nRF24LU1+

Single Chip 2.4 GHz Transceiver with USB Microcontroller and Flash Memory

Product Specification v1.1

Key Features

- nRF24L01+ compatible RF transceiver
- Worldwide 2.4 GHz ISM band operation
- Up to 2 Mbps on air data rate
- Enhanced ShockBurst™ hardware link layer
- Air compatible with nRF24LU1, nRF24LE1, nRF24L01+, nRF24L01, nRF2401A, nRF2402, nRF24E1 and nRF24E2
- Low cost external ±60 ppm 16 MHz crystal
- Full speed USB 2.0 compliant device controller
- Up to 12 Mbps USB transfer rate
- 2 control, 10 bulk/interrupt and 2 ISO endpoints
- Dedicated 512 bytes endpoint buffer RAM
- Software controlled pull-up resistor for D+
- PLL for full-speed USB operation
- Voltage regulator, 4.0 to 5.25V supply range
- Enhanced 8-bit 8051 compatible microcontroller
- Drop-in compatibility with nRF24LU1
- Reduced instruction cycle time
- 32-bit multiplication-division unit
- 16 or 32 kbytes of on-chip flash memory
- 2 kbytes of on-chip SRAM
- 6 general purpose digital input/output pins
- Hardware SPI slave and master, UART
- 3 16-bit timers/counters
- AES encryption/decryption co-processor
- Supports firmware upgrade over USB
- Supports FS2 hardware debugger
- Compact 32-pin 5x5mm QFN package

Applications

- Compact USB dongles for wireless peripherals
- USB dongles for mouse, keyboards and remotes
- USB dongle 3-in-1 desktop bundles
- USB dongle for advanced media center remote controls
- USB dongle for game controllers
- Toys

April 2010

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.

All application information is advisory and does not form part of the specification.

Limiting values

Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the specifications are not implied. Exposure to limiting values for extended periods may affect device reliability.

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.

Data sheet status	
Objective product specification	This product specification contains target specifications for product development.
Preliminary product specification	This product specification contains preliminary data; supplementary data may be published from Nordic Semiconductor ASA later.
Product specification	This product 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.

Contact details

For your nearest dealer, please see www.nordicsemi.com

Main office:

Otto Nielsens veg 12
7004 Trondheim
Phone: +47 72 89 89 00
Fax: +47 72 89 89 89
www.nordicsemi.com

Revision History

Date	Version	Description
April 2010	1.1	Updated section 1.3 on page 11, caption name for Table 46. on page 87. Updated Figure 2. on page 13, Figure 16. on page 46, Figure 18. on page 48, section 1.3 on page 11, section 2.2 on page 15, Table 24. on page 65, Table 53. on page 90,section 7.7.3 on page 80 and Attention box.

RoHS statement

nRF24LU1+ where explicitly stated in this product specification meets the requirements of Directive 2002/95/EC of the European Parliament and of the Council on the Restriction of Hazardous Substances (RoHS). Complete hazardous substance reports as well as material composition reports for all active Nordic products can be found on our web site www.nordicsemi.com.

Contents

1 Introduction

The nRF24LU1+ is a unique single chip solution for compact USB dongles. The internal nRF24L01+ 2.4 GHz RF transceiver supports a wide range of applications including PC peripherals, sports accessories and game peripherals.

With an air data rate of 2 Mbps combined with full speed USB, supporting up to 12 Mbps, the nRF24LU1+ meets the stringent performance requirements of applications such as wireless mouse, game controllers and media center remote controls with displays.

The nRF24LU1+ integrates:

- A nRF24L01+ 2.4 GHz RF transceiver
- A full speed USB 2.0 compliant device controller
- An 8-bit microcontroller
- 16 or 32 kbytes of flash memory

All this is packaged on a compact 5x5mm package, low cost external BOM.

With an internal voltage regulator that enables the chip to be powered directly from the USB bus, it does not require an external voltage regulator, saving cost and board space. With a fully integrated RF synthesizer and PLL for the USB no external loop filters, resonators or VCO varactor diodes are required. All that is needed is a low cost ±60ppm 16 MHz crystal, matching circuitry and the antenna.

The main benefits of nRF24LU1+ are:

- Very compact USB dongle
- Low cost external BOM
- No need for an external voltage regulator
- Single low cost ±60ppm 16 MHz crystal
- Flash memory for firmware upgrades

1.1 Prerequisites

In order to fully understand the product specification, a good knowledge of electronic and software engineering is necessary.

1.2 Writing conventions

This product specification follows a set of typographic rules that makes the document consistent and easy to read. The following writing conventions are used:

- Commands, bit state conditions, and register names are written in `Courier`.
- Pin names and pin signal conditions are written in **`Courier bold`**.
- Cross references are underlined and highlighted in blue.

1.3 Features

Features of the nRF24LU1+ include:

- Fast 8-bit MCU:
  - Intel MCS 51 compliant instruction set
  - Reduced instruction cycle time, up to 12x compared to legacy 8051
  - 32 bit multiplication – division unit
- Memory:
  - 16 or 32 kbytes of on-chip flash memory with security features
  - 2 kbytes of on-chip RAM memory
  - Pre-programmed USB bootloader in the on-chip flash memory.
- 6 programmable digital input/output pins configurable as:
  - GPIO
  - SPI master
  - SPI slave
  - External interrupts
  - Timer inputs
  - Full duplex serial port
  - Debug interface
- High performance 2.4 GHz RF-transceiver
  - True single chip GFSK transceiver
  - Enhanced ShockBurst™ link layer support in HW:
    - Packet assembly/disassembly
    - Address and CRC computation
    - Auto ACK and retransmit
  - On the air data rate 250 kbps, 1 Mbps or 2 Mbps
  - Digital interface (SPI) speed 0-8 Mbps
  - 125 RF channel option, with 79 (2.402 GHz-2.480 GHz) channels within 2.400 - 2.4835 GHz
  - Short switching time enable frequency hopping
  - Fully RF compatible with nRF24LXX
  - RF compatible with nRF2401A, nRF2402, nRF24E1, nRF24E2 in 250 kbps and 1 Mbps mode
- AES encryption/decryption HW-block with 128 bits key length
  - ECB – Electronic Code Book mode
  - CBC – Cipher Block Chaining
  - CFB – Cipher FeedBack mode
  - OFB – Output FeedBack mode
  - CTR – Counter mode
- Full speed USB 2.0 compliant device controller supporting:
  - Data transfer rates up to 12 Mbit/s
  - Control, Interrupt, Bulk and ISO data transfer
  - Endpoint 0 for control
  - 5 input and 5 output Bulk/Interrupt endpoints
  - 1 input and 1 output iso-synchronous endpoints
  - Total 512 bytes of USB buffer endpoint memory sharable between endpoints
  - On-chip USB transceiver PHY
  - On-chip pull-up resistor on D+ line with software controlled disconnect
- Power management function:
  - Low power design supporting fully static stop/ standby/ suspend modes
  - Programmable MCU clock frequency from 64 kHz to 16 MHz
  - On-chip voltage regulators supporting low power mode (supplied from USB power)
  - Watchdog and wakeup functionality running in low power mode

- On-chip oscillator and PLL to obtain full speed USB operation and to reduce the need for external components
- On-chip power on reset generator and brown-out detector
- On-chip support for FS2 and nRFprobe™ HW debugger, supported by Keil development tools.
- Complete firmware platform available:
  - Hardware abstraction layer (HAL) Functions
  - USB library Functions
  - Standard and HID specific USB Requests and Descriptors
  - nRF24LU1+ Library functions
  - AES HAL
  - Application examples
  - Device Firmware Upgrade

1.4 Block diagram

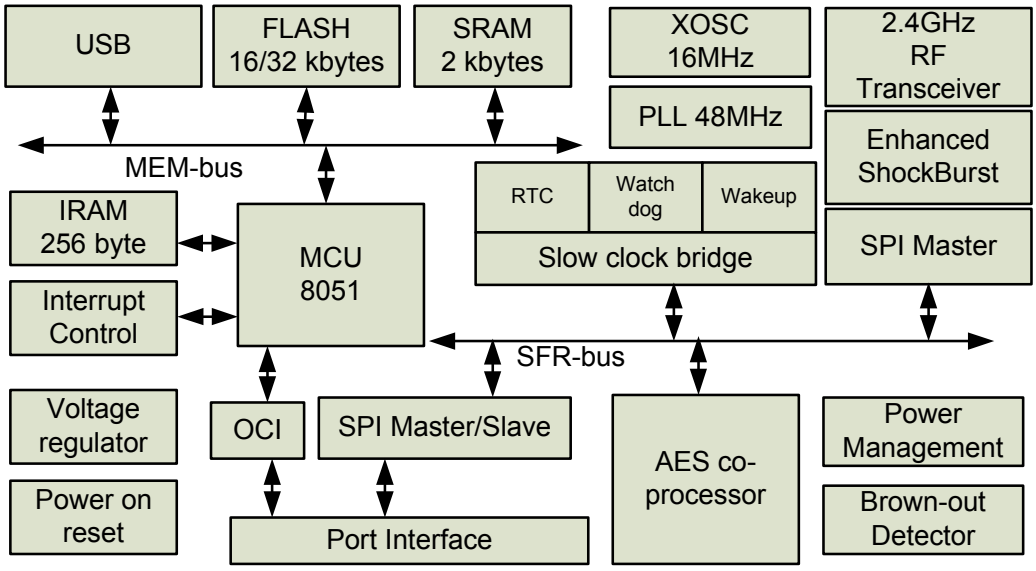


Figure 1. nRF24LU1+ block diagram

To find more information on the block diagram, see Table 1.

Name	Reference
USB	chapter 7 on page 63
FLASH	chapter 17 on page 135
SRAM	chapter 15 on page 127
2.4 GHz RF transceiver	chapter 6 on page 26
XOSC	section 24.2 on page 176
Enhanced ShockBurst™	section 6.4 on page 32
IRAM	chapter 16 on page 134
MCU	chapter 14 on page 119
RTC, Watchdog and Wakeup	chapter 19 on page 160
SPI Master	chapter 9 on page 99
Interrupt control	chapter 21 on page 170
SPI master/slave	chapter 9 on page 99 and chapter 10 on page 101
AES co-processor	chapter 8 on page 94
Power management	chapter 20 on page 165
Brown-out detector	section 17.4 on page 139

Table 1. Block diagram cross references

1.5 Typical system usage

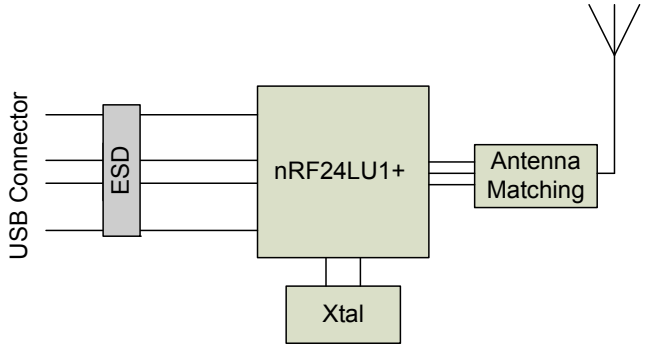


Figure 2. Typical system usage

2 Pin Information

2.1 Pin Assignments

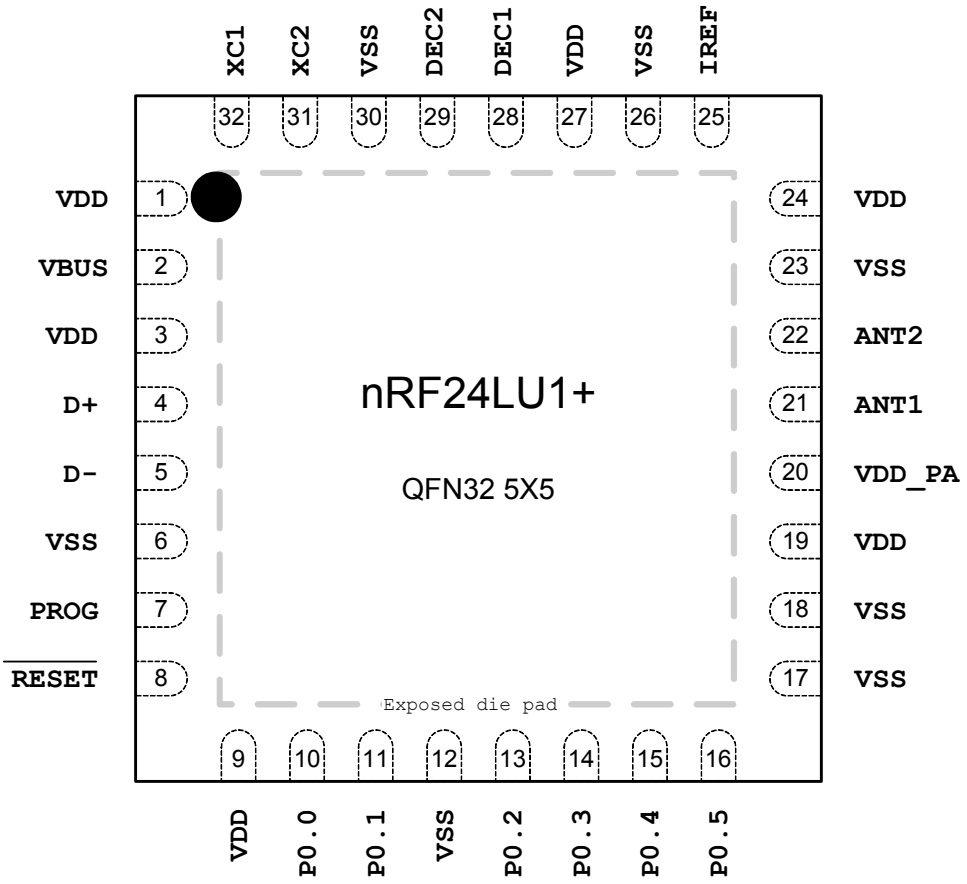


Figure 3. nRF24LU1+ pin assignment (top view) for a QFN32 5x5 mm package.

2.2 Pin Functions

Pin	Name	Type	Description
21, 22	ANT1, ANT2	RF	Antenna connection
5, 4	D-, D+	Digital I/O	USB data
28, 29	DEC1, DEC2	Power	Positive Digital Supply output for de-coupling purposes
25	IREF	Analog Input	Reference current output.
10, 11, 13, 14, 15, 16	P0.0 – P0.5	Digital I/O	General purpose data Port 0, bit 0 - 5. See Table 99. on page 118 for alternative pin functions.
7	PROG	Digital Input	Enables SPI flash programming
8	RESET	Digital Input	Reset for microcontroller, active low
2	VBUS	Power	USB power connection
1, 3, 9, 19, 24, 27	VDD	Power	Alternative power supply pins. The VDD pins must always be connected and de-coupled externally.
20	VDD_PA	Power Output	Power supply (+1.8V) to Power Amplifier
6, 12, 17, 18, 23, 26, 30	VSS	Power	Ground (0V)
32, 31	XC1, XC2	Analog Input	Crystal connection
	Exposed die pad	Power/heat relief	Not connected

Table 2. nRF24LU1+ pin functions

2.2.1 Antenna pins

ANT1 and ANT2 are connections for the external antenna (both receive and transmit).

2.2.2 USB pins

D- and D+ are the connections to the USB data lines. External ESD protection is recommended.

2.2.3 Power supply pins

VBUS and VSS are the power supply and ground pins. The nRF24LU1+ can operate from a single power supply.

The nRF24LU1+ contains an on-chip regulator that produces +3.3V on the VDD pins, from the VBUS supply line (4.0 – 5.25 V). Alternatively, the VBUS pin can be left open and the VDD pins may be fed from an external 3.3V supply. In this case, the on-chip 3.3V regulator is switched off.

Additional on-chip regulators produce voltages for internal analog and digital functions blocks. External decoupling capacitors are required on DEC1 and DEC2.

VDD_PA is a 1.8V output that is used to switch on an external RF Power Amplifier.

2.2.4 PROG pin

When set high this pin enables external SPI flash programming and Port 0 is configured as a slave SPI port.

The PROG pin needs an external pull-down resistor.

2.2.5 Reference current pins

The **IREF** pin must be connected to an external resistor.

2.2.6 Port pins

P0.0 – **P0.5** are six general purpose I/O pins. Their functions are described in chapter 13 on page 116.

2.2.7 External RESET Pin

A logic 0 on the RESET pin forces the nRF24LU1+ to a known start-up state.

2.2.8 Crystal oscillator pins

XC1 and **XC2** are connections to an external crystal.

3 Absolute Maximum Ratings

Maximum ratings are the extreme limits that you can expose the nRF24LU1+ to without permanently damaging it. Exposure to absolute maximum ratings for prolonged periods of time may affect device reliability.

Operating conditions	Minimum	Maximum	Units
Supply voltages			
VBUS	-0.3	+5.75	V
VSS		0	V
VDD	-0.3	+3.6	V
Input voltage			
VI	-0.3	+3.6	V
Temperatures			
Operating Temperature	-40	+85	°C
Storage Temperature	-40	+125	°C

Table 3. Absolute maximum ratings

Attention!
Observe precaution for handling
Electrostatic Sensitive Device.

HBM (Human Body Model): Class 1C

4 Operating Conditions

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
VBUS	Supply voltage		4.0	5	5.25	V
VDD	Alternative supply voltage		3.05	3.27	3.5	V
TEMP	Operating Temperature		-40	+27	+85	ºC

Table 4. Operating conditions

5 Electrical Specifications

This section contains electrical and timing specifications.

5.1 Power consumption and timing characteristics

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
I_{OP}	Average supply current in operating mode	a		24		mA
$I_{STANDBY}$	Supply current in standby mode	b		480		µA
	MCU					
$I_{MCU16MPLL}$	Running @ 16 MHz, generated from PLL			6.3		mA
$I_{MCU12MPLL}$	Running @ 12 MHz, generated from PLL			5		mA
$I_{MCU8MPLL}$	Running @ 8 MHz, generated from PLL			4		mA
$I_{MCU4MPLL}$	Running @ 4 MHz, generated from PLL			3		mA
$I_{MCU1.6MPLL}$	Running @ 1.6 MHz, generated from PLL			2		mA
$I_{MCU4MXO}$	Running @ 4 MHz, generated from XO			2.4		mA
$I_{MCU1.6MXO}$	Running @ 1.6 MHz, generated from XO			1.75		mA
$I_{MCU.32MXO}$	Running @ 0.32 MHz, generated from XO			1.1		mA
$I_{MCU64KXO}$	Running @ 0.064 MHz MHz, generated from XO			1		mA
Trst_act	From RESET to MCU active				2	ms
Tint_act	From INTERRUPT to MCU active				300	µs
Tact_stby	MCU from active to standby	c			32	µs
	RF Transceiver					
I_{TX}	RF Transceiver TX current @0dBm output power			11.1		mA
	RF Transceiver RX current @ 2 Mbps			13.3		mA
I_{RX}	RF Transceiver RX current @ 1 Mbps			12.9		mA
Tstby2a	RF Transceiver from standby to active	c			130	µs
Trst_radio	From RESET to RF Transceiver power down				50	ms
	USB					
I_{USB}	USB active current			4.4		mA
Tusb_wh	USB wakeup from host				500	µs
Tusb_wmcu	USB wakeup from MCU				300	µs
Tusbact_susp	USB from active to suspend	c			32	µs
	PLL					
Tplloff_on	PLL from off to on time	c d			250	µs
Tpllon_off	PLL from on to off time	c d			32	µs

a. MCU running radio receive at 2 Mbps and USB transmit
b. When MCU is in standby, USB is suspended and the RF Transceiver is in standby
c. Measured from start of the software instruction which executes the change of mode, see also Table 15.
d. Only possible when USB is in suspend mode

Table 5. Power consumption and timing characteristics

5.2 RF transceiver characteristics

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
	General RF conditions					
f_{OP}	Operating frequency	a	2400		2525	MHz
PLL_{res}	PLL Programming resolution			1		MHz
f_{XTAL}	Crystal frequency			16		MHz
Δf_{250}	Frequency deviation @ 250 kbps			±160		kHz
Δf_{1M}	Frequency deviation @ 1 Mbps			±160		kHz
Δf_{2M}	Frequency deviation @ 2 Mbps			±320		kHz
R_{GFSK}	Air data rate	b	250		2000	kbps
$F_{CHANNEL\ 1M}$	Non-overlapping channel spacing @ 250 kbps/1 Mbps)	c		1		MHz
$F_{CHANNEL\ 2M}$	Non-overlapping channel spacing @ 2 Mbps			2		MHz
	Transmitter operation					
P_{RF}	Maximum output power	d		0	+4	dBm
P_{RFC}	RF power control range		16	18	20	dB
P_{RFCR}	RF power accuracy				±4	dB
P_{BW2}	20dB bandwidth for modulated carrier (2 Mbps)			1800	2000	kHz
P_{BW1}	20dB bandwidth for modulated carrier (1 Mbps)			950	1100	kHz
P_{BW250}	20dB bandwidth for modulated carrier (250 kbps)			700	800	kHz
$P_{RF1.2}$	1^{st} Adjacent Channel Transmit Power 2 MHz (2 Mbps)				-20	dBc
$P_{RF2.2}$	2^{nd} Adjacent Channel Transmit Power 4 MHz (2 Mbps)				-45	dBc
$P_{RF1.1}$	1^{st} Adjacent Channel Transmit Power 1 MHz (1 Mbps)				-20	dBc
$P_{RF2.1}$	2^{nd} Adjacent Channel Transmit Power 2 MHz (1 Mbps)				-40	dBc
$P_{RF1.250}$	1^{st} Adjacent Channel Transmit Power 1 MHz (250 kbps)				-25	dBc
$P_{RF2.250}$	2^{nd} Adjacent Channel Transmit Power 2 MHz (250 kbps)				-40	dBc
	Receiver operation					
RX_{MAX}	Maximum received signal at < 0.1% BER			0		dBm
RX_{SENS}	Sensitivity (0.1% BER) @ 2 Mbps			-82		dBm
RX_{SENS}	Sensitivity (0.1% BER) @ 1 Mbps			-85		dBm
RX_{SENS}	Sensitivity (0.1% BER) @ 250 kbps	e		-94		dBm
RX selectivity according to ETSI EN 300 440-1 V1.3.1 (2001-09) page 27						
C/I_{CO}	C/I co-channel (2 Mbps)			7		dBc
C/I_{1ST}	1^{st} ACS (Adjacent Channel Selectivity), C/I 2 MHz (2 Mbps)			3		dBc
C/I_{2ND}	2^{nd} ACS, C/I 4 MHz (2 Mbps)			-17		dBc

