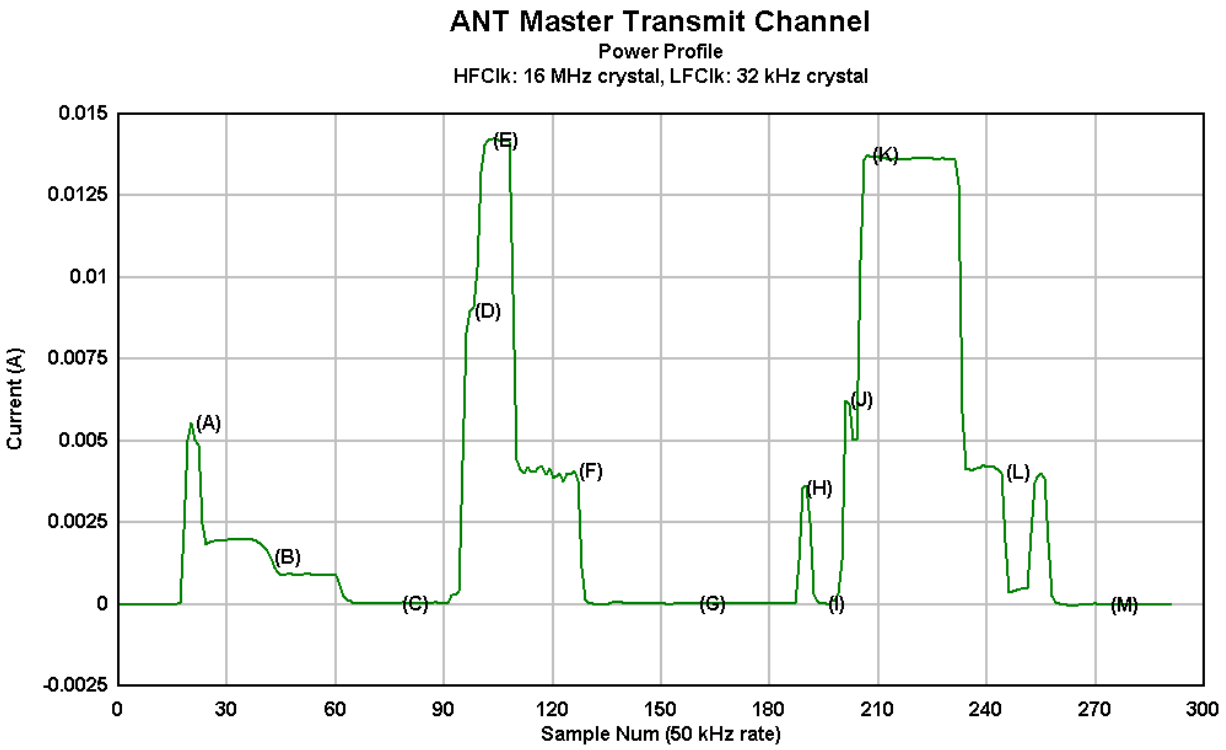

Figure 7 *ANT Master Transmit Channel*

Stage	Description	Current Calculations[1]
(A)	Preprocessing	$I_{ON} + I_{RTC} + I_{X32k} + I_{CPU,Flash}$[2]
(B)	Standby + XO ramp	$I_{ON} + I_{RTC} + I_{X32k} + I_{START,X16M}$
(C)	Standby	$I_{ON} + I_{RTC} + I_{X32k} + I_{STBYX,16M}$
(D)	Radio TX Start	$I_{ON} + I_{RTC} + I_{X32k} + I_{X16M} + \int(I_{START,TX})$
(E)	Radio TX	$I_{ON} + I_{RTC} + I_{X32k} + I_{TX,0dBm} + I_{CPU,Flash}$[3]
(F)	Post-processing	$I_{ON} + I_{RTC} + I_{X32k} + I_{CPU,Flash}$
(G)	Standby	$I_{ON} + I_{RTC} + I_{X32k} + I_{STBYX,16M}$
(H)	Pre-processing	$I_{ON} + I_{RTC} + I_{X32k} + I_{CPU,Flash}$
(I)	Standby	$I_{ON} + I_{RTC} + I_{X32k} + I_{STBYX,16M}$
(J)	Radio RX Start	$I_{ON} + I_{RTC} + I_{X32k} + I_{X16M} + \int(I_{START,RX})$
(K)	Radio RX	$I_{ON} + I_{RTC} + I_{X32k} + I_{X16M} + I_{RX}$[4]
(L)	Post-processing	$I_{ON} + I_{RTC} + I_{X32k} + I_{CPU,Flash}$
(M)	Idle	$I_{ON} + I_{RTC} + I_{X32k}$

1. See the corresponding product specification for the symbol values.
2. Startup transients not included.
3. Application given CPU time to run processing and CPU sleeping is not allowed.
4. Application given CPU time to run processing (account for ~$I_{CPU,Flash}$ if running).

Table 16 *ANT Master Transmit Channel*

Note: When using 32 k synthesized or 32 k RC clock, replace I_{X32k} with $I_{SYNT32k}$ or I_{RC32k}, respectively.