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
C/I_{3RD}	3^{rd} ACS, C/I 6 MHz (2 Mbps)			-21		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 12 MHz (2 Mbps)	f		-40		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 36 MHz (2 Mbps)			-48		dBc
C/I_{CO}	C/I co-channel (1 Mbps)			9		dBc
C/I_{1ST}	1^{st} ACS, C/I 1 MHz (1 Mbps)			8		dBc
C/I_{2ND}	2^{nd} ACS, C/I 2 MHz (1 Mbps)			-20		dBc
C/I_{3RD}	3^{rd} ACS, C/I 3 MHz (1 Mbps)			-30		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 6 MHz (1 Mbps)			-40		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 25 MHz (1 Mbps)	f		-47		dBc
C/I_{CO}	C/I co-channel (250 kbps)			12		dBc
C/I_{1ST}	1^{st} ACS, C/I 1 MHz (250 kbps)			-12		dBc
C/I_{2ND}	2^{nd} ACS, C/I 2 MHz (250 kbps)			-33		dBc
C/I_{3RD}	3^{rd} ACS, C/I 3 MHz (250 kbps)			-38		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 6 MHz (250 kbps)			-50		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 25 MHz (250 kbps)	f		-60		dBc
RX selectivity with nRF24L01 equal modulation on interfering signal (Pin = -67dBm for wanted signal)						
C/I_{CO}	C/I co-channel (2 Mbps) (modulated carrier)			11		dBc
C/I_{1ST}	1^{st} ACS (Adjacent Channel Selectivity), C/I 2 MHz (2 Mbps)			4		dBc
C/I_{2ND}	2^{nd} ACS, C/I 4 MHz (2 Mbps)			-18		dBc
C/I_{3RD}	3^{rd} ACS, C/I 6 MHz (2 Mbps)			-24		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 12 MHz (2 Mbps)			-40		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 36 MHz (2 Mbps)			-48		dBc
C/I_{CO}	C/I co-channel (1 Mbps)			12		dBc
C/I_{1ST}	1^{st} ACS, C/I 1 MHz (1 Mbps)			8		dBc
C/I_{2ND}	2^{nd} ACS, C/I 2 MHz (1 Mbps)			-21		dBc
C/I_{3RD}	3^{rd} ACS, C/I 3 MHz (1 Mbps)			-30		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 6 MHz (1 Mbps)			-40		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 25 MHz (1 Mbps)			-50		dBc
C/I_{CO}	C/I co-channel (250 kbps)			7		dBc
C/I_{1ST}	1^{st} ACS, C/I 1 MHz (250 kbps)			-12		dBc
C/I_{2ND}	2^{nd} ACS, C/I 2 MHz (250 kbps)			-34		dBc
C/I_{3RD}	3^{rd} ACS, C/I 3 MHz (250 kbps)			-39		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 6 MHz (250 kbps)			-50		dBc
C/I_{Nth}	N^{th} ACS, C/I f_i > 25 MHz (250 kbps)			-60		dBc
RX intermodulation performance according to Bluetooth specification version 2.0, 4^{th} November 2004, page 42						
P_IM(6) @ 2 Mbps	Input power of IM interferers at 6 and 12 MHz distance from wanted signal	g		-42		dBm
P_IM(8) @ 2Mbps	Input power of IM interferers at 8 and 16 MHz distance from wanted signal	g		-38		dBm

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
P_IM(10) @ 2Mbps	Input power of IM interferers at 10 and 20 MHz distance from wanted signal	g		-37		dBm
P_IM(3) @ 1Mbps	Input power of IM interferers at 3 and 6 MHz distance from wanted signal	g		-36		dBm
P_IM(4) @ 1Mbps	Input power of IM interferers at 4 and 8 MHz distance from wanted signal	g		-36		dBm
P_IM(5) @ 1Mbps	Input power of IM interferers at 5 and 10 MHz distance from wanted signal	g		-36		dBm
P_IM(3) @ 250 kbps	Input power of IM interferers at 3 and 6 MHz distance from wanted signal	g		-36		dBm
P_IM(4) @ 250 kbps	Input power of IM interferers at 4 and 8 MHz distance from wanted signal	g		-36		dBm
P_IM(5) @ 250 kbps	Input power of IM interferers at 5 and 10 MHz distance from wanted signal	g		-36		dBm

a. Usable band is determined by local regulations.
b. Data rate in each burst on-air.
c. The minimum channel spacing is 1 MHz.
d. Antenna load impedance = 15Ω + j88Ω.
e. For 250 kpbs sensitivity, frequencies which are integer multiples of 16 MHz (2400, 2416 and so on,) sensitivity is reduced.
f. Narrow Band (In Band) Blocking measurements:
0 to ±40 MHz; 1 MHz step size
For Interferer frequency offsets n*2*fxtal, blocking performance is degraded by approximately 5dB compared to adjacent figures.
g. Wanted signal level at Pin = -64dBm. Two interferers with equal input power are used. The interferer closest in frequency is unmodulated, the other interferer is modulated equal with the wanted signal. The input power of interferers where the sensitivity equals BER = 0.1% is presented.

Table 6. RF Transceiver specifications

5.3 USB interface

The USB interface electrical performance is compliant with the USB specification 2.0.

Characteristic	Symbol	Conditions	Min.	Typ.	Max	Unit
Electrical characteristics						
Input high voltage (driven)	VIH		2.0			V
Input low voltage	VIL				0.8	V
Differential input sensitivity	VDI	|(D+) – (D-)|	0.2			V
Differential common mode range	VCM	Includes VDI range	0.8		2.5	V
Single ended receiver threshold	VSE		0.8		2.0	V
Single ended receiver hysteresis	VSEH			200		mV
Output low voltage	VOL		0		0.3	V
Output high voltage	VOH		2.8		3.6	V
Differential output signal cross-point voltage	VCRS		1.3		2.0	V
Internal pull-up resistor (Standby mode)	R_{PU1}		900	1100	1575	Ω
Internal pull-up resistor (Active mode)	R_{PU2}		1425	2100	3090	Ω
Termination voltage connected to R_{PU}	VTRM		3.05		3.5	V
Output driver resistance (does not include the series resistance)	ZDRV	Steady state drive		15		Ω
Timing characteristics						
Driver rise time	TFR	CL=50pF	4		20	ns
Driver fall time	TFF	CL=50pF	4		20	ns
Rise/fall time matching	TFRFF	TRF / TFF	90		111	%
Transceiver pad capacitance	CIN	Pad to ground			20	pF

Table 7. USB interface characteristics

5.4 Flash memory

Characteristic	Symbol	Conditions	Min.	Typ.	Max	Unit
Endurance	Nendur		1000			cycles
Data retention	Tret	25°C	100			years

Table 8. Flash memory characteristics

Name	Size	Unit
Flash memory MainBlock	32768	bytes
Flash InfoPage	512	bytes
Flash page size	512	bytes

Table 9. Flash memory and page size

5.5 Crystal specifications

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
f_{NOM}	Nominal frequency (parallel resonant)			16.000		MHz
f_{TOL}	Frequency tolerance	a b			±60	ppm
C_L	Load capacitance			9	16	pF
C_0	Shunt capacitance			3	7	pF
ESR	Equivalent series resistance			50	100	Ω
P_D	Drive level				100	µW

a. Includes initial accuracy, stability over temperature, aging and frequency pulling due to incorrect load capacitance.
b. Frequency regulations in certain regions set tighter requirements on frequency tolerance (for example Japan and South Korea max ±50ppm).

Table 10. Crystal specifications

5.6 DC Electrical Characteristics

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
	Operating conditions					
VBUS	Supply voltage		4.0	5.0	5.25	V
TEMP	Operating Temperature		-40	+27	+85	ºC
	On-chip voltage regulators					
VDD	Output voltage	a	3.05	3.27	3.5	V
IVDD	External load current				2	mA

a. Also valid for VDD input voltage.

Table 11. DC characteristics

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
VIH	HIGH level input voltage		0.7 VDD		VDD	V
VIL	LOW level input voltage		VSS		0.3 VDD	V

Table 12. Digital input pin

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
VOH	HIGH level output voltage (IOH= -1.0mA)	a	VDD-0.3		VDD	V
VOL	LOW level output voltage (IOL= 1.0mA))		VSS		0.3	V

a. When the nRF24LU1+ is supplied from VBUS, there is a limit (IVDD) on the current that can be drawn from VDD by external devices. Current sourced by high outputs are supplied to external devices for this purpose.

Table 13. Digital output pin

6 RF Transceiver

The nRF24LU1+ uses the same 2.4 GHz GFSK RF transceiver with embedded protocol engine (Enhanced ShockBurst™) that is found in the nRF24L01+ single chip RF Transceiver and in the nRF24LE1 on-chip solution. The RF Transceiver is designed for operation in the world wide ISM frequency band at 2.400 - 2.4835 GHz and is very well suited for ultra low power wireless applications.

The RF Transceiver module is configured and operated through the RF transceiver map. This register map is accessed by the MCU through a dedicated on-chip Serial Peripheral interface (SPI) and is available in all power modes of the RF Transceiver module.

The embedded protocol engine (Enhanced ShockBurst™) enables data packet communication and supports various modes from manual operation to advanced autonomous protocol operation. Data FIFOs in the RF Transceiver module ensure a smooth data flow between the RF Transceiver module and the nRF24LU1+ MCU.

The rest of this chapter is written in the context of the RF Transceiver module as the core and the rest of the nRF24LU1+ as external circuitry to this module.

6.1 Features

Features of the RF Transceiver include:

- General
  - Worldwide 2.4GHz ISM band operation
  - Common antenna interface in transmit and receive
  - GFSK modulation
  - 250 kbps, 1 and 2Mbps on air data rate
- Transmitter
  - Programmable output power: 0, -6, -12 or -18dBm
  - 11.1mA at 0dBm output power
- Receiver
  - Integrated channel filters
  - 13.3mA at 2 Mbps
  - -82dBm sensitivity at 2 Mbps
  - -85dBm sensitivity at 1 Mbps
  - -94dBm sensitivity at 250 kbps
- RF Synthesizer
  - Fully integrated synthesizer
  - 1 MHz frequency programming resolution
  - Accepts low cost ±60ppm 16 MHz crystal
  - 1 MHz non-overlapping channel spacing at 1 Mbps
  - 2 MHz non-overlapping channel spacing at 2 Mbps
- Enhanced ShockBurst™
  - 1 to 32 bytes dynamic payload length
  - Automatic packet handling (assembly/disassembly)
  - Automatic packet transaction handling (auto ACK, auto retransmit)
- 6 data pipe MultiCeiver™ for 6:1 star networks

6.2 Block diagram

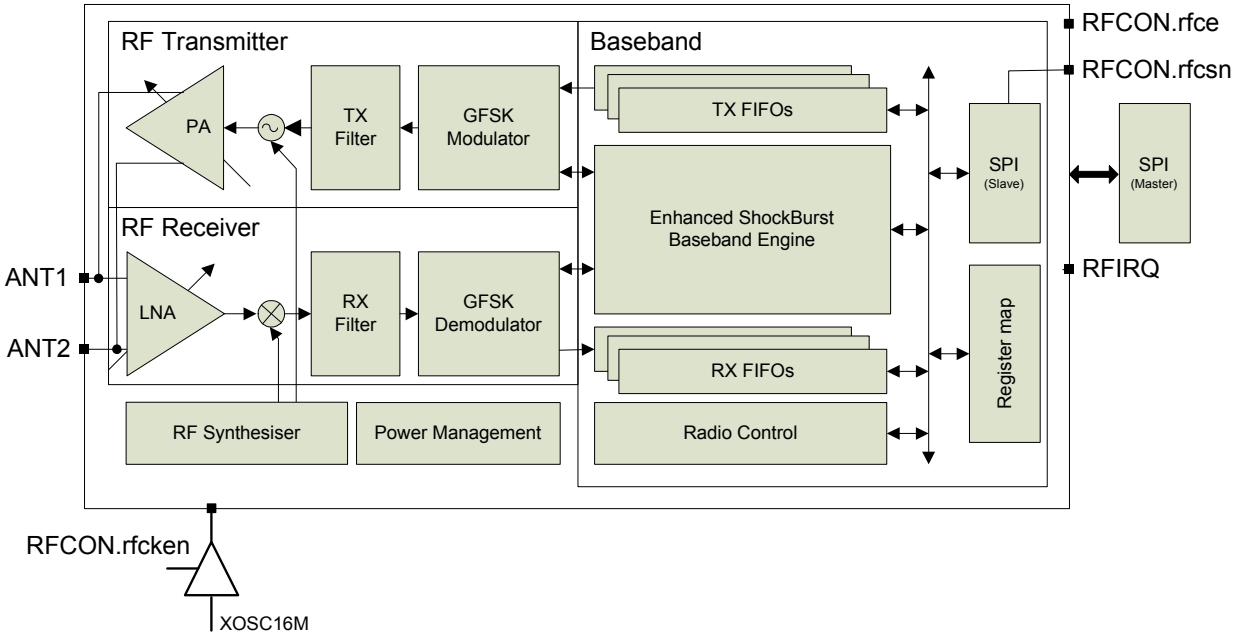


Figure 4. RF Transceiver block diagram

6.3 Functional description

This section describes the different operating modes of the RF Transceiver and the parameters used to control it.

The RF Transceiver module has a built-in state machine that controls the transitions between the different operating modes. The state machine is controlled by SFR register `RFCON` and RF transceiver register `CONFIG`, see section 6.5 for details.

6.3.1 Operational Modes

You can configure the RF Transceiver to power down, standby, RX and TX mode. This section describes these modes in detail.

6.3.1.1 State diagram

The state diagram (Figure 5.) shows the operating modes of the RF Transceiver and how they function. At the end of the reset sequence the RF Transceiver enters Power Down mode. When the RF Transceiver enters Power Down mode the MCU can still control the module through the SPI and the `rfcsn` bit in the `RFCON` register.

There are three types of distinct states highlighted in the state diagram:

- **Recommended operating mode:** is a recommended state used during normal operation.
- **Possible operating mode:** is a possible operating state, but is not used during normal operation.
- **Transition state:** is a time limited state used during start up of the oscillator and settling of the PLL.

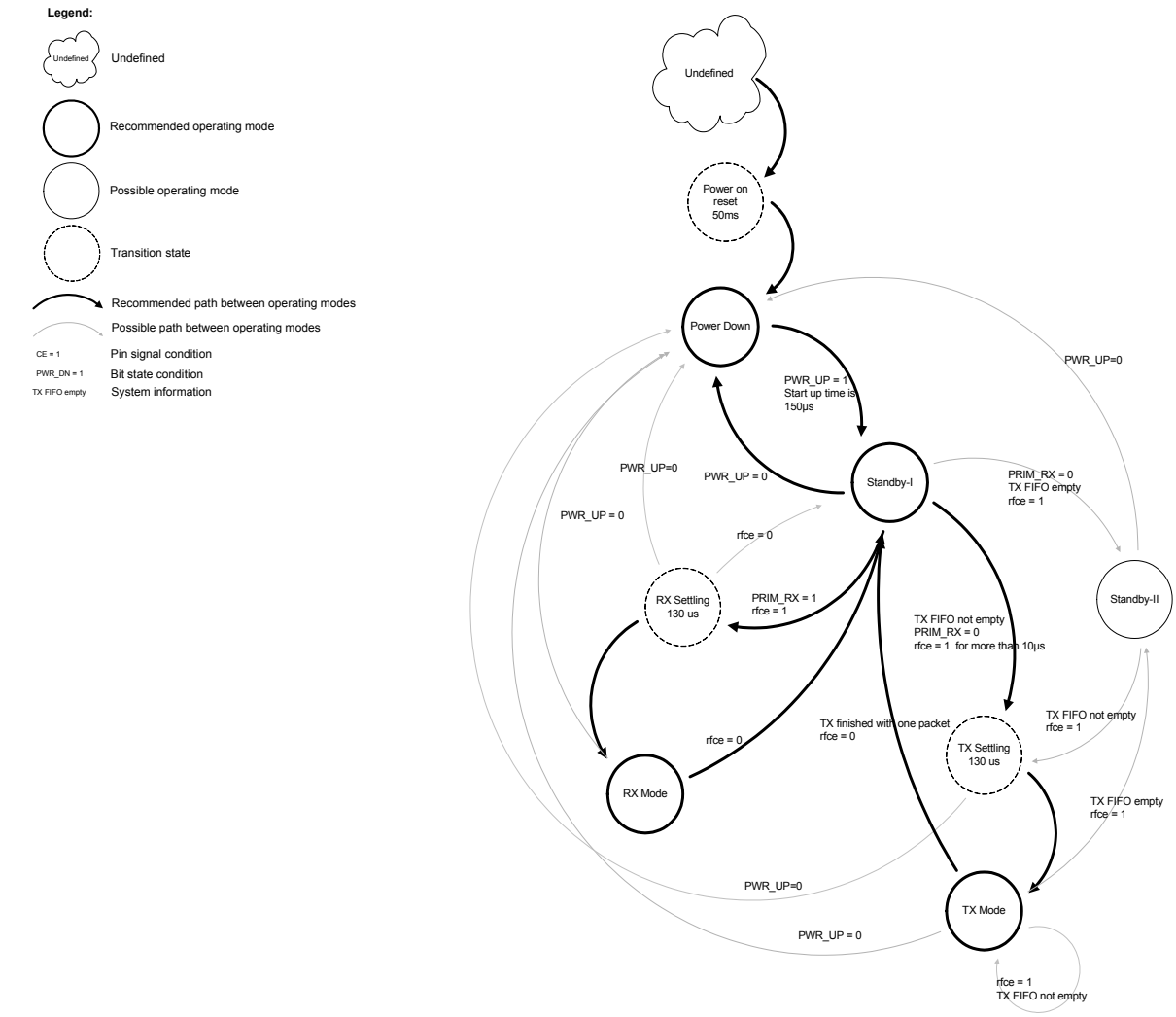


Figure 5. Radio control state diagram

6.3.1.2 Power down mode

In power down mode the RF Transceiver is disabled with minimal current consumption. All the register values available from the SPI are maintained and the SPI can be activated. For start up times see Table 15. Power down mode is entered by setting the `PWR_UP` bit in the `CONFIG` register low.

6.3.1.3 Standby modes

Standby-I mode

By setting the `PWR_UP` bit in the `CONFIG` register to 1, the RF Transceiver enters standby-I mode. Standby-I mode is used to minimize average current consumption while maintaining short start up times. Change to the active mode only happens if the `rfce` bit is enabled and when it is not enabled, the RF Transceiver returns to standby-I mode from both the TX and RX modes.

Standby-II mode

In standby-II mode extra clock buffers are active and more current is used compared to standby-I mode. The RF Transceiver enters standby-II mode if the `rfce` bit is held high on a PTX operation with an empty TX FIFO. If a new packet is downloaded to the TX FIFO, the PLL immediately starts and the packet is transmitted after the normal PLL settling delay (130µs).

The register values are maintained and the SPI can be activated during both standby modes. For start up times see Table 15.

6.3.1.4 RX mode

The RX mode is an active mode where the RF Transceiver is used as a receiver. To enter this mode, the RF Transceiver must have the `PWR_UP` bit, `PRIM_RX` bit and the `rfce` bit is set high.

In RX mode the receiver demodulates the signals from the RF channel, constantly presenting the demodulated data to the baseband protocol engine. The baseband protocol engine constantly searches for a valid packet. If a valid packet is found (by a matching address and a valid CRC) the payload of the packet is presented in a vacant slot in the RX FIFOs. If the RX FIFOs are full, the received packet is discarded.

The RF Transceiver remains in RX mode until the MCU configures it to standby-I mode or power down mode. However, if the automatic protocol features (Enhanced ShockBurst™) in the baseband protocol engine are enabled, the RF Transceiver can enter other modes in order to execute the protocol.

In RX mode a Received Power Detector (RPD) signal is available. The RPD is a signal that is set high when a RF signal higher than -64dBm is detected inside the receiving frequency channel. The internal `RPD` signal is filtered before presented to the `RPD` register. The RF signal must be present for at least 40µs before the `RPD` is set high. How to use the RPD is described in Section 6.3.4 on page 31.

6.3.1.5 TX mode

The TX mode is an active mode for transmitting packets. To enter this mode, the RF Transceiver must have the `PWR_UP` bit set high, `PRIM_RX` bit set low, a payload in the TX FIFO and a high pulse on the `rfce` bit for more than 10µs.

The RF Transceiver stays in TX mode until it finishes transmitting a packet. If `rfce` = 0, RF Transceiver returns to standby-I mode. If `rfce` = 1, the status of the TX FIFO determines the next action. If the TX FIFO is not empty the RF Transceiver remains in TX mode and transmits the next packet. If the TX FIFO is empty the RF Transceiver goes into standby-II mode. The RF Transceiver transmitter PLL operates in open loop when in TX mode. It is important never to keep the RF Transceiver in TX mode for more than 4ms at a time. If the Enhanced ShockBurst™ features are enabled, RF Transceiver is never in TX mode longer than 4ms.

6.3.1.6 Operational modes configuration

The following table (Table 14.) describes how to configure the operational modes.

Mode	PWR_UP register	PRIM_RX register	rfce	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFO. Will empty all levels in TX FIFO[a].
TX mode	1	0	Minimum 10µs high pulse	Data in TX FIFO.Will empty one level in TX FIFO[b].
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

a. If the rfce bit is held high the TX FIFO is emptied and all necessary ACK and possible retransmits are carried out. The transmission continues as long as the TX FIFO is refilled. If the TX FIFO is empty when the rfce bit is still high, the RF Transceiver enters standby-II mode. In this mode the transmission of a packet is started as soon as the rfcsn is set high after an upload (UL) of a packet to TX FIFO.

b. This operating mode pulses the rfce bit high for at least 10µs. This allows one packet to transmit. This is the normal operating mode. After the packet is transmitted, the RF Transceiver enters standby-I mode.

Table 14. RF Transceiver main modes

6.3.1.7 Timing information

The timing information in this section relates to the transitions between modes and the timing for the rfce bit. The transition from TX mode to RX mode or vice versa is the same as the transition from the standby modes to TX mode or RX mode (130µs), as described in Table 15.

Name	RF Transceiver	Max.	Min.	Comments
Tpd2stby	Power Down ➔ Standby mode	150µs		
Tstby2a	Standby modes ➔ TX/RX mode	130µs		
Thce	Minimum rfce high		10µs	
Tpece2csn	Delay from rfce pos. edge to rfcsn low		4µs	

Table 15. Operational timing of RF Transceiver

Note: If VDD is turned off, the register values are lost and you must reconfigure the RF Transceiver before entering the TX or RX modes.

6.3.2 Air data rate

The air data rate is the modulated signaling rate the RF Transceiver uses when transmitting and receiving data. It can be 250 kbps, 1 Mbps or 2 Mbps. Using lower air data rate gives better receiver sensitivity than higher air data rate. But, high air data rate gives lower average current consumption and reduced probability of on-air collisions.

The air data rate is set by the `RF_DR` bit in the `RF_SETUP` register. A transmitter and a receiver must be programmed with the same air data rate to communicate with each other.

The RF Transceiver is fully compatible with nRF24L01. For compatibility with nRF2401A, nRF2402, nRF24E1, and nRF24E2 the air data rate must be set to 250 kbps or 1 Mbps.

6.3.3 RF channel frequency

The RF channel frequency determines the center of the channel used by the RF Transceiver. The channel occupies a bandwidth of less than 1 MHz at 250 kbps and 1 Mbps and a bandwidth of less than 2 MHz at 2 Mbps. The RF Transceiver can operate on frequencies from 2.400 GHz to 2.525 GHz. The programming resolution of the RF channel frequency setting is 1 MHz.

At 2 Mbps the channel occupies a bandwidth wider than the resolution of the RF channel frequency setting. To ensure non-overlapping channels in 2 Mbps mode, the channel spacing must be 2 MHz or more. At 1 Mbps and 250 kbps the channel bandwidth is the same or lower than the resolution of the RF frequency.

The RF channel frequency is set by the `RF_CH` register according to the following formula:

$F_0 = 2400 + RF\_CH$ *MHz*

You must program a transmitter and a receiver with the same RF channel frequency to communicate with each other.

6.3.4 Received Power Detector measurements

Received Power Detector (RPD), located in register 09, bit 0, triggers at received power levels above -64dBm that are present in the RF channel you receive on. If the received power is less than -64dBm, RDP = 0.

The RPD can be read out at any time while the RF Transceiver is in receive mode. This offers a snapshot of the current received power level in the channel. The RPD is latched whenever a packet is received or when the MCU sets rfce low

The status of RPD is correct when RX mode is enabled and after a wait time of Tstby2a +Tdelay_AGC= 130 µs + 40 µs. The RX gain varies over temperature which means that the RPD threshold also varies over temperature. The RPD threshold value is reduced by - 5dB at T = -40°C and increased by + 5dB at 85°C.

6.3.5 PA control

The PA (Power Amplifier) control is used to set the output power from the RF Transceiver power amplifier. In TX mode PA control has four programmable steps, see Table 16.

The PA control is set by the RF_PWR bits in the RF_SETUP register.

SPI RF-SETUP (RF_PWR)	RF output power	DC current consumption
11	0dBm	11.1mA
10	-6dBm	8.8mA
01	-12dBm	7.3
00	-18dBm	6.8mA

Conditions: VDD = 3.0V, VSS = 0V, T_A = 27ºC, Load impedance = 15Ω+j88Ω.

Table 16. RF output power setting for the RF Transceiver

6.3.6 RX/TX control

The RX/TX control is set by PRIM_RX bit in the CONFIG register and sets the RF Transceiver in transmit/ receive.

6.4 Enhanced ShockBurst™

Enhanced ShockBurst™ is a packet based data link layer that features automatic packet assembly and timing, automatic acknowledgement and retransmissions of packets. Enhanced ShockBurst™ enables the implementation of ultra low power and high performance communication. The Enhanced ShockBurst™ features enable significant improvements of power efficiency for bi-directional and uni-directional systems, without adding complexity on the host controller side.

6.4.1 Features

The main features of Enhanced ShockBurst™ are:

- 1 to 32 bytes dynamic payload length
- Automatic packet handling
- Auto packet transaction handling
  - Auto Acknowledgement
  - Auto retransmit
- 6 data pipe MultiCeiver™ for 1:6 star networks

6.4.2 Enhanced ShockBurst™ overview

Enhanced ShockBurst™ uses ShockBurst™ for automatic packet handling and timing. During transmit, ShockBurst™ assembles the packet and clocks the bits in the data packet for transmission. During receive, ShockBurst™ constantly searches for a valid address in the demodulated signal. When ShockBurst™ finds a valid address, it processes the rest of the packet and validates it by CRC. If the packet is valid the payload is moved into a vacant slot in the RX FIFOs. All high speed bit handling and timing is controlled by ShockBurst™.

Enhanced ShockBurst™ features automatic packet transaction handling for the easy implementation of a reliable bi-directional data link. An Enhanced ShockBurst™ packet transaction is a packet exchange between two transceivers, with one transceiver acting as the Primary Receiver (PRX) and the other transceiver acting as the Primary Transmitter (PTX). An Enhanced ShockBurst™ packet transaction is always initiated by a packet transmission from the PTX, the transaction is complete when the PTX has received an

acknowledgment packet (ACK packet) from the PRX. The PRX can attach user data to the ACK packet enabling a bi-directional data link.

The automatic packet transaction handling works as follows:

1. You begin the transaction by transmitting a data packet from the PTX to the PRX. Enhanced ShockBurst™ automatically sets the PTX in receive mode to wait for the ACK packet.
2. If the packet is received by the PRX, Enhanced ShockBurst™ automatically assembles and transmits an acknowledgment packet (ACK packet) to the PTX before returning to receive mode.
3. If the PTX does not receive the ACK packet immediately, Enhanced ShockBurst™ automatically retransmits the original data packet after a programmable delay and sets the PTX in receive mode to wait for the ACK packet.

In Enhanced ShockBurst™ it is possible to configure parameters such as the maximum number of retransmits and the delay from one transmission to the next retransmission. All automatic handling is done without the involvement of the MCU.

6.4.3 Enhanced Shockburst™ packet format

The format of the Enhanced ShockBurst™ packet is described in this section. The Enhanced ShockBurst™ packet contains a preamble field, address field, packet control field, payload field and a CRC field. Figure 6. shows the packet format with MSB to the left.

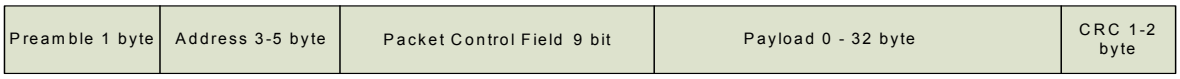


Figure 6. An Enhanced ShockBurst™ packet with payload (0-32 bytes)

6.4.3.1 Preamble

The preamble is a bit sequence used to synchronize the receivers demodulator to the incoming bit stream. The preamble is one byte long and is either 01010101 or 10101010. If the first bit in the address is 1 the preamble is automatically set to 10101010 and if the first bit is 0 the preamble is automatically set to 01010101. This is done to ensure there are enough transitions in the preamble to stabilize the receiver.

6.4.3.2 Address

This is the address for the receiver. An address ensures that the correct packet is detected by the receiver. The address field can be configured to be 3, 4 or, 5 bytes long with the `AW` register.

Note: Addresses where the level shifts only one time (that is, 000FFFFFFF) can often be detected in noise and can give a false detection, which may give a raised Packet-Error-Rate. Addresses as a continuation of the preamble (hi-low toggling) raises the Packet-Error-Rate.

6.4.3.3 Packet Control Field

Figure 7. shows the format of the 9-bit packet control field, MSB to the left.

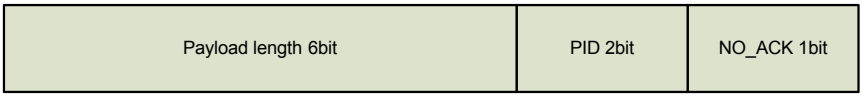


Figure 7. Packet control field

The packet control field contains a 6-bit payload length field, a 2-bit PID (Packet Identity) field and a 1-bit `NO_ACK` flag.

Payload length

This 6-bit field specifies the length of the payload in bytes. The length of the payload can be from 0 to 32 bytes.

Coding: 000000 = 0 byte (only used in empty ACK packets.) 100000 = 32 byte, 100001 = Don't care.

This field is only used if the Dynamic Payload Length function is enabled.

PID (Packet identification)

The 2-bit PID field is used to detect if the received packet is new or retransmitted. PID prevents the PRX operation from presenting the same payload more than once to the MCU. The PID field is incremented at the TX side for each new packet received through the SPI. The PID and CRC fields (see section 6.4.3.5 on page 35) are used by the PRX operation to determine if a packet is retransmitted or new. When several data packets are lost on the link, the PID fields may become equal to the last received PID. If a packet has the same PID as the previous packet, the RF Transceiver compares the CRC sums from both packets. If the CRC sums are also equal, the last received packet is considered a copy of the previously received packet and discarded.

No Acknowledgment flag (NO_ACK)

The Selective Auto Acknowledgement feature controls the `NO_ACK` flag.

This flag is only used when the auto acknowledgement feature is used. Setting the flag high, tells the receiver that the packet is not to be auto acknowledged.

6.4.3.4 Payload

The payload is the user defined content of the packet. It can be 0 to 32 bytes wide and is transmitted on-air when it is uploaded (unmodified) to the device.

Enhanced ShockBurst™ provides two alternatives for handling payload lengths; static and dynamic.

The default is static payload length. With static payload length all packets between a transmitter and a receiver have the same length. Static payload length is set by the `RX_PW_Px` registers on the receiver side. The payload length on the transmitter side is set by the number of bytes clocked into the TX_FIFO and must equal the value in the `RX_PW_Px` register on the receiver side.

Dynamic Payload Length (DPL) is an alternative to static payload length. DPL enables the transmitter to send packets with variable payload length to the receiver. This means that for a system with different payload lengths it is not necessary to scale the packet length to the longest payload.

With the DPL feature the nRF24L01+ can decode the payload length of the received packet automatically instead of using the `RX_PW_Px` registers. The MCU can read the length of the received payload by using the `R_RX_PL_WID` command.

Note: Always check if the packet width reported is 32 bytes or shorter when using the `R_RX_PL_WID` command. If its width is longer than 32 bytes then the packet contains errors and must be discarded. Discard the packet by using the `Flush_RX` command.

In order to enable DPL the `EN_DPL` bit in the `FEATURE` register must be enabled. In RX mode the `DYNPD` register must be set. A PTX that transmits to a PRX with DPL enabled must have the `DPL_P0` bit in `DYNPD` set.

6.4.3.5 CRC (Cyclic Redundancy Check)

The CRC is the error detection mechanism in the packet. It may either be 1 or 2 bytes and is calculated over the address, Packet Control Field and Payload.

The polynomial for 1 byte CRC is $X^8 + X^2 + X + 1$. Initial value 0xFF.

The polynomial for 2 byte CRC is $X^{16} + X^{12} + X^5 + 1$. Initial value 0xFFFF.

No packet is accepted by Enhanced ShockBurst™ if the CRC fails.

6.4.4 Automatic packet assembly

The automatic packet assembly assembles the preamble, address, packet control field, payload and CRC to make a complete packet before it is transmitted.

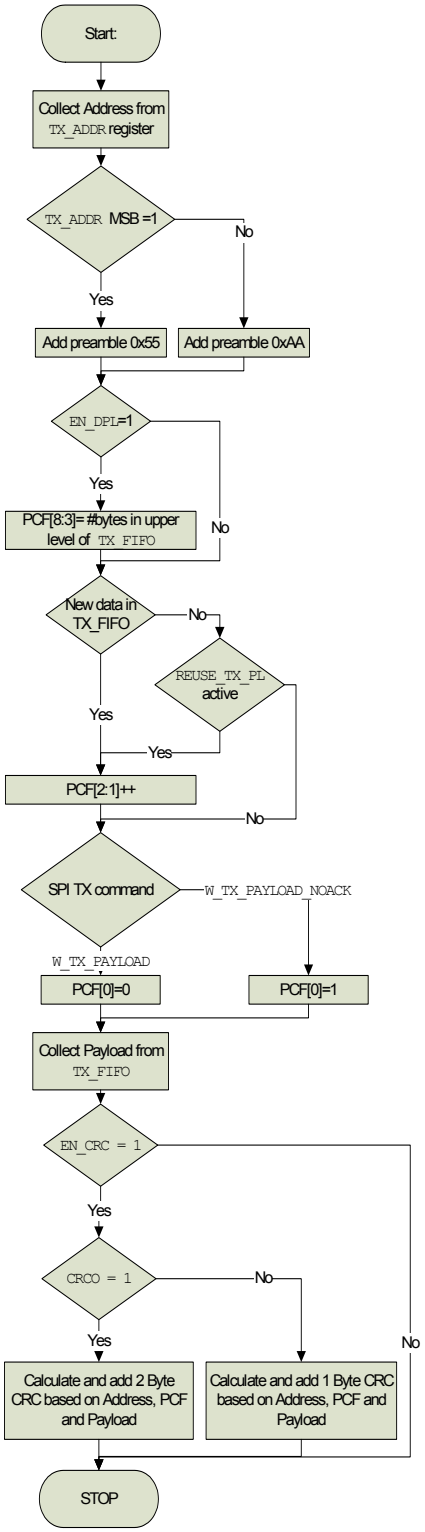


Figure 8. Automatic packet assembly

6.4.5 Automatic packet disassembly

After the packet is validated, Enhanced ShockBurst™ disassembles the packet and loads the payload into the RX FIFO, and asserts the RX_DR IRQ.

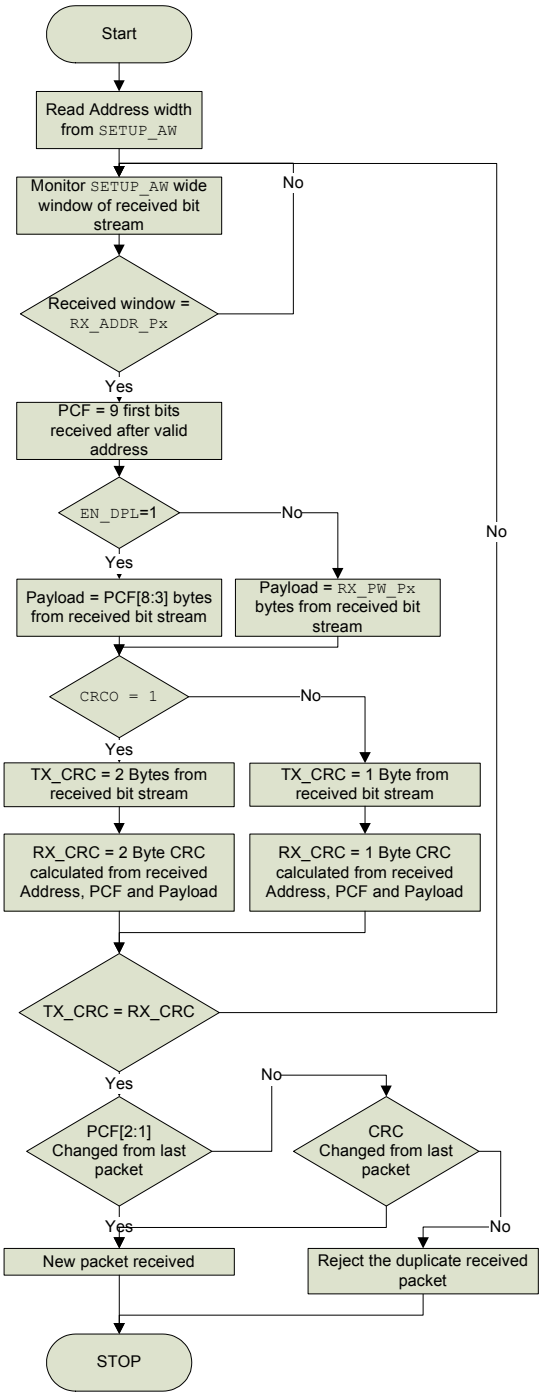


Figure 9. Automatic packet disassembly

6.4.6 Automatic packet transaction handling

Enhanced ShockBurst™ features two functions for automatic packet transaction handling; auto acknowledgement and auto re-transmit.

6.4.6.1 Auto Acknowledgement

Auto acknowledgment is a function that automatically transmits an ACK packet to the PTX after it has received and validated a packet. The auto acknowledgement function reduces the load of the system MCU and reduces average current consumption. The Auto Acknowledgement feature is enabled by setting the `EN_AA` register.

Note: If the received packet has the `NO_ACK` flag set, auto acknowledgement is not executed.

An ACK packet can contain an optional payload from PRX to PTX. In order to use this feature, the Dynamic Payload Length (DPL) feature must be enabled. The MCU on the PRX side has to upload the payload by clocking it into the TX FIFO by using the `W_ACK_PAYLOAD` command. The payload is pending in the TX FIFO (PRX) until a new packet is received from the PTX. The RF Transceiver can have three ACK packet payloads pending in the TX FIFO (PRX) at the same time.

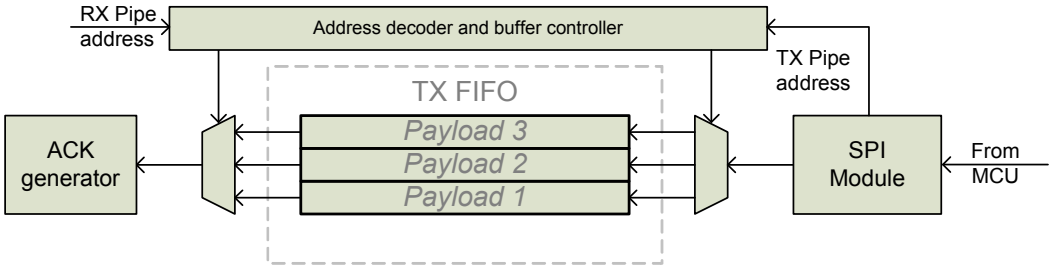


Figure 10. TX FIFO (PRX) with pending payloads

Figure 10. shows how the TX FIFO (PRX) is operated when handling pending ACK packet payloads. From the MCU the payload is clocked in with the `W_ACK_PAYLOAD` command. The address decoder and buffer controller ensure that the payload is stored in a vacant slot in the TX FIFO (PRX). When a packet is received, the address decoder and buffer controller are notified with the PTX address. This ensures that the right payload is presented to the ACK generator.

If the TX FIFO (PRX) contains more than one payload to a PTX, payloads are handled using the first in – first out principle. The TX FIFO (PRX) is blocked if all pending payloads are addressed to a PTX where the link is lost. In this case, the MCU can flush the TX FIFO (PRX) by using the `FLUSH_TX` command.

In order to enable Auto Acknowledgement with payload the `EN_ACK_PAY` bit in the `FEATURE` register must be set.

6.4.6.2 Auto Retransmission (ART)

The auto retransmission is a function that retransmits a packet if an ACK packet is not received. It is used in an auto acknowledgement system on the PTX. When a packet is not acknowledged, you can set the number of times it is allowed to retransmit by setting the ARC bits in the `SETUP_RETR` register. PTX enters RX mode and waits a time period for an ACK packet each time a packet is transmitted. The amount of time the PTX is in RX mode is based on the following conditions:

- Auto Retransmit Delay (ARD) elapsed.
- No address match within 250µs.
- After received packet (CRC correct or not) if address match within 250µs.

The RF Transceiver asserts the TX_DS IRQ when the ACK packet is received.

The RF Transceiver enters standby-I mode if there is no more untransmitted data in the TX FIFO and the rfce bit in the RFCON register is low. If the ACK packet is not received, the RF Transceiver goes back to TX mode after a delay defined by ARD and retransmits the data. This continues until acknowledgment is received, or the maximum number of retransmits is reached.

Two packet loss counters are incremented each time a packet is lost, ARC_CNT and PLOS_CNT in the OBSERVE_TX register. The ARC_CNT counts the number of retransmissions for the current transaction. You reset ARC_CNT by initiating a new transaction. The PLOS_CNT counts the total number of retransmissions since the last channel change. You reset PLOS_CNT by writing to the RF_CH register. It is possible to use the information in the OBSERVE_TX register to make an overall assessment of the channel quality.

The ARD defines the time from the end of a transmitted packet to when a retransmit starts on the PTX. ARD is set in SETUP_RETR register in steps of 250µs. A retransmit is made if no ACK packet is received by the PTX.

There is a restriction on the length of ARD when using ACK packets with payload. The ARD time must never be shorter than the sum of the startup time and the time on-air for the ACK packet.

- For 2 Mbps data rate and 5 byte address; 15 byte is maximum ACK packet payload length for ARD=250 µs (reset value).
- For 1 Mbps data rate and 5 byte address; 5 byte is maximum ACK packet payload length for ARD=250 µs (reset value).

ARD=500µs is long enough for any ACK payload length in 1 or 2 Mbps mode.