11.2 Slave Receive Channel

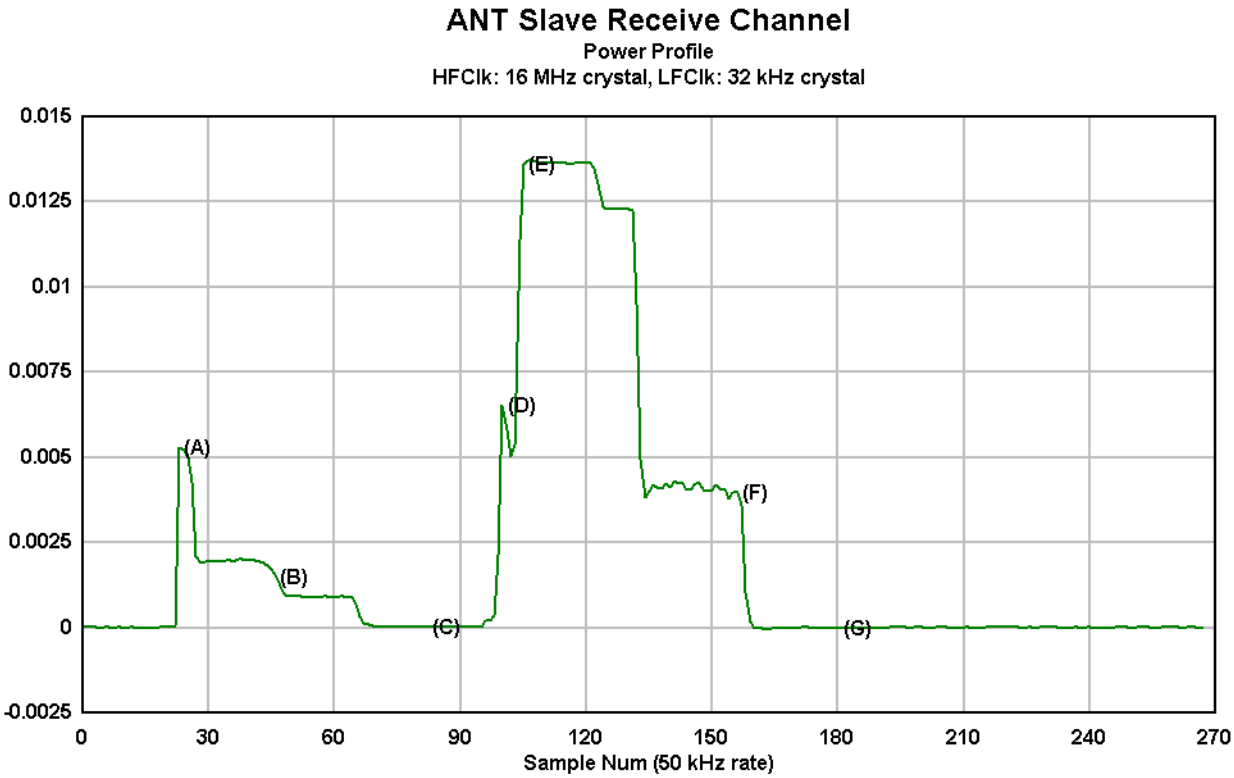

Figure 8 *ANT Slave Receive Channel*

Stage	Description	Current Calculations[1]
(A)	Preprocessing	$I_{ON} + I_{RTC} + I_{X32k} + I_{CPU,Flash}$[2]
(B)	Standby + XO ramp	$I_{ON} + I_{RTC} + I_{X32k} + I_{START,X16M}$
(C)	Standby	$I_{ON} + I_{RTC} + I_{X32k} + I_{STBYX,16M}$
(D)	Radio RX Start	$I_{ON} + I_{RTC} + I_{X32k} + I_{X16M} + \int(I_{START,RX})$
(E)	Radio RX	$I_{ON} + I_{RTC} + I_{X32k} + I_{X16M} + I_{RX}$[3]
(F)	Post-processing	$I_{ON} + I_{RTC} + I_{X32k} + I_{CPU,Flash}$
(G)	Idle	$I_{ON} + I_{RTC} + I_{X32k}$

1. See the corresponding product specification for the symbol values.
2. Startup current transients not included.
3. Application given CPU time to run processing (account for ~$I_{CPU,Flash}$ if running).

Table 17 *ANT Slave Receive Channel*

Note: When using 32 k synthesized or 32 k RC clock, replace I_{X32k} with $I_{SYNT32k}$ or I_{RC32k}, respectively.

11.3 ANT CPU Usage + Power Profile overlay

The following figures show the SoftDevice CPU and radio usage during ANT activities overlaid with the power profiles. The CPU profile timing is not drawn to scale but provided for a better overview