- For 250 kbps data rate and 5byte address the following values apply:

ARD	ACK packet size (in bytes)
1500 µs	All ACK payload sizes
1250 µs	≤ 24
1000 µs	≤ 16
750 µs	≤ 8
500 µs	Empty ACK with no payload

Table 17. Maximum ACK payload length for different retransmit delays at 250 kbps

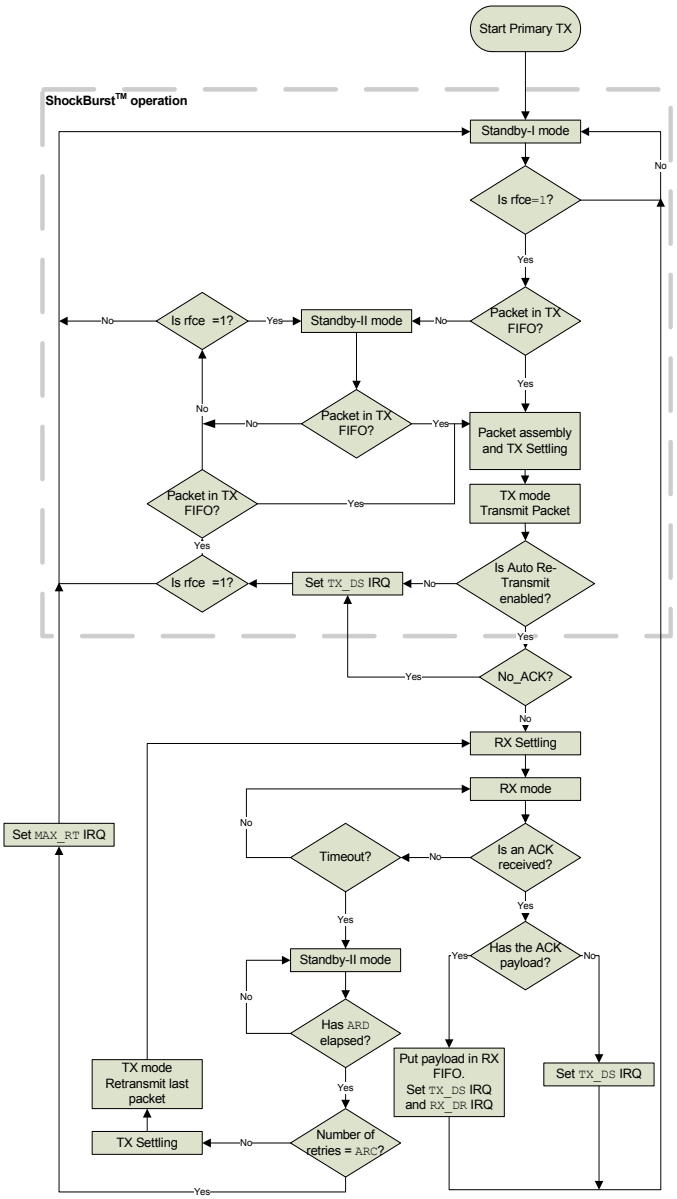
As an alternative to Auto Retransmit it is possible to manually set the RF Transceiver to retransmit a packet a number of times. This is done by the REUSE_TX_PL command. The MCU must initiate each transmission of the packet with a pulse on the **CE** pin when this command is used.

6.4.7 Enhanced ShockBurst™ flowcharts

This section contains flowcharts outlining PTX and PRX operation in Enhanced ShockBurst™.

6.4.7.1 PTX operation

The flowchart in Figure 11. outlines how a RF Transceiver configured as a PTX behaves after entering standby-I mode.



Note: ShockBurst™ operation is outlined with a dashed square.

Figure 11. PTX operations in Enhanced ShockBurst™

Activate PTX mode by setting the rfce bit in the RFCON register high. If there is a packet present in the TX FIFO the RF Transceiver enters TX mode and transmits the packet. If Auto Retransmit is enabled, the state machine checks if the NO_ACK flag is set. If it is not set, the RF Transceiver enters RX mode to receive an ACK packet. If the received ACK packet is empty, only the TX_DS IRQ is asserted. If the ACK packet contains a payload, both TX_DS IRQ and RX_DR IRQ are asserted simultaneously before the RF Transceiver returns to standby-I mode.

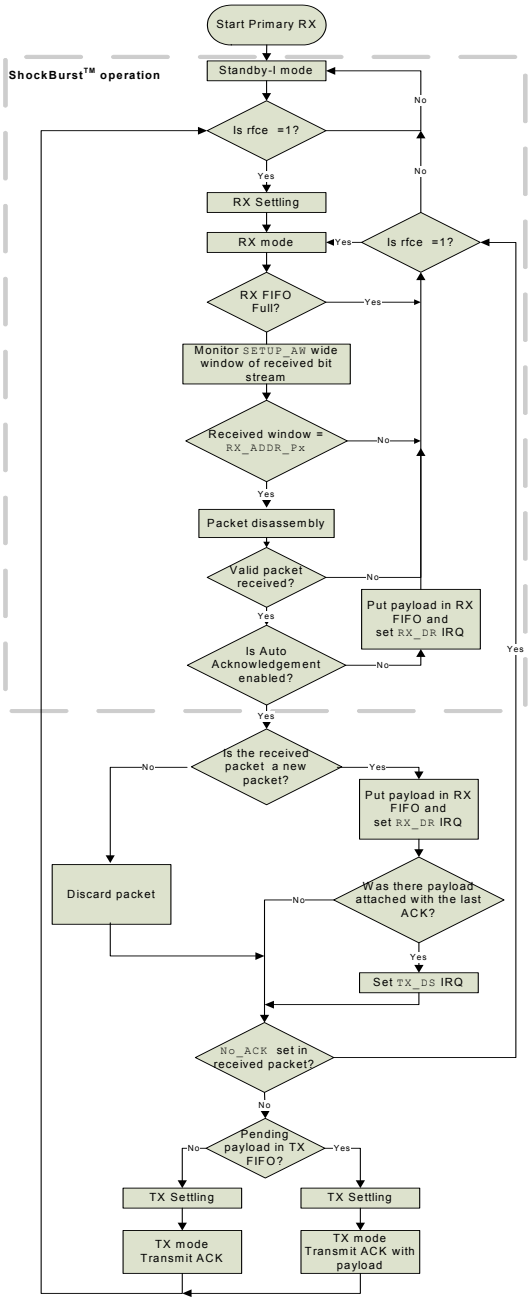
If the ACK packet is not received before timeout occurs, the RF Transceiver returns to standby-II mode. It stays in standby-II mode until the ARD has elapsed. If the number of retransmits has not reached the ARC, the RF Transceiver enters TX mode and transmits the last packet once more.

While executing the Auto Retransmit feature, the number of retransmits can reach the maximum number defined in ARC. If this happens, the RF Transceiver asserts the MAX_RT IRQ and returns to standby-I mode.

If the rfce bit in the RFCON register is high and the TX FIFO is empty, the RF Transceiver enters Standby-II mode.

6.4.7.2 PRX operation

The flowchart in Figure 12. outlines how a RF Transceiver configured as a PRX behaves after entering standby-I mode.



Note: ShockBurst™ operation is outlined with a dashed square.

Figure 12. PRX operations in Enhanced ShockBurst™

Activate PRX mode by setting the `rfce` bit in the `RFCON` register high. The RF Transceiver enters RX mode and starts searching for packets. If a packet is received and Auto Acknowledgement is enabled, the RF Transceiver decides if the packet is new or a copy of a previously received packet. If the packet is new

the payload is made available in the RX FIFO and the RX_DR IRQ is asserted. If the last received packet from the transmitter is acknowledged with an ACK packet with payload, the TX_DS IRQ indicates that the PTX received the ACK packet with payload. If the No_ACK flag is not set in the received packet, the PRX enters TX mode. If there is a pending payload in the TX FIFO it is attached to the ACK packet. After the ACK packet is transmitted, the RF Transceiver returns to RX mode.

A copy of a previously received packet might be received if the ACK packet is lost. In this case, the PRX discards the received packet and transmits an ACK packet before it returns to RX mode.

6.4.8 MultiCeiver™

MultiCeiver™ is a feature used in RX mode that contains a set of six parallel data pipes with unique addresses. A data pipe is a logical channel in the physical RF channel. Each data pipe has its own physical address (data pipe address) decoding in the RF Transceiver.

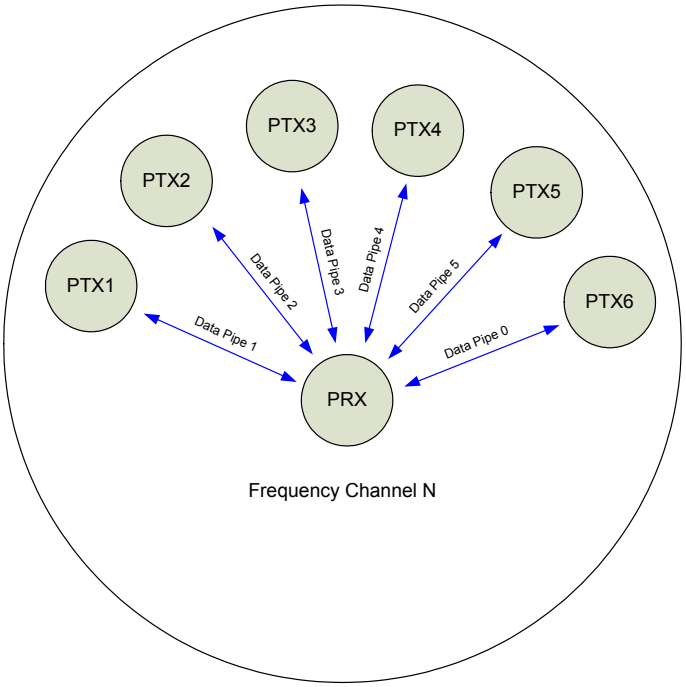


Figure 13. PRX using MultiCeiver™

The RF Transceiver configured as PRX (primary receiver) can receive data addressed to six different data pipes in one frequency channel as shown in Figure 13. Each data pipe has its own unique address and can be configured for individual behavior.

Up to six RF Transceivers configured as PTX can communicate with one RF Transceiver configured as PRX. All data pipe addresses are searched for simultaneously. Only one data pipe can receive a packet at a time. All data pipes can perform Enhanced ShockBurst™ functionality.

The following settings are common to all data pipes:

- CRC enabled/disabled (CRC always enabled when Enhanced ShockBurst™ is enabled)
- CRC encoding scheme
- RX address width
- Frequency channel
- Air data rate
- LNA gain

The data pipes are enabled with the bits in the `EN_RXADDR` register. By default only data pipe 0 and 1 are enabled. Each data pipe address is configured in the `RX_ADDR_PX` registers.

Note: Always ensure that none of the data pipes have the same address.

Each pipe can have up to a 5 byte configurable address. Data pipe 0 has a unique 5 byte address. Data pipes 1-5 share the four most significant address bytes. The LSByte must be unique for all six pipes. Figure 14. is an example of how data pipes 0-5 are addressed.

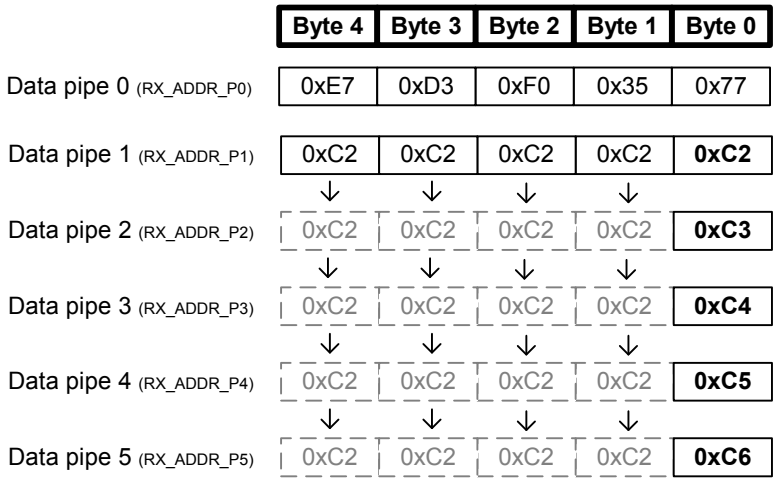


Figure 14. Addressing data pipes 0-5

The PRX, using MultiCeiver™ and Enhanced ShockBurst™, receives packets from more than one PTX. To ensure that the ACK packet from the PRX is transmitted to the correct PTX, the PRX takes the data pipe address where it received the packet and uses it as the TX address when transmitting the ACK packet. Figure 15. is an example of an address configuration for the PRX and PTX. On the PRX the RX_ADDR_Pn, defined as the pipe address, must be unique. On the PTX the TX_ADDR must be the same as the RX_ADDR_P0 and as the pipe address for the designated pipe.

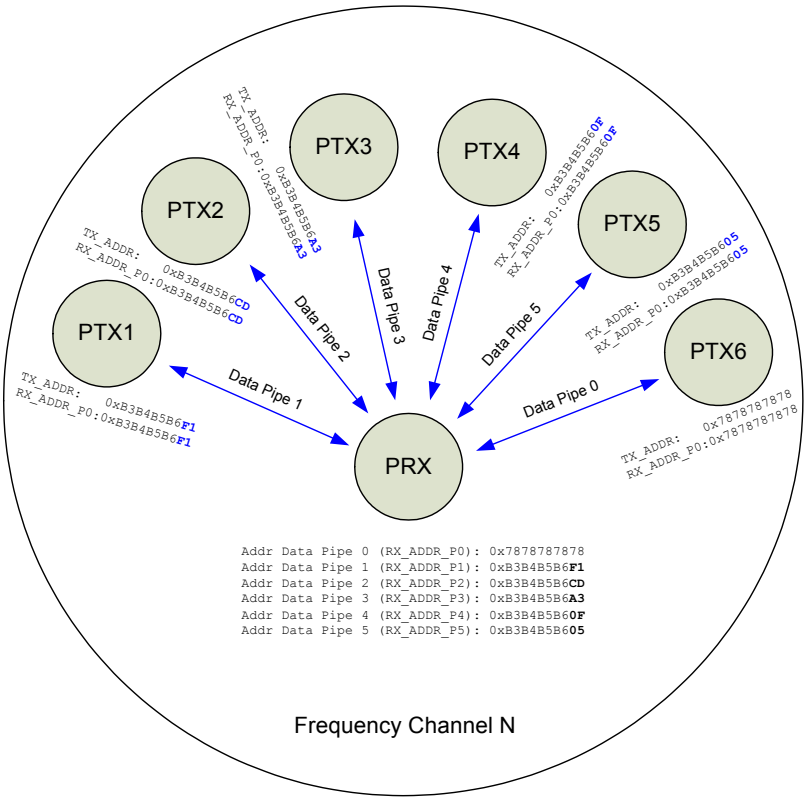


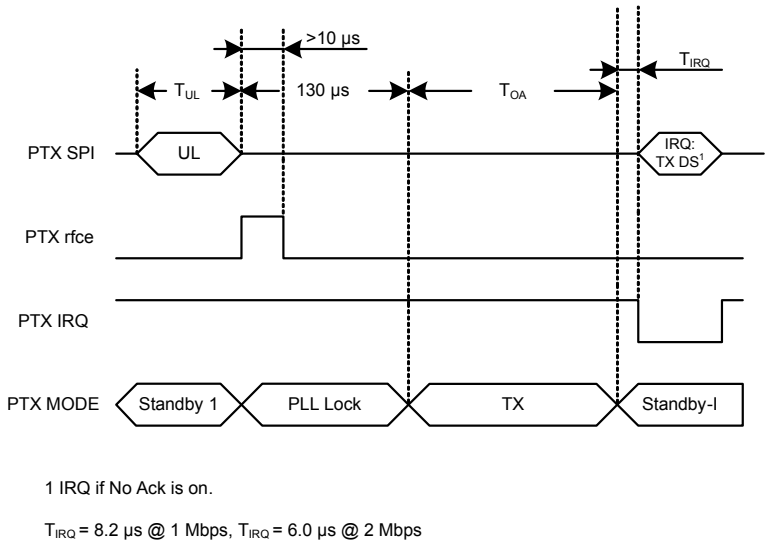
Figure 15. Example of data pipe addressing in MultiCeiver™

Only when a data pipe receives a complete packet can other data pipes begin to receive data. When multiple PTXs are transmitting to a PRX, the ARD can be used to skew the auto retransmission so that they only block each other once.

6.4.9 Enhanced ShockBurst™ timing

This section describes the timing sequence of Enhanced ShockBurst™ and how all modes are initiated and operated. The Enhanced ShockBurst™ timing is controlled through the Data and Control interface. The RF Transceiver can be set to static modes or autonomous modes where the internal state machine

controls the events. Each autonomous mode/sequence ends with a RFIRQ interrupt. All the interrupts are indicated as IRQ events in the timing diagrams.



1 IRQ if No Ack is on.

T_{IRQ} = 8.2 µs @ 1 Mbps, T_{IRQ} = 6.0 µs @ 2 Mbps

Figure 16. Transmitting one packet with NO_ACK on

The following equations calculate various timing measurements:

Symbol	Description	Equation
T_{OA}	Time on-air	$T_{OA} = \frac{packet\ length}{air\ data\ rate} = \frac{8\left[\frac{bit}{byte}\right]\cdot\left(1[byte]_{preamble}+3,4\ or\ 5[bytes]_{address}+N[bytes]_{payload}+1\ or\ 2[bytes]_{CRC}\right)+9[bit]_{packet\ control\ field}}{air\ data\ rate\left[\frac{bit}{s}\right]}$
T_{ACK}	Time on-air Ack	$T_{ACK} = \frac{packet\ length}{air\ data\ rate} = \frac{8\left[\frac{bit}{byte}\right]\cdot\left(1[byte]_{preamble}+3,4\ or\ 5[bytes]_{address}+N[bytes]_{payload}+1\ or\ 2[bytes]_{CRC}\right)+9[bit]_{packet\ control\ field}}{air\ data\ rate\left[\frac{bit}{s}\right]}$
T_{UL}	Time Upload	$T_{UL} = \frac{payload\ length}{SPI\ data\ rate} = \frac{8\left[\frac{bit}{byte}\right]\cdot N[bytes]_{payload}}{SPI\ data\ rate\left[\frac{bit}{s}\right]}$
T_{ESB}	Time Enhanced Shock-Burst™ cycle	$T_{ESB} = T_{UL} + 2\ .\ T_{stby2a} + T_{OA} + T_{ACK} + T_{IRQ}$

Table 18. Timing equations

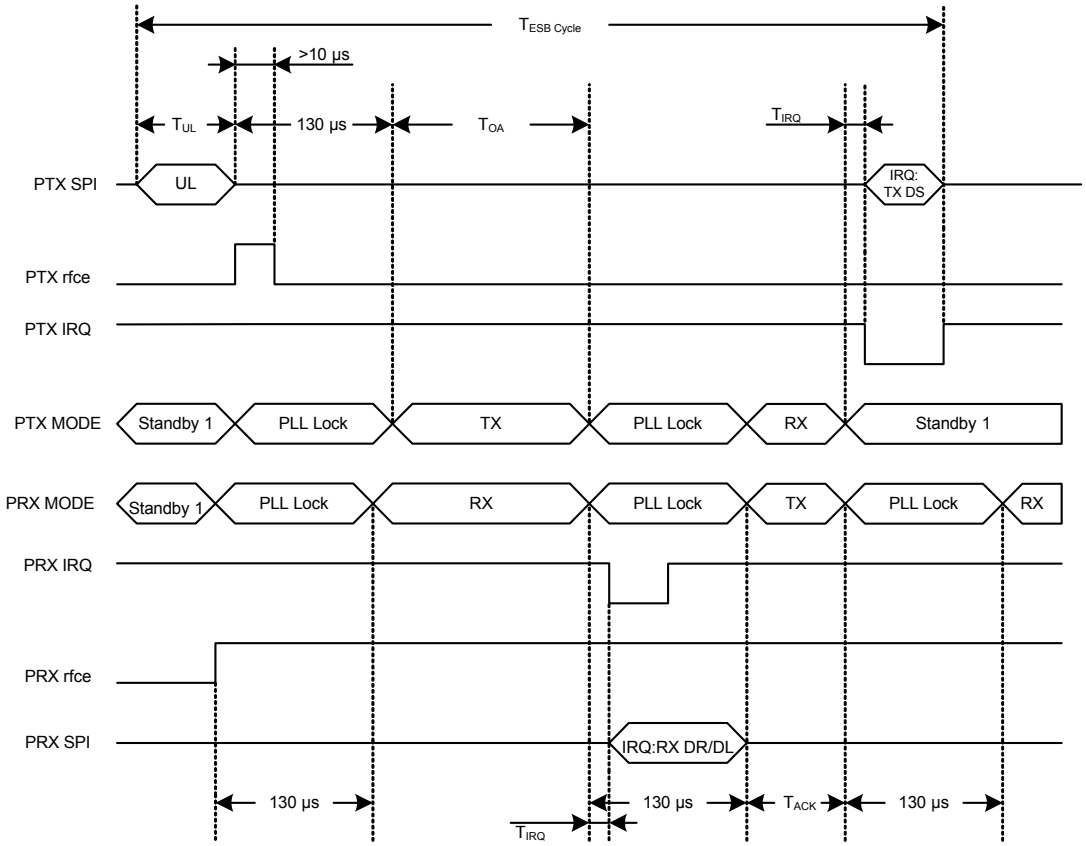


Figure 17. Timing of Enhanced ShockBurst™ for one packet upload (2 Mbps)

In Figure 17. the transmission and acknowledgement of a packet is shown. The PRX operation activates RX mode (`rfce`=1), and the PTX operation is activated in TX mode (`rfce`=1 for minimum 10 µs). After 130 µs the transmission starts and finishes after the elapse of T_{OA}.

When the transmission ends the PTX operation automatically switches to RX mode to wait for the ACK packet from the PRX operation. When the PRX operation receives the packet it sets the interrupt for the host MCU and switches to TX mode to send an ACK. After the PTX operation receives the ACK packet it sets the interrupt to the MCU and clears the packet from the TX FIFO.

In Figure 18. the PTX timing of a packet transmission is shown when the first ACK packet is lost. To see the complete transmission when the ACK packet fails see Figure 21.

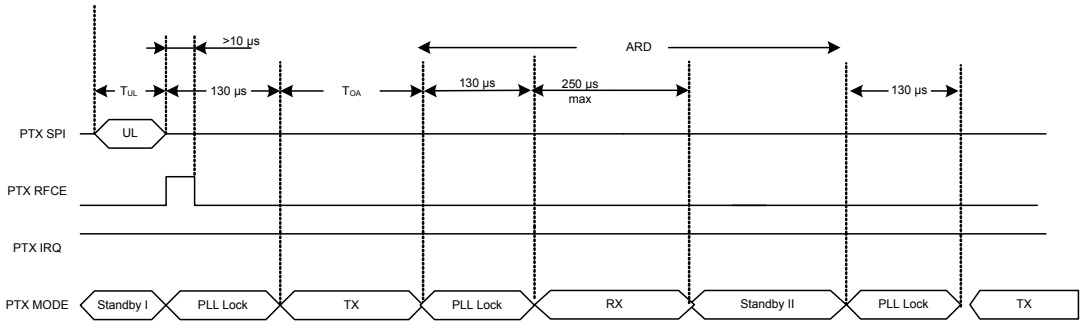


Figure 18. Timing of Enhanced ShockBurst™ when the first ACK packet is lost (2 Mbps)

6.4.10 Enhanced ShockBurst™ transaction diagram

This section describes several scenarios for the Enhanced ShockBurst™ automatic transaction handling. The call outs in this section's figures indicate the IRQs and other events. For MCU activity the event may be placed at a different timeframe.

Note: The figures in this section indicate the earliest possible download (DL) of the packet to the MCU and the latest possible upload (UL) of payload to the transmitter.

6.4.10.1 Single transaction with ACK packet and interrupts

In Figure 19. the basic auto acknowledgement is shown. After the packet is transmitted by the PTX and received by the PRX the ACK packet is transmitted from the PRX to the PTX. The `RX_DR` IRQ is asserted after the packet is received by the PRX, whereas the `TX_DS` IRQ is asserted when the packet is acknowledged and the ACK packet is received by the PTX.

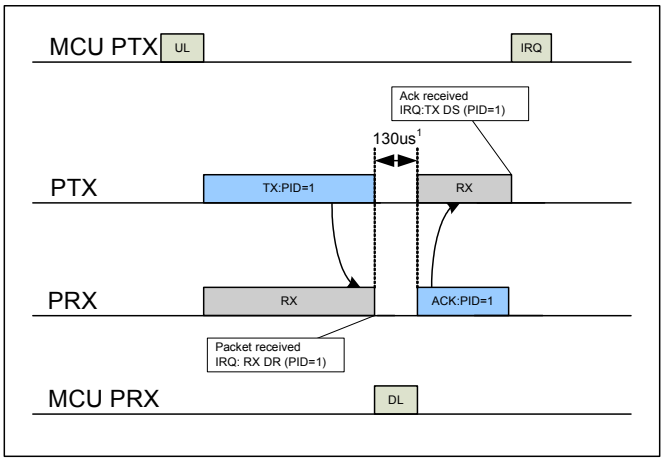


Figure 19. TX/RX cycles with ACK and the according interrupts

6.4.10.2 Single transaction with a lost packet

Figure 20. is a scenario where a retransmission is needed due to loss of the first packet transmit. After the packet is transmitted, the PTX enters RX mode to receive the ACK packet. After the first transmission, the PTX waits a specified time for the ACK packet, if it is not in the specific time slot the PTX retransmits the packet as shown in Figure 20.

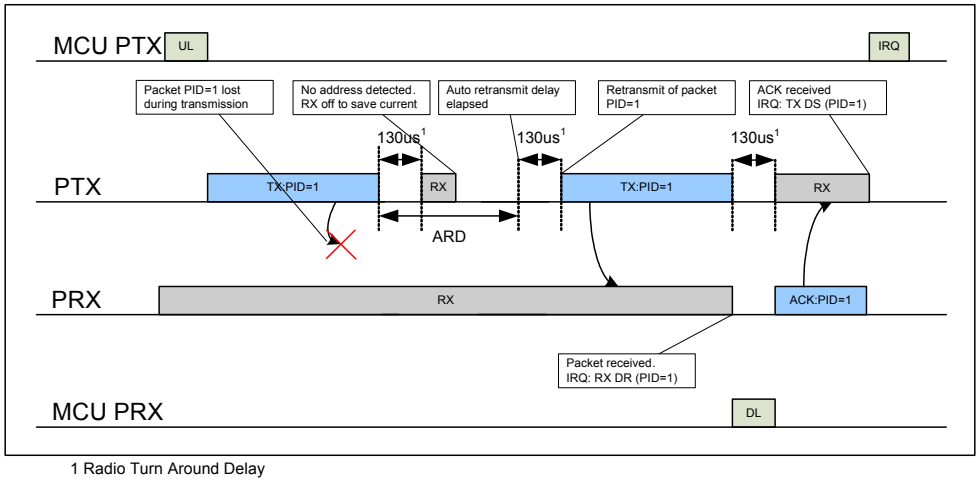


Figure 20. TX/RX cycles with ACK and the according interrupts when the first packet transmit fails

When an address is detected the PTX stays in RX mode until the packet is received. When the retransmitted packet is received by the PRX (see Figure 20.) , the RX_DR IRQ is asserted and an ACK is transmitted back to the PTX. When the ACK is received by the PTX, the TX_DS IRQ is asserted.

6.4.10.3 Single transaction with a lost ACK packet

Figure 21. is a scenario where a retransmission is needed after a loss of the ACK packet. The corresponding interrupts are also indicated.

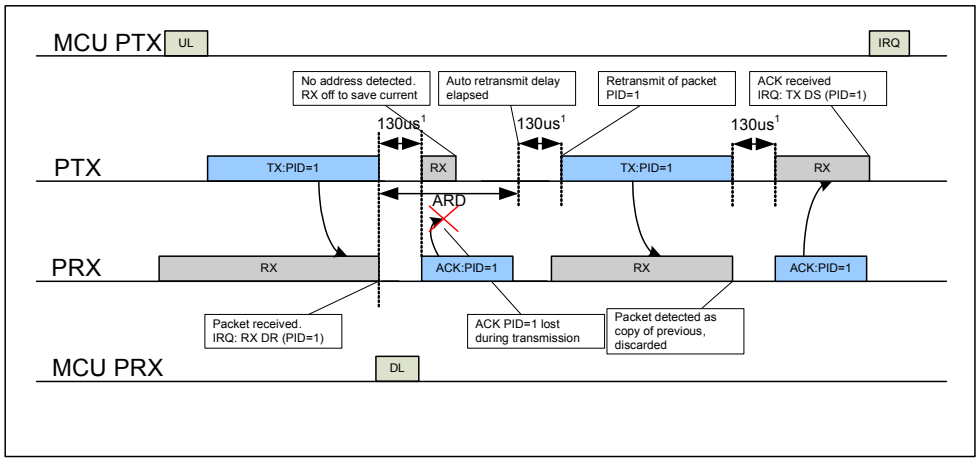


Figure 21. TX/RX cycles with ACK and the according interrupts when the ACK packet fails

6.4.10.4 Single transaction with ACK payload packet

Figure 22. is a scenario of the basic auto acknowledgement with payload. After the packet is transmitted by the PTX and received by the PRX the ACK packet with payload is transmitted from the PRX to the PTX. The RX_DR IRQ is asserted after the packet is received by the PRX, whereas on the PTX side the TX_DS IRQ is asserted when the ACK packet is received by the PTX. On the PRX side, the TX_DS IRQ for the ACK packet payload is asserted after a new packet from PTX is received. The position of the IRQ in Figure 22. shows where the MCU can respond to the interrupt.

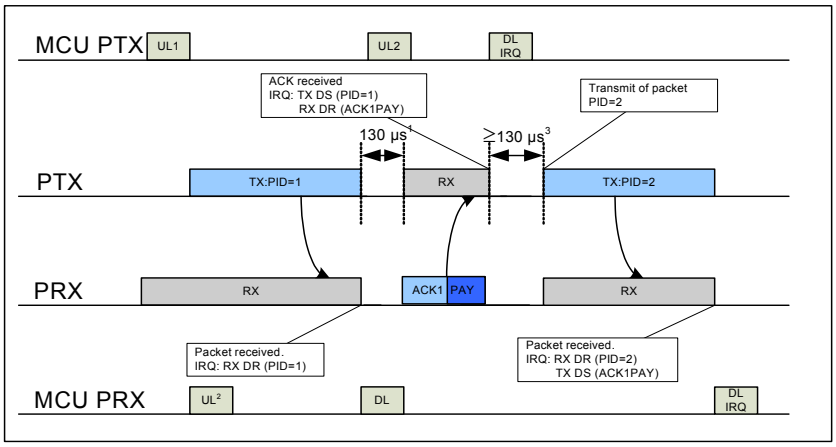
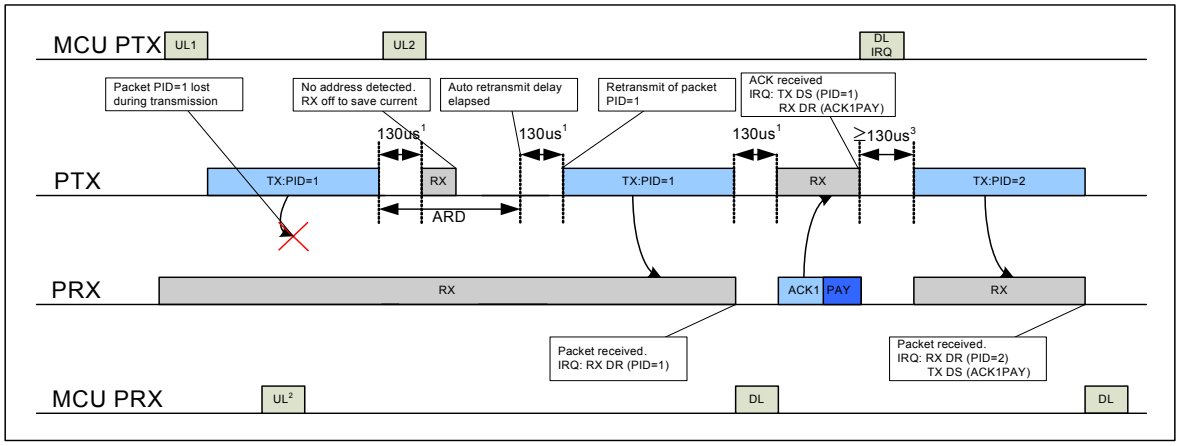


Figure 22. TX/RX cycles with ACK Payload and the according interrupts

6.4.10.5 Single transaction with ACK payload packet and lost packet

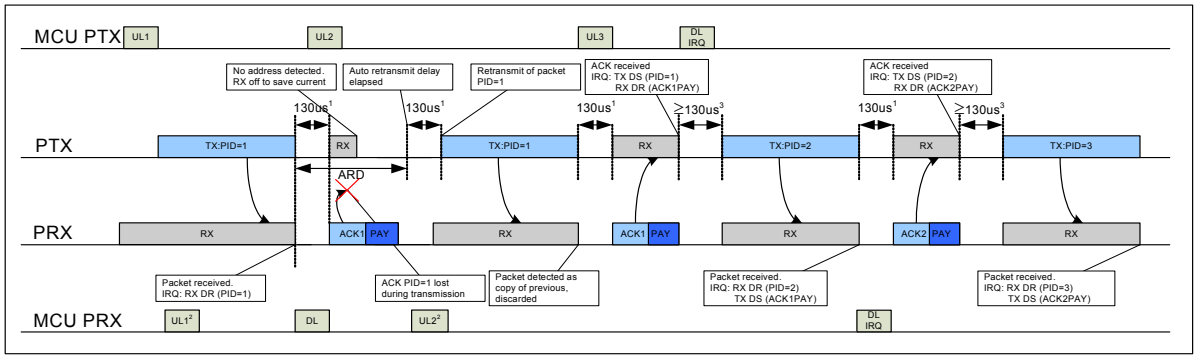
Figure 23. is a scenario where the first packet is lost and a retransmission is needed before the RX_DR IRQ on the PRX side is asserted. For the PTX both the TX_DS and RX_DR IRQ are asserted after the ACK packet is received. After the second packet (PID=2) is received on the PRX side both the RX_DR (PID=2) and TX_DS (ACK packet payload) IRQ are asserted.



1 Radio Turn Around Delay
2 Uploading Paylod for Ack Packet
3 Delay defined by MCU on PTX side, ≥ 130us

Figure 23. TX/RX cycles and the according interrupts when the packet transmission fails

6.4.10.6 Two transactions with ACK payload packet and the first ACK packet lost

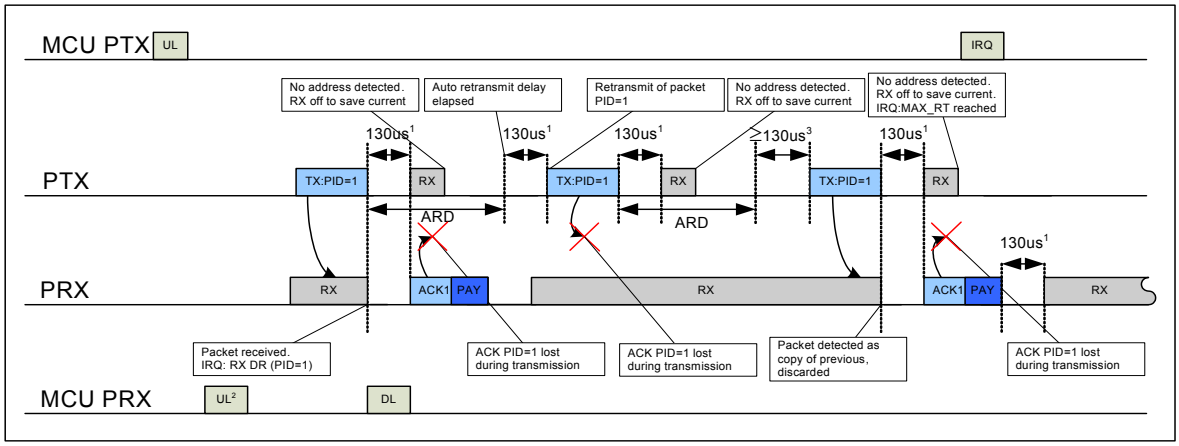


1 Radio Turn Around Delay
2 Uploading Payload for Ack Packet
3 Delay defined by MCU on PTX side, ≥ 130us

Figure 24. TX/RX cycles with ACK Payload and the according interrupts when the ACK packet fails

In Figure 24. the ACK packet is lost and a retransmission is needed before the TX_DS IRQ is asserted, but the RX_DR IRQ is asserted immediately. The retransmission of the packet (PID=1) results in a discarded packet. For the PTX both the TX_DS and RX_DR IRQ are asserted after the second transmission of ACK, which is received. After the second packet (PID=2) is received on the PRX both the RX_DR (PID=2) and TX_DS (ACK1PAY) IRQ is asserted. The callouts explains the different events and interrupts.

6.4.10.7 Two transactions where max retransmissions is reached



1 Radio Turn Around Delay
2 Uploading Paylod for Ack Packet
3 Delay defined by MCU on PTX side, ≥ 130us

Figure 25. TX/RX cycles with ACK Payload and the according interrupts when the transmission fails. ARC is set to 2.

`MAX_RT` IRQ is asserted if the auto retransmit counter (`ARC_CNT`) exceeds the programmed maximum limit (ARC). In Figure 25. the packet transmission ends with a `MAX_RT` IRQ. The payload in TX FIFO is NOT removed and the MCU decides the next step in the protocol. A toggle of the `rfce` bit in the `RFCON` register starts a new transmitting sequence of the same packet. The payload can be removed from the TX FIFO using the `FLUSH_TX` command.

6.4.11 Compatibility with ShockBurst™

You must disable Enhanced ShockBurst™ for backward compatibility with the nRF2401A, nRF2402, nRF24E1 and, nRF24E2. Set the register `EN_AA` = 0x00 and `ARC` = 0 to disable Enhanced ShockBurst™. In addition, the RF Transceiver air data rate must be set to 1 Mbps or 250 kbps.

6.4.11.1 ShockBurst™ packet format

The ShockBurst™ packet format is described in this chapter. Figure 26. shows the packet format with MSB to the left.

Preamble 1 byte	Address 3-5 byte	Payload 1 - 32 byte	CRC 1-2 byte

Figure 26. A ShockBurst™ packet compatible with nRF2401/nRF2402/nRF24E1/nRF24E2 devices.

The ShockBurst™ packet format has a preamble, address, payload and CRC field that are the same as the Enhanced ShockBurst™ packet format described in section 6.4.3 on page 33.

The differences between the ShockBurst™ packet and the Enhanced ShockBurst™ packet are:

- The 9-bit Packet Control Field is not present in the ShockBurst™ packet format.
- The CRC is optional in the ShockBurst™ packet format and is controlled by the EN_CRC bit in the CONFIG register.

6.5 Data and control interface

The data and control interface gives you access to all the features in the RF Transceiver. Compared to the standalone nRF24L01+ chip, SFR registers are used instead of port pins, so that the SFR RCON bits rfcsn, rfce and rfcken control the **CSN**, **CE** and **CKEN** pins of the standalone component.

6.5.1 SFR registers

The MCU uses an internal SPI to communicate with the RF Transceiver. This SPI is controlled by the SFR registers shown in the tables below.

Address	Reset value	Bit	Name	R/W	Function
0xE5	0x00		data	RW	SPI data input/output

Table 19. RFDAT register

Address	Reset value	Bit	Name	R/W	Function
0xE6	0x00	7:5	-		Must be zero
		4	ss	RW	SPI enable: 0: disable, 1: enable
		3:0	rfctl	RW	Divider factor from MCU clock (Cclk) to SPI clock frequency 000X: 1/2 of Cclk frequency 0010: 1/4 of Cclk frequency 0011: 1/8 of Cclk frequency 0100: 1/16 of Cclk frequency 0101: 1/32 of Cclk frequency other: 1/64 of Cclk frequency

Table 20. RFCTL register

Address	Reset value	Bit	Name	R/W	Function
0x90	0x02	7:3	-		Reserved
		2	rfcken	RW	RF Clock Enable (16 MHz)
		1	rfcsn	RW	RF SPI CSN 0: enabled 1: disabled
		0	rfce	RW	RF CE 1: enabled 0: disabled

Table 21. RFCON register

6.5.2 SPI operation

This section describes the SPI commands and timing.

6.5.2.1 SPI commands

The SPI commands are shown in Table 22. Every new command must be started by writing 0 to rfcsn in the RFCON register.

The SPI command is transferred to RF Transceiver by writing the command to the RFDAT register. After the first transfer the RF Transceiver's STATUS register can be read from RFDAT when the transfer is completed.

The serial shifting SPI commands is in the following format:

<**Command word**: MSBit to LSBit (one byte)>

<**Data bytes**: LSByte to MSByte, MSBit in each byte first>

Command name	Command word (binary)	# Data bytes	Operation
R_REGISTER	000A AAAA	1 to 5 LSByte first	Read command and status registers. AAAAA = 5 bit Register Map Address
W_REGISTER	001A AAAA	1 to 5 LSByte first	Write command and status registers. AAAAA = 5 bit Register Map Address Executable in power down or standby modes only.
R_RX_PAYLOAD	0110 0001	1 to 32 LSByte first	Read RX-payload: 1 – 32 bytes. A read operation always starts at byte 0. Payload is deleted from FIFO after it is read. Used in RX mode.
W_TX_PAYLOAD	1010 0000	1 to 32 LSByte first	Write TX-payload: 1 – 32 bytes. A write operation always starts at byte 0 used in TX payload.
FLUSH_TX	1110 0001	0	Flush TX FIFO, used in TX mode
FLUSH_RX	1110 0010	0	Flush RX FIFO, used in RX mode Should not be executed during transmission of acknowledge, that is, acknowledge package will not be completed.
REUSE_TX_PL	1110 0011	0	Used for a PTX operation Reuse last transmitted payload. TX payload reuse is active until W_TX_PAYLOAD or FLUSH TX is executed. TX payload reuse must not be activated or deactivated during package transmission.
R_RX_PL_WID[a]	0110 0000	1	Read RX payload width for the top R_RX_PAYLOAD in the RX FIFO. **Note:** Flush RX FIFO if the read value is larger than 32 bytes.

Command name	Command word (binary)	# Data bytes	Operation
W_ACK_PAYLOAD[a]	1010 1PPP	1 to 32 LSByte first	Used in RX mode. Write Payload to be transmitted together with ACK packet on PIPE PPP. (PPP valid in the range from 000 to 101). Maximum three ACK packet payloads can be pending. Payloads with same PPP are handled using first in - first out principle. Write payload: 1– 32 bytes. A write operation always starts at byte 0.
W_TX_PAYLOAD_NOACK	1011 0000	1 to 32 LSByte first	Used in TX mode. Disables AUTOACK on this specific packet.
NOP	1111 1111	0	No Operation. Might be used to read the STATUS register

a. The bits in the FEATURE register shown in Table 23. have to be set.

Table 22. Command set for the RF Transceiver SPI

The W_REGISTER and R_REGISTER commands operate on single or multi-byte registers. When accessing multi-byte registers read or write to the MSBit of LSByte first. You can terminate the writing before all bytes in a multi-byte register are written, leaving the unwritten MSByte(s) unchanged. For example, the LSByte of RX_ADDR_P0 can be modified by writing only one byte to the RX_ADDR_P0 register. The content of the status register is always read to **MISO** after a high to low transition on **CSN**.

Note: The 3-bit pipe information in the STATUS register is updated during the RFIRQ high to low transition. The pipe information is unreliable if the STATUS register is read during an RFIRQ high to low transition.

6.5.3 Data FIFO

The data FIFOs store transmitted payloads (TX FIFO) or received payloads that are ready to be clocked out (RX FIFO). The FIFOs are accessible in both PTX mode and PRX mode.

The following FIFOs are present in the RF Transceiver:

- TX three level, 32 byte FIFO
- RX three level, 32 byte FIFO

Both FIFOs have a controller and are accessible through the SPI by using dedicated SPI commands. A TX FIFO in PRX can store payloads for ACK packets to three different PTX operations. If the TX FIFO contains more than one payload to a pipe, payloads are handled using the first in - first out principle. The TX FIFO in a PRX is blocked if all pending payloads are addressed to pipes where the link to the PTX is lost. In this case, the MCU can flush the TX FIFO using the FLUSH_TX command.

The RX FIFO in PRX can contain payloads from up to three different PTX operations and a TX FIFO in PTX can have up to three payloads stored.

You can write to the TX FIFO using these three commands; W_TX_PAYLOAD and W_TX_PAYLOAD_NO_ACK in PTX mode and W_ACK_PAYLOAD in PRX mode. All three commands provide access to the TX_PLD register.

The RX FIFO can be read by the command R_RX_PAYLOAD in PTX and PRX mode. This command provides access to the RX_PLD register.