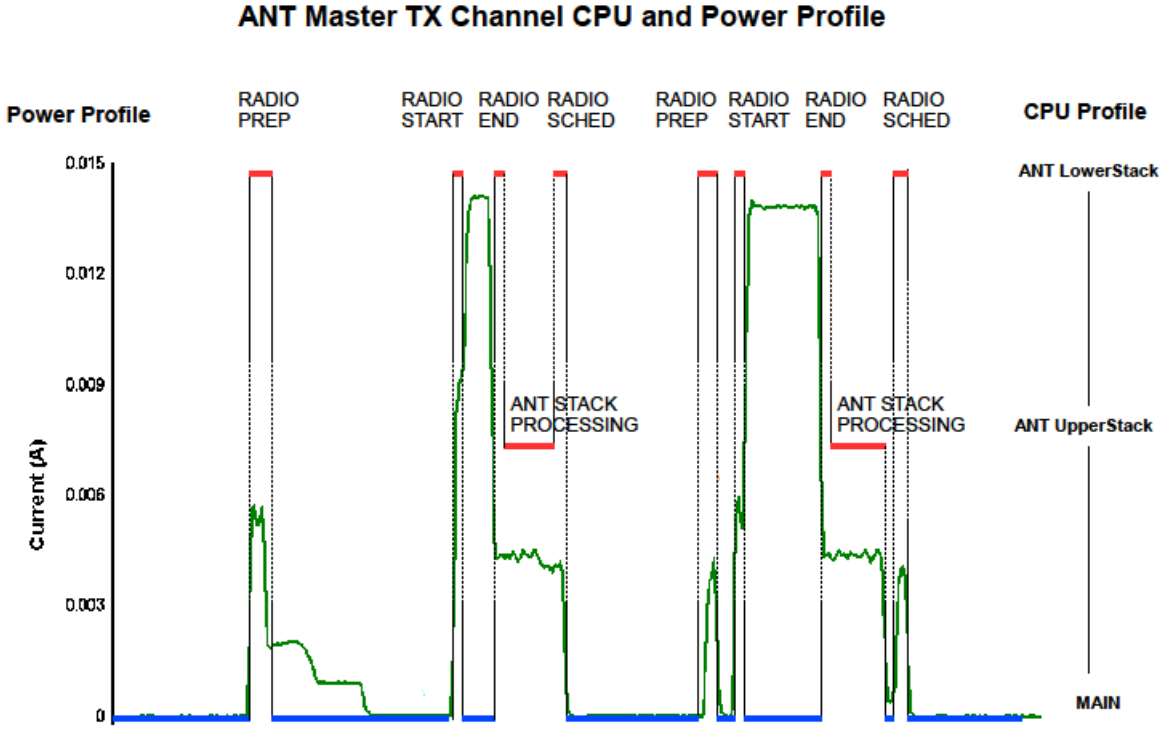

Figure 9 *ANT Master TX Channel CPU and Power Profile*

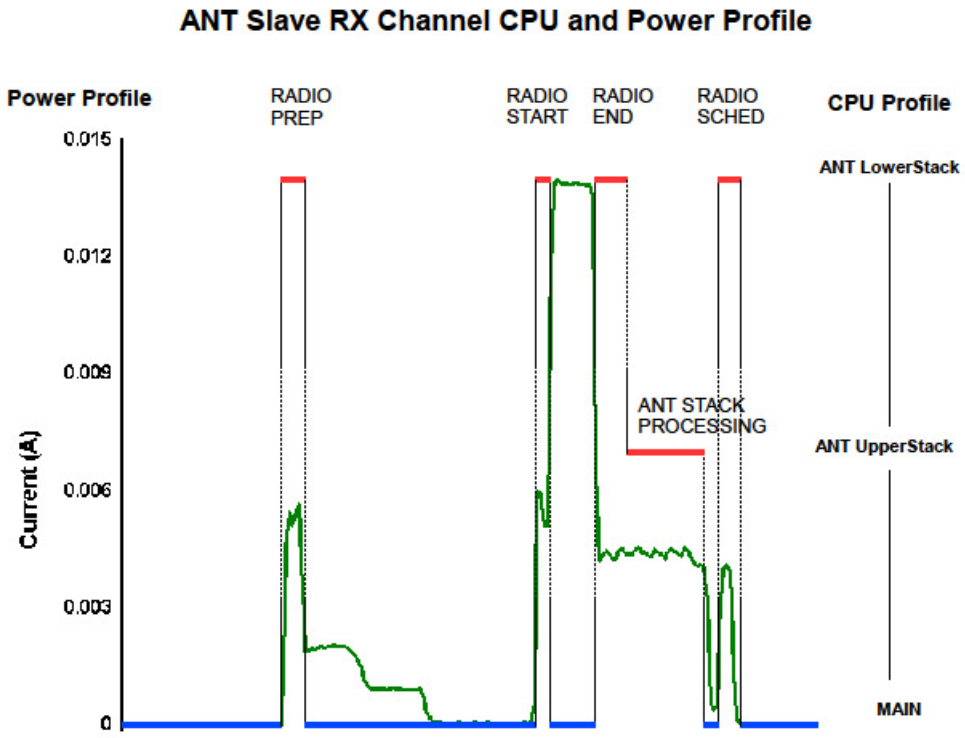

Figure 10 *ANT Slave RX Channel CPU and Power Profile*

12 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 18***.

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 19***.

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 18 *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 19 *SoftDevice revision examples*

The test qualification levels are outlined in ***Table 20***.

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 20 *Test qualification levels*

12.1 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.