The payload in TX FIFO in a PTX is not removed if the MAX_RT IRQ is asserted.

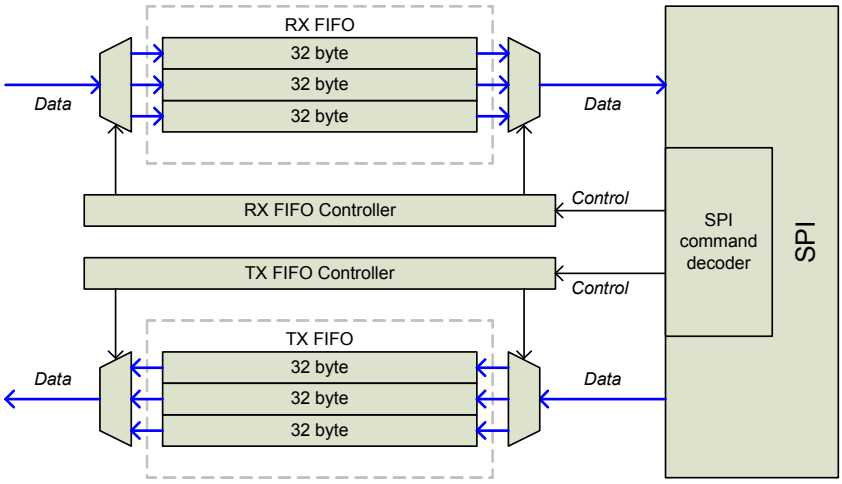


Figure 27. FIFO (RX and TX) block diagram

You can read if the TX and RX FIFO are full or empty in the FIFO_STATUS register. TX_REUSE (also available in the FIFO_STATUS register) is set by the SPI command REUSE_TX_PL, and is reset by the SPI commands W_TX_PAYLOAD or FLUSH TX.

6.5.4 Interrupt

The RF Transceiver can send interrupts to the MCU. The interrupt (RFIRQ) is activated when TX_DS, RX_DR or MAX_RT are set high by the state machine in the STATUS register. RFIRQ is deactivated when the MCU writes '1' to the interrupt source bit in the STATUS register. The interrupt mask in the CONFIG register is used to select the IRQ sources that are allowed to activate RFIRQ. By setting one of the mask bits high, the corresponding interrupt source is disabled. By default all interrupt sources are enabled.

Note: The 3-bit pipe information in the STATUS register is updated during the RFIRQ high to low transition. The pipe information is unreliable if the STATUS register is read during a RFIRQ high to low transition.

6.6 Register map

You can configure and control the radio (using read and write commands) by accessing the register map through the SPI.

6.6.1 Register map table

All undefined bits in the table below are redundant. They are read out as '0'.

Note: Addresses 18 to 1B are reserved for test purposes, altering them makes the chip malfunction.

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
00	CONFIG				Configuration Register
	Reserved	7	0	R/W	Only '0' allowed
	MASK_RX_DR	6	0	R/W	Mask interrupt caused by RX_DR 1: Interrupt not reflected on the RFIRQ 0: Reflect RX_DR as active low on RFIRQ
	MASK_TX_DS	5	0	R/W	Mask interrupt caused by TX_DS 1: Interrupt not reflected on the RFIRQ 0: Reflect TX_DS as active low interrupt on RFIRQ
	MASK_MAX_RT	4	0	R/W	Mask interrupt caused by MAX_RT 1: Interrupt not reflected on RFIRQ 0: Reflect MAX_RT as active low on RFIRQ
	EN_CRC	3	1	R/W	Enable CRC. Forced high if one of the bits in the EN_AA is high
	CRCO	2	0	R/W	CRC encoding scheme '0' - 1 byte '1' – 2 bytes
	PWR_UP	1	0	R/W	1: POWER UP, 0:POWER DOWN
	PRIM_RX	0	0	R/W	RX/TX control 1: PRX, 0: PTX
01	EN_AA Enhanced ShockBurst™				Enable 'Auto Acknowledgment' Function Disable this functionality to be compatible with nRF2401.
	Reserved	7:6	00	R/W	Only '00' allowed
	ENAA_P5	5	1	R/W	Enable auto acknowledgement data pipe 5
	ENAA_P4	4	1	R/W	Enable auto acknowledgement data pipe 4
	ENAA_P3	3	1	R/W	Enable auto acknowledgement data pipe 3
	ENAA_P2	2	1	R/W	Enable auto acknowledgement data pipe 2
	ENAA_P1	1	1	R/W	Enable auto acknowledgement data pipe 1
	ENAA_P0	0	1	R/W	Enable auto acknowledgement data pipe 0
02	EN_RXADDR				Enabled RX Addresses
	Reserved	7:6	00	R/W	Only '00' allowed
	ERX_P5	5	0	R/W	Enable data pipe 5.
	ERX_P4	4	0	R/W	Enable data pipe 4.
	ERX_P3	3	0	R/W	Enable data pipe 3.
	ERX_P2	2	0	R/W	Enable data pipe 2.
	ERX_P1	1	1	R/W	Enable data pipe 1.
	ERX_P0	0	1	R/W	Enable data pipe 0.

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
03	SETUP_AW				Setup of Address Widths (common for all data pipes)
	Reserved	7:2	000000	R/W	Only '000000' allowed
	AW	1:0	11	R/W	RX/TX Address field width '00' - Illegal '01' - 3 bytes '10' - 4 bytes '11' – 5 bytes LSByte is used if address width is below 5 bytes
04	SETUP_RETR				Setup of Automatic Retransmission
	ARD[a]	7:4	0000	R/W	Auto Retransmit Delay ‘0000’ – Wait 250 µs ‘0001’ – Wait 500 µs ‘0010’ – Wait 750 µs …….. ‘1111’ – Wait 4000 µs (Delay defined from end of transmission to start of next transmission)[b]
	ARC	3:0	0011	R/W	Auto Retransmit Count ‘0000’ –Re-Transmit disabled ‘0001’ – Up to 1 Re-Transmit on fail of AA …… ‘1111’ – Up to 15 Re-Transmit on fail of AA
05	RF_CH				RF Channel
	Reserved	7	0	R/W	Only '0' allowed
	RF_CH	6:0	0000010	R/W	Sets the frequency channel the RF Transceiver operates on
06	RF_SETUP				RF Setup Register
	CONT_WAVE	7	0	R/W	Enables continuous carrier transmit when high.
	Reserved	6	0	R/W	Only '0' allowed
	RF_DR_LOW	5	0	R/W	Set RF Data Rate to 250 kbps. See RF_DR_HIGH for encoding.
	PLL_LOCK	4	0	R/W	Force PLL lock signal. Only used in test
	RF_DR_HIGH	3	1	R/W	Select between the high speed data rates. This bit is don’t care if RF_DR_LOW is set. Encoding: RF_DR_LOW, RF_DR_HIGH: ‘00’ – 1 Mbps ‘01’ – 2 Mbps ‘10’ – 250 kbps ‘11’ – Reserved
	RF_PWR	2:1	11	R/W	Set RF output power in TX mode '00' – -18dBm '01' – -12dBm '10' – -6dBm '11' – 0dBm

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
	Obsolete	0			Don't care
07	STATUS				Status Register (In parallel to the SPI command word applied on the **MOSI** pin, the STATUS register is shifted serially out on the **MISO** pin)
	Reserved	7	0	R/W	Only '0' allowed
	RX_DR	6	0	R/W	Data Ready RX FIFO interrupt. Asserted when new data arrives RX FIFO[c]. Write 1 to clear bit.
	TX_DS	5	0	R/W	Data Sent TX FIFO interrupt. Asserted when packet transmitted on TX. If AUTO_ACK is activated, this bit is set high only when ACK is received. Write 1 to clear bit.
	MAX_RT	4	0	R/W	Maximum number of TX retransmits interrupt Write 1 to clear bit. If MAX_RT is asserted it must be cleared to enable further communication.
	RX_P_NO	3:1	111	R	Data pipe number for the payload available for reading from RX_FIFO 000-101: Data Pipe Number 110: Not Used 111: RX FIFO Empty
	TX_FULL	0	0	R	TX FIFO full flag. 1: TX FIFO full. 0: Available locations in TX FIFO.
08	OBSERVE_TX				Transmit observe register
	PLOS_CNT	7:4	0	R	Count lost packets. The counter is overflow protected to 15, and discontinues at max until reset. The counter is reset by writing to RF_CH.
	ARC_CNT	3:0	0	R	Count retransmitted packets. The counter is reset when transmission of a new packet starts.
09	RPD				
	Reserved	7:1	000000	R	
	RPD	0	0	R	Received Power Detector. This register is called CD (Carrier Detect) in the nRF24L01. The name is different in the RF Transceiver due to the different input power level threshold for this bit. See section 6.3.4 on page 31.
0A	RX_ADDR_P0	39:0	0xE7E7E7E7E7	R/W	Receive address data pipe 0. 5 Bytes maximum length. (LSByte is written first. Write the number of bytes defined by SETUP_AW)
0B	RX_ADDR_P1	39:0	0xC2C2C2C2C2	R/W	Receive address data pipe 1. 5 Bytes maximum length. (LSByte is written first. Write the number of bytes defined by SETUP_AW)
0C	RX_ADDR_P2	7:0	0xC3	R/W	Receive address data pipe 2. Only LSB. MSBytes are equal to RX_ADDR_P1 39:8
0D	RX_ADDR_P3	7:0	0xC4	R/W	Receive address data pipe 3. Only LSB. MSBytes are equal to RX_ADDR_P139:8

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
0E	RX_ADDR_P4	7:0	0xC5	R/W	Receive address data pipe 4. Only LSB. MSBytes are equal to RX_ADDR_P139:8
0F	RX_ADDR_P5	7:0	0xC6	R/W	Receive address data pipe 5. Only LSB. MSBytes are equal to RX_ADDR_P139:8
10	TX_ADDR	39:0	0xE7E7E7E7E7	R/W	Transmit address. Used for a PTX operation only. (LSByte is written first) Set RX_ADDR_P0 equal to this address to handle automatic acknowledge if this is a PTX operation with Enhanced ShockBurst™ enabled.
11	RX_PW_P0				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P0	5:0	0	R/W	Number of bytes in RX payload in data pipe 0 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
12	RX_PW_P1				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P1	5:0	0	R/W	Number of bytes in RX payload in data pipe 1 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
13	RX_PW_P2				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P2	5:0	0	R/W	Number of bytes in RX payload in data pipe 2 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
14	RX_PW_P3				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P3	5:0	0	R/W	Number of bytes in RX payload in data pipe 3 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
15	RX_PW_P4				
	Reserved	7:6	00	R/W	Only '00' allowed

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
	RX_PW_P4	5:0	0	R/W	Number of bytes in RX payload in data pipe 4 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
16	RX_PW_P5				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P5	5:0	0	R/W	Number of bytes in RX payload in data pipe 5 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
17	FIFO_STATUS				FIFO Status Register
	Reserved	7	0	R/W	Only '0' allowed
	TX_REUSE	6	0	R	Used for a PTX operation Pulse the rfce high for at least 10µs to Reuse last transmitted payload. TX payload reuse is active until W_TX_PAYLOAD or FLUSH TX is executed. TX_REUSE is set by the SPI command REUSE_TX_PL, and is reset by the SPI commands W_TX_PAYLOAD or FLUSH TX
	TX_FULL	5	0	R	TX FIFO full flag. 1: TX FIFO full. 0: Available locations in TX FIFO.
	TX_EMPTY	4	1	R	TX FIFO empty flag. 1: TX FIFO empty. 0: Data in TX FIFO.
	Reserved	3:2	00	R/W	Only '00' allowed
	RX_FULL	1	0	R	RX FIFO full flag. 1: RX FIFO full. 0: Available locations in RX FIFO.
	RX_EMPTY	0	1	R	RX FIFO empty flag. 1: RX FIFO empty. 0: Data in RX FIFO.
N/A	ACK_PLD	255:0	X	W	Written by separate SPI command ACK packet payload to data pipe number PPP given in SPI command. Used in RX mode only. Maximum three ACK packet payloads can be pending. Payloads with same PPP are handled first in first out.
N/A	TX_PLD	255:0	X	W	Written by separate SPI command TX data payload register 1 - 32 bytes. This register is implemented as a FIFO with three levels. Used in TX mode only.

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
N/A	RX_PLD	255:0	X	R	Read by separate SPI command. RX data payload register. 1 - 32 bytes. This register is implemented as a FIFO with three levels. All RX channels share the same FIFO.
1C	DYNPD				Enable dynamic payload length
	Reserved	7:6	0	R/W	Only '00' allowed
	DPL_P5	5	0	R/W	Enable dynamic payload length data pipe 5. (Requires EN_DPL and ENAA_P5)
	DPL_P4	4	0	R/W	Enable dynamic payload length data pipe 4. (Requires EN_DPL and ENAA_P4)
	DPL_P3	3	0	R/W	Enable dynamic payload length data pipe 3. (Requires EN_DPL and ENAA_P3)
	DPL_P2	2	0	R/W	Enable dynamic payload length data pipe 2. (Requires EN_DPL and ENAA_P2)
	DPL_P1	1	0	R/W	Enable dynamic payload length data pipe 1. (Requires EN_DPL and ENAA_P1)
	DPL_P0	0	0	R/W	Enable dynamic payload length data pipe 0. (Requires EN_DPL and ENAA_P0)
1D	FEATURE			R/W	Feature Register
	Reserved	7:3	0	R/W	Only '00000' allowed
	EN_DPL	2	0	R/W	Enables Dynamic Payload Length
	EN_ACK_PAY[d]	1	0	R/W	Enables Payload with ACK
	EN_DYN_ACK	0	0	R/W	Enables the W_TX_PAYLOAD_NOACK command

a. Please take care when setting this parameter. If the ACK payload is more than 15 byte in 2 Mbps mode the ARD must be 500 µs or more, if the ACK payload is more than 5 byte in 1 Mbps mode the ARD must be 500 µs or more. In 250 kbps mode (even when the payload is not in ACK) the ARD must be 500 µs or more.

b. This is the time the PTX is waiting for an ACK packet before a retransmit is made. The PTX is in RX mode for a minimum of 250 µs, but it stays in RX mode to the end of the packet if that is longer than 250 µs. Then it goes to standby-I mode for the rest of the specified ARD. After the ARD it goes to TX mode and then retransmits the packet.

c. The RX_DR IRQ is asserted by a new packet arrival event. The procedure for handling this interrupt should be: 1) read payload through SPI, 2) clear RX_DR IRQ, 3) read FIFO_STATUS to check if there are more payloads available in RX FIFO, 4) if there are more data in RX FIFO, repeat from step 1).

d. If ACK packet payload is activated, ACK packets have dynamic payload lengths and the Dynamic Payload Length feature should be enabled for pipe 0 on the PTX and PRX. This is to ensure that they receive the ACK packets with payloads. If the ACK payload is more than 15 byte in 2 Mbps mode the ARD must be 500 µs or more, and if the ACK payload is more than 5 byte in 1 Mbps mode the ARD must be 500 µs or more. In 250 kbps mode (even when the payload is not in ACK) the ARD must be 500 µs or more.

Table 23. Register map of the RF Transceiver

7 USB Interface

The USB device controller provides a full speed USB function interface that meets the 1.1 and 2.0 revision of the USB specification. It handles byte transfers autonomously and bridges the USB interface to a simple read/write parallel interface.

7.1 Features

- Serial Interface Engine
  - Supports full speed devices
  - Extraction of clock and data signals in internal DPLL
  - NRZI decoding/encoding
  - Bit stuffing/stripping
  - CRC checking/generation
  - On-chip transceiver
  - On-chip pull-up resistor on D+ with software controlled disconnect
- 2 control, 10 bulk/interrupt and 2 ISO endpoints
  - Supports control transfers by endpoint #0
  - Supports bulk, interrupt on endpoint #1 - #5 (in/out)
  - Support double buffering for isochronous endpoint #8 (in/out)
  - Programmable double buffering for bulk and interrupt endpoints
- Automatic data retry mechanism
- Data toggle synchronization mechanism
- Suspend and resume power management functions
- Remote Wakeup function
- Flexible endpoint buffers RAM
  - 512 bytes buffer total
  - Up to 64 bytes buffer size for endpoint 0-5
  - Up to 128bytes buffer size for endpoint 8

The endpoint set up allows for five different applications (for example, Mouse, Keyboard, Remote Control, Gamepad and Joystick) to use both input and output data transfer on separate endpoints.

The nRF24LU1+ supports a total of 14 endpoints. EP0 IN/OUT supports input and output control data transfer, EP1-5 IN/OUT supports input and output bulk and interrupt data transfer. In addition, EP8 IN/OUT can be configured for input and output isynchronous data transfer. These two endpoints share memory buffer area with EP0-5. This sharing is controlled by nRF24LU1+ firmware.

7.2 Block diagram

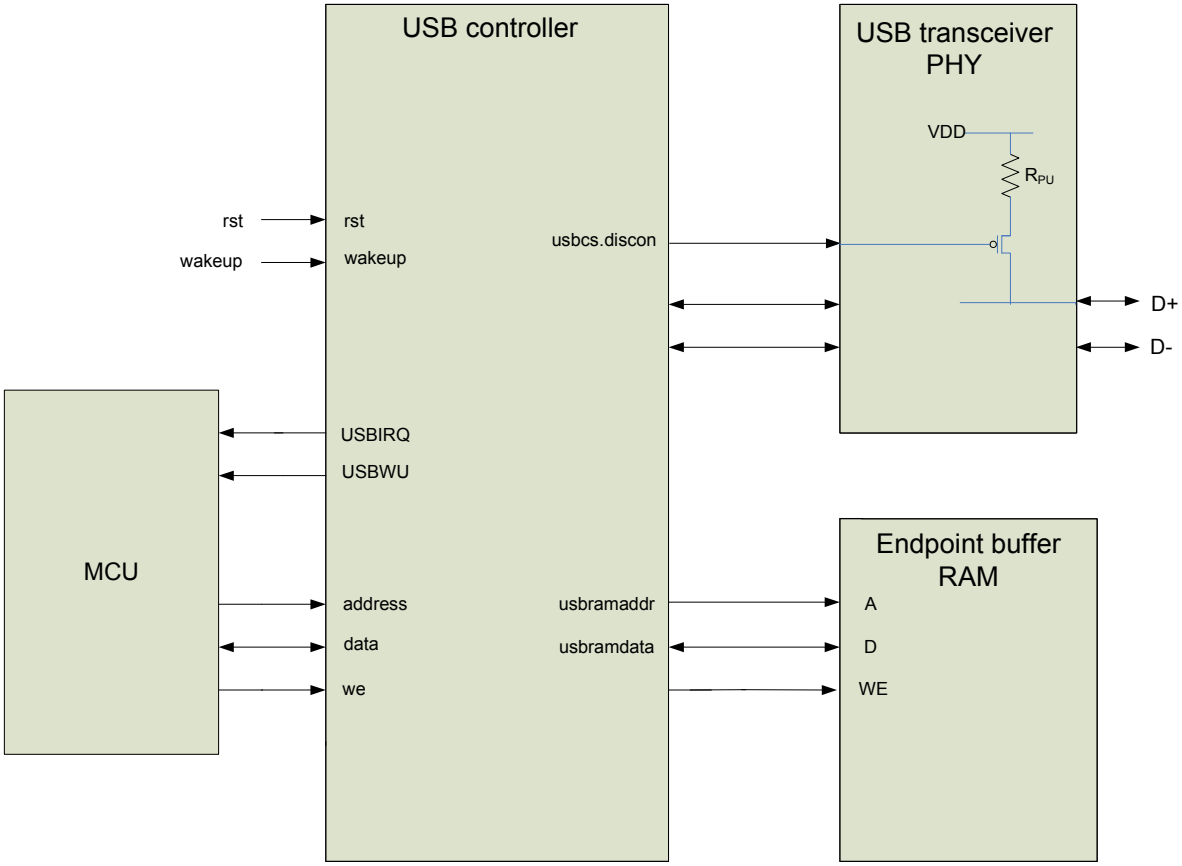


Figure 28. USB block diagram

7.3 Functional description

The USB module is designed to serve as a Full Speed (FS) USB device as defined in the Universal Serial Bus Specification Rev 2.0. It is controlled both with SFR registers and XDATA mapped registers. There are two SFR registers, USBCON and USBSLP, and the rest of the registers are XDATA mapped registers.

Address	Reset value	Bit	Name	R/W	Description
0xA0	0x00	7	swrst	RW	1: reset USB
		6	wu	RW	1: wakeup USB, must be cleared before setting USBSLP.
		5	suspend	R	1: USB is suspended. This bit acknowledges USBSLP=1, after a delay of up to 32 µs.
		4:0	ivec[6:2]	R	Interrupt vector ivec, see Table 34. on page 84

Table 24. USBCON register

Address	Reset value	Bit	Name	R/W	Description
0xD9	0x00	7:1	-	-	Not used
		0	Sleep	WO	1: Disable USB clock, bit automatically cleared.

Set wu=1 in USBCON to enable USB clock

Table 25. USBSLP register

The other USB registers and buffer RAM are accessible through a 2k “window” in XDATA space using the MOVX instruction.

Note: Undefined addresses should not be written or read.

Note: Key to abbreviations used in Table 27. on page 70:
- u - unchanged value after reset
- x - unknown

Hex address	Name	Hex hard reset	USB reset	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
C440-C47F[a]	out5buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C480-C4BF	in5buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C4C0-C4FF	out4buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C500-C53F	in4buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C540-C57F	out3buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C580-C5BF	in3buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C5C0-C5FF	out2buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0

Hex address	Name	Hex hard reset	USB reset	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
C600-C63F	in2buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C640-C67F	out1buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C680-C6BF	in1buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C6C0-C6FF	out0buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C700-C73F	in0buf	x	x	d7	d6	d5	d4	d3	d2	d1	d0
C760	out8data	x	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C768	in8data	x	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C770	out8bch	00	uuuuuuuu	0	0	0	0	0	0	bc9	bc8
C771	out8bcl	00	uuuuuuuu	bc7	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C781	bout1addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C782	bout2addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C783	bout3addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C784	bout4addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C785	bout5addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C788	binstaddr	00	uuuuuuuu	addr9	addr8	addr7	addr6	addr5	addr4	addr3	addr2
C789	bin1addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C78A	bin2addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C78B	bin3addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C78C	bin4addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C78D	bin5addr	00	uuuuuuuu	addr8	addr7	addr6	addr5	addr4	addr3	addr2	addr1
C7A0	isoerr	00	uuuuuuuu	0	0	0	0	0	0	0	iso8err
C7A2	zbcout	00	uuuuuuuu	0	0	0	0	0	0	0	ep8
C7A8	ivec	00	uuuuuuuu	0	iv4	iv3	iv2	iv1	iv0	0	0
C7A9	in_irq	00	uuuuuuuu	0	0	in5ir	in4ir	in3ir	in2ir	in1ir	in0ir
C7AA	out_irq	00	uuuuuuuu	0	0	out5ir	out4ir	out3ir	out2ir	out1ir	out0ir

Hex address	Name	Hex hard reset	USB reset	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
C7AB	usbirq	00	uuuuuuuu	0	0	ibnir	uresir	suspir	sutokir	sofir	sudavir
C7AC	in_ien	00	uuuuuuuu	0	0	in5ien	in4ien	in3ien	in2ien	in1ien	in0ien
C7AD	out_ien	00	uuuuuuuu	0	0	out5ien	out4ien	out3ien	out2ien	out1ien	out0ien
C7AE	usbien	00	uuuuuuuu	0	0	ibnie	uresie	suspie	sutokie	sofie	sudavie
C7AF	usbbav	00	uuuuuuuu	0	0	0	0	0	0	0	aven
C7B4	ep0cs	08	uuuuu0uu	0	0	chgset	dstall	outbsy	inbsy	hsnak	ep0stall
C7B5	in0bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7B6	in1cs	00	uuuuuu00	0	0	0	0	0	0	in1bsy	in1stl
C7B7	in1bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7B8	in2cs	00	uuuuuu00	0	0	0	0	0	0	in2bsy	in2stl
C7B9	in2bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7BA	in3cs	00	uuuuuu00	0	0	0	0	0	0	in3bsy	in3stl
C7BB	in3bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7BC	in4cs	00	uuuuuu00	0	0	0	0	0	0	in4bsy	in4stl
C7BD	in4bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7BE	in5cs	00	uuuuuu00	0	0	0	0	0	0	in5bsy	in5stl
C7BF	in5bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7C5	out0bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7C6	out1cs	02	uuuuuuuu	0	0	0	0	0	0	out1bsy	out1stl
C7C7	out1bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7C8	out2cs	02	uuuuuuuu	0	0	0	0	0	0	out2bsy	out2stl
C7C9	out2bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7CA	out3cs	02	uuuuuuuu	0	0	0	0	0	0	out3bsy	out3stl
C7CB	out3bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7CC	out4cs	02	uuuuuuuu	0	0	0	0	0	0	out4bsy	out4stl

Hex address	Name	Hex hard reset	USB reset	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
C7CD	out4bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7CE	out5cs	02	uuuuuuuu	0	0	0	0	0	0	out5bsy	out5stl
C7CF	out5bc	00	uuuuuuuu	0	bc6	bc5	bc4	bc3	bc2	bc1	bc0
C7D6	usbcs	00	uuuuuuuu	wakesrc	0	sofgen	0	discon	0	forcej	sigr-sume
C7D7	togctl[b]	00	uuuuuuuu	q	s	r	io	0	ep2	ep1	ep0
C7D8	usbfrml	00	uuuuuuuu	fc7	fc6	fc5	fc4	fc3	fc2	fc1	fc0
C7D9	usbfrmh	00	uuuuuuuu	0	0	0	0	0	fc10	fc9	fc8
C7DB	fnaddr	00	00000000	0	fa6	fa5	fa4	fa3	fa2	fa1	fa0
C7DD	usbpair	00	uuuuuuuu	isosend0	0	0	pr4out	pr2out	0	pr4in	pr2in
C7DE	inbulkval	57	uuuuuuuu	0	0	in5val	in4val	in3val	in2val	in1val	1
C7DF	outbulkval	55	uuuuuuuu	0	0	out5val	out4val	out3val	out2val	out1val	1
C7E0	inisoval	07	uuuuuuuu	0	0	0	0	0	0	0	in8val
C7E1	outisoval	07	uuuuuuuu	0	0	0	0	0	0	0	out8val
C7E2	isostaddr	00	uuuuuuuu	0	addr10	addr9	addr8	addr7	addr6	addr5	addr4
C7E3	isosize	00	uuuuuuuu	0	size10	size9	size8	size7	size6	size5	size4
C7E8	setupbuf	00	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C7E9	setupbuf	00	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C7EA	setupbuf	00	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C7EB	setupbuf	00	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C7EC	setupbuf	00	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C7ED	setupbuf	00	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0

Hex address	Name	Hex hard reset	USB reset	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
C7EE	setupbuf	00	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C7EF	setupbuf	00	uuuuuuuu	d7	d6	d5	d4	d3	d2	d1	d0
C7F0	out8addr	00	uuuuuuuu	a9	a8	a7	a6	a5	a4	0	0
C7F8	in8addr	00	uuuuuuuu	a9	a8	a7	a6	a5	a4	0	0

a. The addresses for outxbuf and inxbuf are indirect addresses which are mapped according to the end-point definitions given in register boutxaddr and binxaddr.
b. See also section 7.5.4

Note: Key to abbreviations used in the table:

- U - unchanged value after reset
- Un - unknown

Table 26. USB buffer and register map

7.4 Control endpoints

Each USB device is allocated by endpoint numbers. The endpoint 0 (EP0) is reserved for control transfers. Using USB requests, the host uses EP0 for device configuration.

The device processes the `SET_ADDRESS` request and sets the address in the `fnaddr` register. Firmware interrupts this request as configured in the `usbien` register.

All other USB device requests must be processed by firmware.

7.4.1 Control endpoint 0 implementation

Every USB device must have the endpoint 0, it is the special control endpoint.

7.4.2 Endpoint 0 registers

Register name	Bit name	Bit description
usbien(0)	sudavie	Setup data valid interrupt enable
usbien(2)	sutokie	Setup token interrupt enable
usbirq(0)	sudavir	Setup data valid interrupt request
usbirq(2)	sutokir	Setup token interrupt request
setupdat0 setupdat7	Setup Data Buffer	8 bytes setup data packet
in_irq(0)	in0ir	IN 0 endpoint interrupt request
out_irq(0)	out0ir	OUT 0 endpoint interrupt request
in_ien(0)	in0ien	IN 0 endpoint interrupt enable
out_ien(0)	out0ien	OUT 0 endpoint interrupt enable
ep0cs(0)	ep0stall	Endpoint 0 STALL bit
ep0cs(1)	hsnak	Handshake NAK
ep0cs(2)	inbsy	IN 0 buffer busy flag
ep0cs(3)	outbsy	OUT 0 buffer busy flag
ep0cs(4)	dstall	Send STALL in the data stage
ep0cs(5)	chgset	Setup Buffer content was changed
in0bc	Register	IN 0 byte counter
out0bc	Register	OUT 0 byte counter

Table 27. Endpoint 0 Register

7.4.3 Control transfer examples

A control transfer consists of two or three stages:

- Setup stage
- Data stage (optional)
- Status stage

7.4.3.1 Control write transfer example

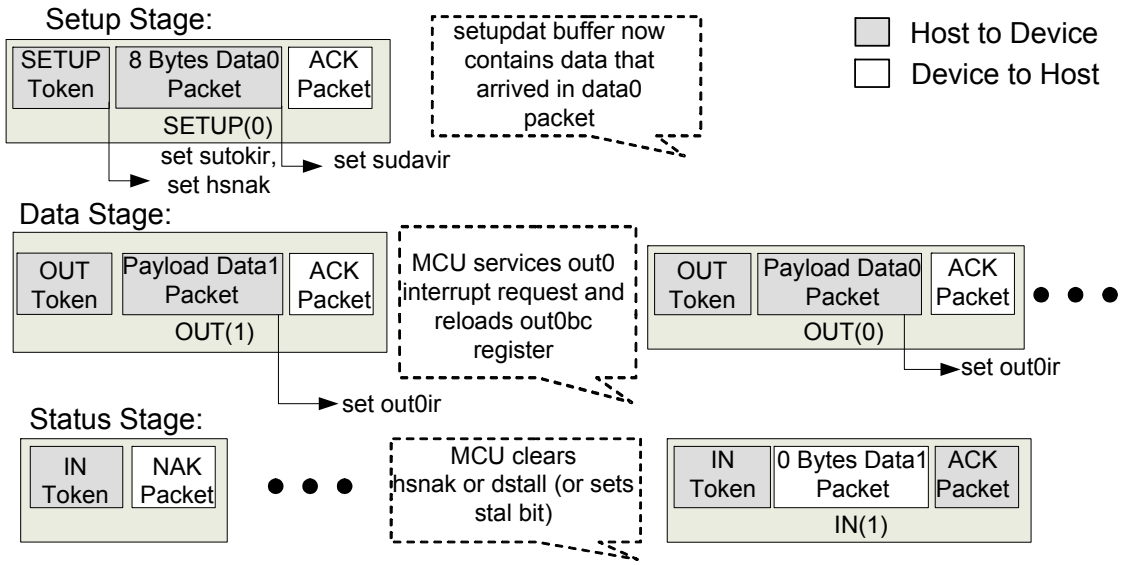


Figure 29. Control Write Transfer

After receiving the SETUP token, the USB controller sets the hsnak and sutokir bits. If an 8-byte data packet is received correctly, the USB controller sets the sudavir bit. Setting sutokir and (or) sudavir bits generates the appropriate interrupts. The data stage consists of one or more OUT bulk-like transactions.

The USB controller generates the OUT 0 interrupt request by setting the out0ir bit after each correct OUT transaction during the data stage. Out0bc register contains the number of data bytes received in the last OUT transaction. The MCU services the interrupt request and then prepares the endpoint for the next transaction by reloading the out0bc register with any value (setting `outbsy` bit). The status stage of a control transfer is the last operation in the sequence.

The MCU clears the hsnak bit (by writing 1 to it) to instruct the USB controller to ACK the status stage. The USB controller sends a STALL handshake when both hsnak and stall bits are set.

7.4.3.2 Control read transfer example

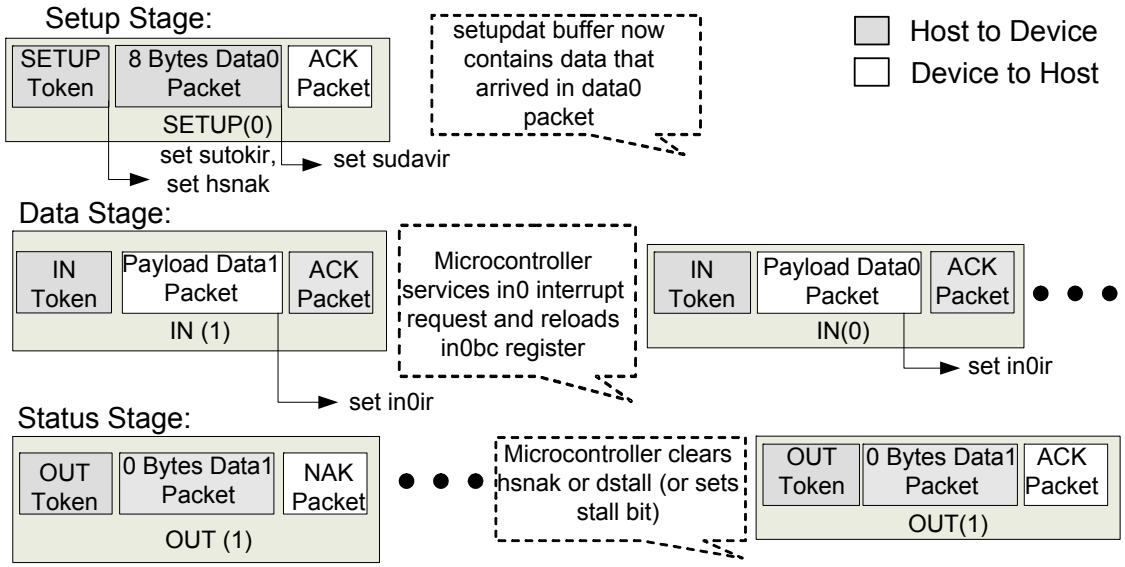


Figure 30. Control Read Transfer

Control read transfer is similar to control write transfer with the only difference in the data stage. During the data stage of control read transfers, the USB controller generates the IN 0 interrupt request by setting in0ir bit. This is done after each acknowledge by the host data packet. The MCU loads new data into the IN 0 buffer and then reloads the in0bc register with a valid number of loaded data. Reloading the in0bc register causes the inbsy bit to set and arms the endpoint for the next IN transaction.

The status stage of a control transfer is the last operation in the sequence. The MCU clears the hsnak bit (by writing 1 to it) to instruct the USB controller to ACK the status stage. The USB controller sends the STALL handshake when both hsnak and stall bits are set.

7.4.3.3 No-data control transfer example

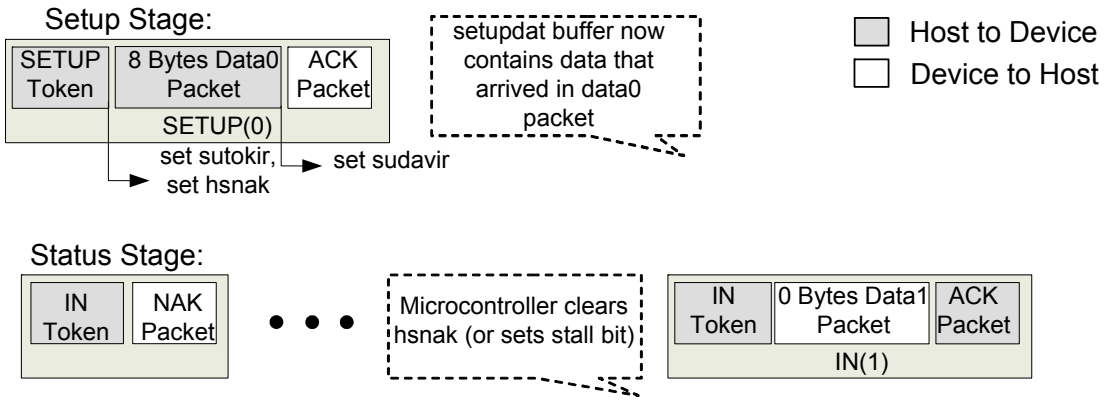


Figure 31. No-data Control Transfer

Some control transfers do not have a data stage. In this case the status stage consists of the IN data packet. The MCU clears the hsnak bit (by writing 1 to it) to instruct the USB controller to ACK (acknowledge) the status stage.

7.5 Bulk/Interrupt endpoints

Each USB transaction is formed as a token packet, optional data packet and, optional handshake packet.

Data transfers consist of two or three phases:

- Token packet
- Data packet
- Handshake packet (optional)

Only control, bulk and, interrupt transfers have their own handshake phase.

Isochronous transfers do not contain a handshake phase. Data is transferred during the data packet phase. Two PID types are available for this: DATA0 and DATA1.

7.5.1 Bulk/Interrupt endpoints implementation

The USB controller has 1 to 5 bulk IN endpoints and 1 to 5 bulk OUT endpoints.

7.5.2 Bulk/Interrupt endpoints registers

Register name	Bit name	Bit description
inbulkval(x)	Inxval	IN x endpoint valid (x = endpoint number)
usbpair	Register	Endpoint pairing register
in_ien(x)	Inxien	IN x endpoint interrupt enable (x = endpoint number)
inxbuf	Buffer	Endpoint x buffer (x = endpoint number)
inxbc	Register	IN x byte count register (x = endpoint number)
inxcs(0)	inxstl	IN x endpoint stall bit (x = endpoint number)
inxcs(1)	inxbsy	IN x endpoint busy bit (x = endpoint number)
in_irq(x)	inxir	IN x endpoint interrupt request

Table 28. Bulk/Interrupt IN endpoints registers

Register name	Bit name	Bit description
outbulkval(x)	Outxval	OUT x endpoint valid (x = endpoint number)
usbpair	Register	Endpoint pairing register
out_ien(x)	Outxien	OUT x endpoint interrupt enable (x = endpoint number)
outxbuf	Buffer	Endpoint x buffer (x = endpoint number)
outxbc	Register	OUT x byte count register (x = endpoint number)
outxcs(0)	Outxstl	OUT x endpoint stall bit (x = endpoint number)
outxcs(1)	Outxbsy	OUT x endpoint busy bit (x = endpoint number)
out_irq(x)	Outxir	OUT x endpoint interrupt request

Table 29. Bulk OUT endpoints registers

7.5.3 Bulk and interrupt endpoints initialization

The MCU sets the appropriate valid bits in the in(out)bulkval register to enable bulk IN (OUT) endpoints for normal operation.

7.5.3.1 Bulk and interrupt transfers

a) IN transfers

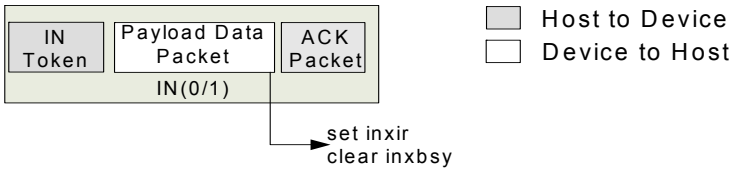


Figure 32. Bulk IN transfer

The host issues an IN token to receive bulk data. If the inxbsy bit is set, the USB controller responds by returning a data packet. If the host receives a valid data packet, it responds with an ACK handshake.

After receiving a valid ACK handshake from the host, the USB controller sets the inxir bit and clears the inxbsy bit. Setting the inxir bit generates an interrupt request for IN x endpoint (x = appropriate number of endpoint).

The MCU services the interrupt request. During a service interrupt request the MCU loads new data into the inxbuf buffer and then reloads the inxbc register with a valid number of data bytes to set the inxbsy bit. IN x endpoint is armed for the next transfer when the inxbsy bit is set.

When the inxbsy bit is not set, the USB controller returns NAK handshake for each IN token from the host. When the inxstl bit is set, the USB controller returns the STALL handshake.

Errors in IN token	Inxbsy	Inxstl	USB controller to host response
NO	1	0	Inxbc bytes data packet
NO	0	1	STALL
NO	0	0	NAK
NO	1	1	STALL
YES	-	-	No response

Table 30. The USB controller response for IN Token

b) OUT transfers

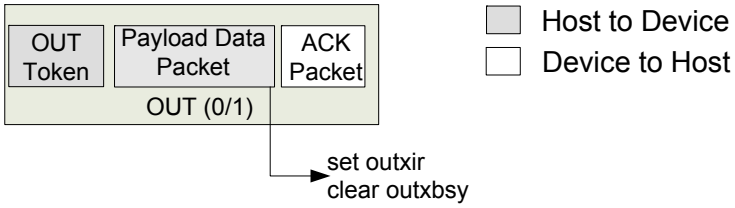


Figure 33. Bulk OUT transfer

When the host wants to transmit bulk data, it issues an OUT Token packet followed by a data packet. An ACK handshake is returned to the host and the outxir bit is set when the USB controller receives an error free OUT, data packets and, the outxbsy bit is set. Setting the outxir bit generates an interrupt request for the OUT x endpoint (x = appropriate number of endpoints).

The MCU services the OUT x interrupt request. The received data packet is available in the outxbuf buffer. After servicing an interrupt request, the MCU reloads the outxbc register with any value to set the outxbsy bit. When the outxbsy bit is set the OUT x endpoint is armed for the next OUT transfer.

A NAK handshake is returned to the host when the USB controller receives data packets and an error free OUT but the outxbsy bit is not set.

A STALL handshake is returned to the host when the USB controller receives an error free OUT and data packets and the outxstl bit is set. The USB controller does not return a handshake if any transmission error occurs during an OUT token or data phase.

Errors in OUT token or in data packet	outxbsy	outxstl	USB controller to host response
NO	0	0	NAK
NO	0	1	STALL
NO	1	0	ACK
NO	1	1	STALL
YES	-	-	No response

Table 31. The USB controller response for OUT transfers

7.5.4 Data packet synchronization

Data packet synchronization is achieved through the use of the data sequence toggle bits and the DATA0/DATA1 PIDs. The USB controller automatically toggles DATA0/DATA1 PIDs every bulk transfer.

The MCU can directly set or clear data toggle bits using the togctl register. The MCU clears the toggle bits when the host issues Clear Feature, Set Interface or, selects alternate settings.

To write a toggle bit the MCU performs the following sequence:

- Write to togctl register “000d0eee” value to select endpoint “eee” (“eee” – binary value). Endpoint direction bit “d”: “d”=’0’ – OUT endpoint; “d”=’1’ – IN endpoint.
- Clear or set toggle bit by writing to togctl register “0srd0eee” value. “sr”=’10’ – setting toggle bit; “sr”=’01’ – clearing toggle bit.

7.5.5 Endpoint pairing

To enable double buffering the MCU sets the appropriate bits in the usbpair register to '1' (see section 15.1 on page 128).

When double buffering is enabled, the MCU may access one buffer of the pair while the USB host accesses the other. When an endpoint is paired, the MCU uses only an even numbered endpoint of the pair.

For example, if the usbpair(0) bit is set, that means that the IN 2 and the IN 3 endpoints are paired. The MCU should not access in3buf data buffer, in3val bit, in3bc register, in3ir bit, in3ien bit, or in3cs registers.

7.5.5.1 Paired IN endpoint status

When both endpoint buffers of the pair are filled and armed, the inxbsy bit is set to '1' by the USB controller and the MCU does not load new data into the inxbuf buffer.

When one or both buffers of the pair are empty (unarmed), the inxbsy bit is set to '0' by the USB controller and the MCU may fill inxbuf with new data and reload the inxbc register to arm the endpoint for transmission. Clearing the inxbsy bit (write a '1') causes both of the paired endpoints to unarm. An interrupt request is generated after each data packet is correctly sent, independent of the inxbsy bit.

7.5.5.2 Paired OUT endpoint status

When the MCU pairs OUT endpoints by setting bit in the usbpair register, it also reloads twice the outxbc register to arm paired OUT endpoints.

When both endpoint buffers of the pair are empty and no data is available for the MCU, the outxbsy bit is set to '1' by the USB controller.

When one or both of the buffers contain valid data, the outxbsy bit is reset to '0' by the USB controller. Clearing the outxbsy bit (write a '1') causes both of the paired endpoints to unarm. An interrupt request is generated after each data packet is correctly received, independent of outxbsy bit or dstall. To receive an interrupt you must arm the endpoint by setting outxbc to a non-zero value.

7.6 Isochronous endpoints

Isochronous (ISO) transactions have a token and a data phase, but no handshake phase. ISO transactions do not support a handshake phase or retry capability and they do not support a data toggle synchronization mechanism.

Isochronous transmission is double buffered. An ISO FIFO swap occurs for every start of frame packet.

7.6.1 Isochronous endpoints implementation

The USB controller contains one IN endpoint and one isochronous OUT endpoint (Endpoint 8 IN/OUT).

7.6.2 Isochronous endpoints registers

Register name	Bit name	Bit description
inisoval	in8val	IN 8 endpoint valid
in8addr	register	IN 8 endpoint address register
usbpair(7)	isosend0	ISO endpoints send a zero length data packet if it is empty
usbien(1)	sofie	Start of Frame interrupt enable
in8data	register	IN 8 endpoint data register
usbirq(1)	sofir	Start of Frame interrupt request

Table 32. ISO IN endpoint registers

Register name	Bit name	Bit description
outisoval	out8val	OUT 8 endpoint valid
out8addr	register	OUT 8 endpoint address register
usbien(1)	sofie	Start of Frame interrupt enable
out8data	register	OUT 8 endpoint data register
usbirq(1)	sofir	Start of Frame interrupt request
out8bch	register	Received byte count register high
out8bcl	register	Received byte count register low
isoerr	iso8err	OUT 8 endpoint CRC error

Table 33. ISO OUT endpoint registers

7.6.3 ISO endpoints initialization

The MCU performs the following steps to enable isochronous IN (OUT) endpoints for normal operation:

- Sets the appropriate valid bits into the in(out)isoval register.
- Sets the endpoint's FIFO size by loading the start address into the in(out)8addr register.
- Sets the isosend0 bit into the usbpair register - for IN endpoints only.
- Enables the start of frame interrupt by setting the sofie bit in the usbien register.

7.6.4 ISO transfers

The MCU serves all the ISO endpoints in response to a start of frame interrupt request.

7.6.4.1 ISO IN transfers

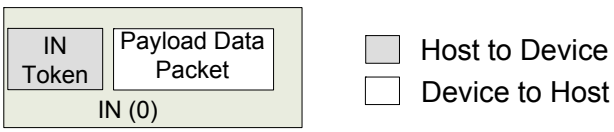


Figure 34. ISO IN transfer

The MCU loads new data into the ISO IN endpoint buffer(s) at every start of frame interrupt request. The ISO IN endpoint is accessed through the in8data register. The USB controller keeps track of the number of bytes that the MCU loads and sends loaded data during the next frame.

When the host wants to receive ISO data, it issues an IN token for a specific endpoint. If the IN buffer the host selected contains data, the USB controller responds by returning a data packet.

If the buffer is empty the USB controller behavior depends on the isosend0 bit:

- If the isosend0 bit is set, the USB controller responds with a zero byte length data packet.
- If the isosend0 bit is not set, USB controller does not respond.

7.6.4.2 ISO OUT transfers

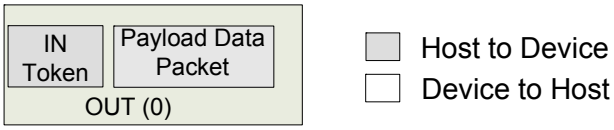


Figure 35. ISO OUT transfer

With every start of frame interrupt request the MCU reads data that was sent by the host in the previous frame. Out8bch and out8bcl registers contain the number of transferred bytes. Data is accessible through the out8data register. The USB controller sets the iso8err bit when the ISO data packet is corrupted.

7.7 Memory configuration

7.7.1 On-chip memory map

All endpoint buffers are located in a single 512 byte memory block. Bulk OUT buffers block start at address 0. You can program localization of the Bulk IN buffers using binstaddr register. If the host sends a packet which is larger than the configured buffer size, the USB controller will not NAK or STALL.

You can program start of ISO buffers using isostaddr register. Additionally, program size of the ISO buffers using isosize register.

Note: All ISO endpoints are double buffered.

Figure 36. shows on-chip memory organization. See Appendix A on page 184 for various USB memory configurations.

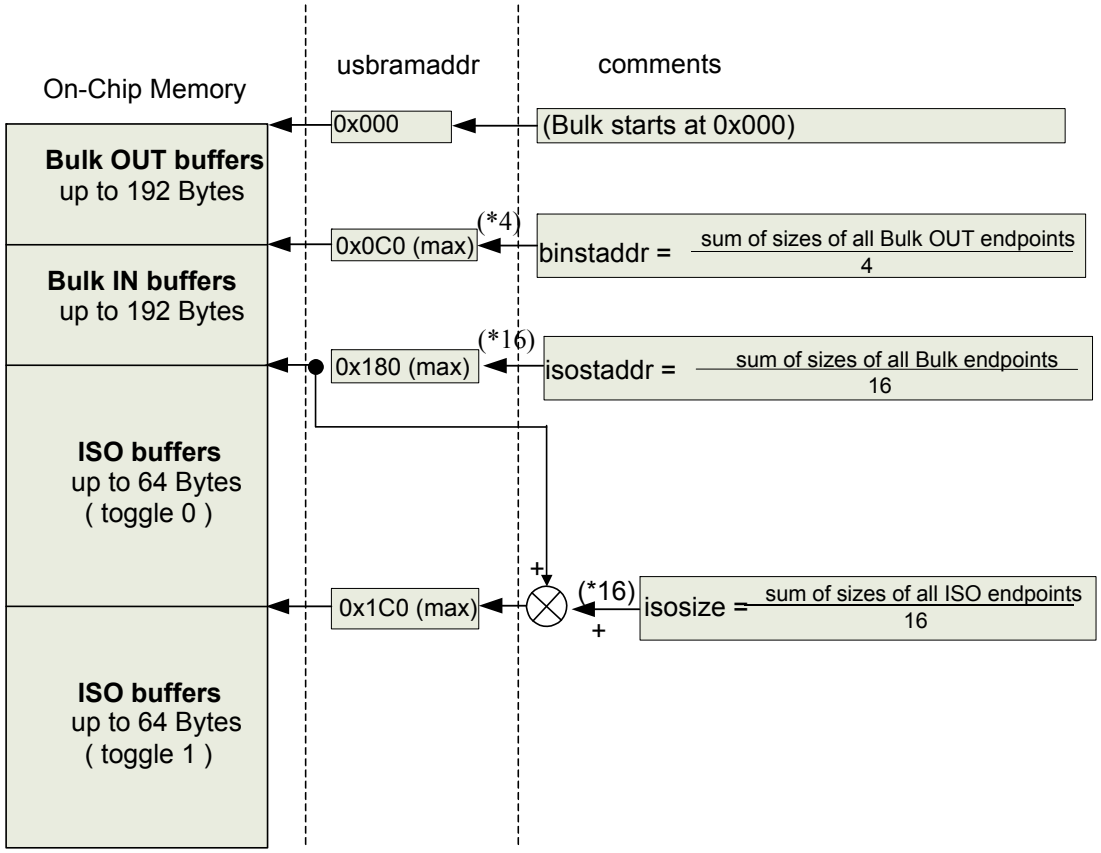


Figure 36. On-chip memory map

7.7.2 Setting ISO FIFO size

128 byte ISO buffers memory may be distributed over the two endpoint addresses: EP8 IN and EP8 OUT. The MCU initializes the endpoint FIFO sizes by setting the starting address for each FIFO. The first FIFO starting address is 0x000. The size of an isochronous endpoint FIFO is determined by subtracting consecutive values of FIFO 8 starting addresses.

Note: Only the six most significant bits can be written by the MCU (see Figure 37. on page 79).

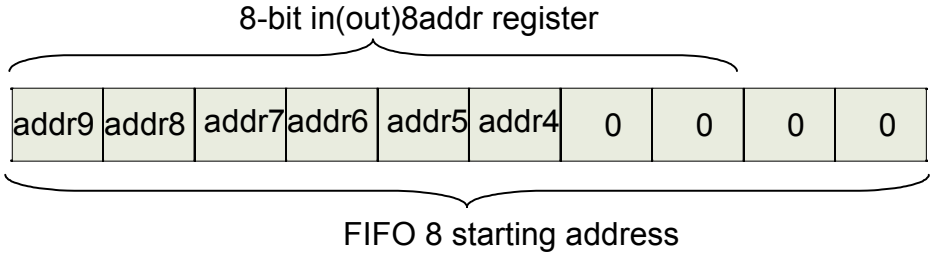


Figure 37. FIFO 8 starting address

The LSB values of the in(out)8addr register are always zero, that is, the smallest size of FIFO buffer for each ISO endpoints is 16 bytes.

7.7.3 Setting Bulk OUT size

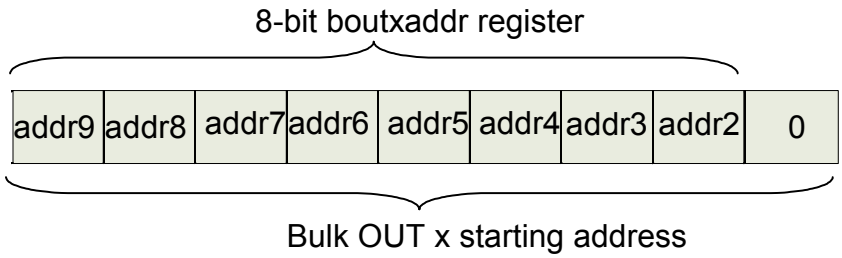


Figure 38. Bulk OUT x starting address

Bulk OUT buffers memory can be distributed over the 6 bulk OUT endpoints. Size of each Bulk OUT endpoint should be programmed using boutxaddr registers. When OUT x endpoint is not used the boutxaddr for this endpoint should be set to 0x000.

The first starting address (EP0 OUT) is 0x000. The size of a bulk OUT endpoint is determined by subtracting consecutive values of bulk OUT x starting addresses. The size of Bulk OUT buffer is a multiple of two bytes.

Here is an example initialization of the boutxaddr registers:

```
const uint8_t EP0OUTSTARTADDR = 0; // start address for EP0 OUT
bout1addr = EP0OUTSTARTADDR + (EP0OUT_SIZE/2);
bout2addr = bout1addr + (EP1OUT_SIZE/2);
bout3addr = bout2addr + (EP2OUT_SIZE/2);
bout4addr = bout3addr + (EP3OUT_SIZE/2);
bout5addr = bout4addr + (EP4OUT_SIZE/2);
binstaddr = (bout5addr + (EP5OUT_SIZE/2))/2;    // beginning of
Bulk IN buffers
```

7.7.4 Setting Bulk IN size

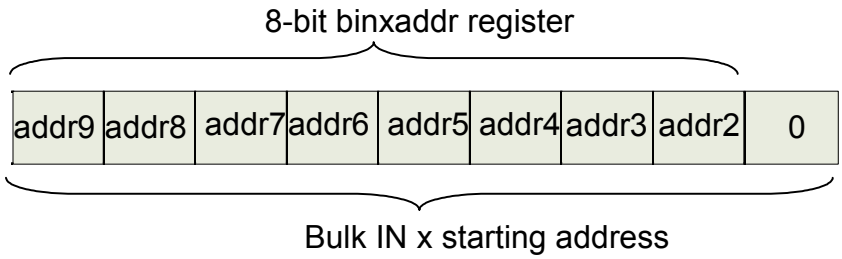


Figure 39. Bulk IN x starting address

Bulk IN buffers memory can be distributed over the 6 bulk IN endpoints. Size of each bulk IN endpoint is programmed using binxaddr registers.

When IN x endpoint does not exist (or is not used) the binxaddr for it should be set to 0x000. The first starting address (EP0 IN) is 0x000. The size of a Bulk IN endpoint is determined by subtracting consecutive values of Bulk IN x starting addresses. The size of Bulk IN buffer is a multiple of two bytes.

Here is an example initialization of the binxaddr registers:

```
const unsigned char EP0INSTARTADDR = 0; // start address for EP0 IN
bin1addr = EP0INSTARTADDR + (EP0IN_SIZE/2);
bin2addr = bin1addr + (EP1IN_SIZE/2);
bin3addr = bin2addr + (EP2IN_SIZE/2);
bin4addr = bin3addr + (EP3IN_SIZE/2);
bin5addr = bin4addr + (EP4IN_SIZE/2);
isostaddr = (bin5addr + (EP5IN_SIZE/2))/8 + binstaddr/4;  // beginning of the ISO buffers
```

7.8 The USB controller interrupts

The USB controller provides the two following interrupt signals for MCUs:

- USBWU
- USBIRQ

The USB controller generates interrupts by setting the USBWU or USBIRQ signal high and then setting it low. This interrupt request pulse is detected by the MCU as an edge triggered interrupt.

7.8.1 Wakeup interrupt request

When the USB controller is suspended by the host, it can be resumed in two ways:

- By the MCU setting the wakeup bit 6 of `USBCON` SFR register.
- By receiving a resume request from the host.

After resuming, the USB controller generates a wakeup interrupt request by setting the USBWU signal high.

7.8.2 USB interrupt request

The USB interrupt request is provided through the USBIRQ signal and includes:

- 12 bulk endpoint interrupts
- Start of frame interrupt (sofir)
- Suspend interrupt (suspir)
- USB reset interrupt (uresir)
- Setup token interrupt (sutokir)
- Setup data valid interrupt (sudavir)

Figure 40. on page 83 shows all the interrupt sources and their natural priority.

After servicing the USB controller interrupt, the MCU clears the individual interrupt request flag in the USB registers. If any other USB interrupts are pending, the act of clearing the interrupt request flag causes the USB controller to generate another pulse for the highest priority pending interrupt. If more than one interrupt is pending, each is serviced in the priority order.

The sequence of clearing the interrupt requests is important. The MCU first clears the main interrupt request flag (USBIRQ) and then each individual interrupt request in the USB controller register (usbirq).

Clearing the interrupt source immediately generates an interrupt pulse for the next pending interrupt. The interrupt may be lost when the MCU clears the main interrupt request flag after clearing the individual interrupt source.

Note: There is a difference between the interrupt USBIRQ, which is defined in Table 138., and the register usbirq described in Table 44.

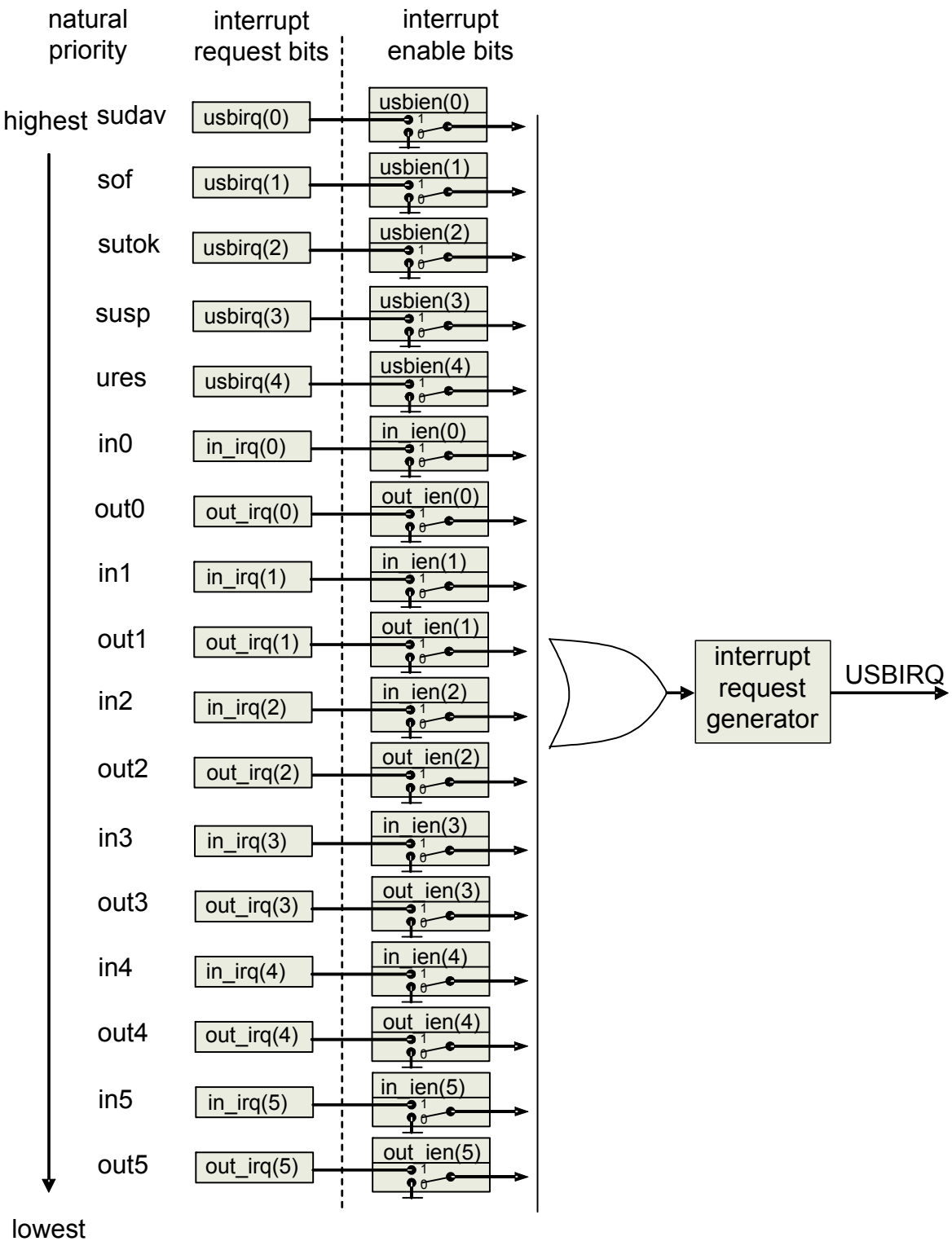


Figure 40. The USB controller interrupt sources

7.8.3 USB interrupt vectors

The USB controller prioritizes the USB interrupts if two or more occur simultaneously. The vector of the active interrupt is available in the ivec register. Table 34. shows the contents of the ivec register for the USB interrupts.

Source of interrupt	Register bit	Contents of ivec register
sudav	usbirq(0)	0x00
sof	usbirq(1)	0x04
sutok	usbirq(2)	0x08
suspend	usbirq(3)	0x0C
usbreset	usbirq(4)	0x10
ep0in	In_irq(0)	0x18
ep0out	out07irq(0)	0x1C
ep1in	In_irq(1)	0x20
ep1out	out07irq(1)	0x24
ep2in	In_irq(2)	0x28
ep2out	out07irq(2)	0x2C
ep3in	In_irq(3)	0x30
ep3out	out07irq(3)	0x34
ep4in	In_irq(4)	0x38
ep4out	out07irq(4)	0x3C
ep5in	In_irq(5)	0x40
ep5out	out07irq(5)	0x44

Table 34. Interrupt vectors

7.9 The USB controller registers

The microprocessor interfaces with the USB controller logic through the following registers and RAM buffers.

7.9.1 Bulk IN data buffers (inxbuf)

Six 32 byte bulk IN buffers are in RAM memory.

Address	Name	Function
0xC700-0xC73F	in0buf	Max 64 bytes bulk 0 IN buffer
0xC680-0xC6BF	in1buf	Max 64 bytes bulk 1 IN buffer
0xC600-0xC63F	in2buf	Max 64 bytes bulk 2 IN buffer
0xC580-0xC5BF	in3buf	Max 64 bytes bulk 3 IN buffer
0xC500-0xC53F	in4buf	Max 64 bytes bulk 4 IN buffer
0xC480-0xC4BF	in5buf	Max 64 bytes bulk 5 IN buffer

Note: The sum of all endpoints (IN+OUT+ISO) must be less or equal to 512.

Table 35.Bulk IN endpoints memory locations

7.9.2 Bulk OUT data buffers (outxbuf)

Six 32 byte bulk OUT buffers are in RAM memory.

Address	Name	Function
0xC6C0-0xC6FF	out0buf	Max 64 bytes bulk 0 OUT buffer
0xC640-0xC67F	out1buf	Max 64 bytes bulk 1 OUT buffer
0xC5C0-0xC5FF	out2buf	Max 64 bytes bulk 2 OUT buffer
0xC540-0xC57F	out3buf	Max 64 bytes bulk 3 OUT buffer
0xC4C0-0xC4FF	out4buf	Max 64 bytes bulk 4 OUT buffer
0xC440-0xC47F	out5buf	Max 64 bytes bulk 5 OUT buffer

Note: The sum of all endpoints (IN+OUT+ISO) must be less or equal to 512.

Table 36.Bulk OUT endpoints memory locations

7.9.3 Isochronous OUT endpoint data FIFO (out8dat)

Address	Name	Function
0xC760	out8data	ISO OUT endpoint 8 data FIFO register

Table 37. The out8dat register

7.9.4 Isochronous IN endpoint data FIFOs (in8dat)

Address	Name	Function
0xC768	in8data	ISO IN endpoint 8 FIFO data register

Table 38. The in8dat register

7.9.5 Isochronous data bytes counter (out8bch/out8bcl)

Address	Name	Function
0xC770	out8bch	ISO OUT endpoint 8 data counter high
0xC771	out8bcl	ISO OUT endpoint 8 data counter low

Table 39. The outxbch/bcl register

7.9.6 Isochronous transfer error register (isoerr)

Address	MSB							LSB
0xC7A0	-	-	-	-	-	-	-	iso8err

Table 40. The isoerr register

The isoerr register is updated at every Start Of Frame. The iso8err bits indicate that an error occurred during the receiving of ISO OUT 8 endpoint data packet.

Iso8err bit = 1 means that a CRC error occurred, but received data is available in the out8data register.

7.9.7 The zero byte count for ISO OUT endpoints (zbcout)

Address	MSB							LSB
0xC7A2	-	-	-	-	-	-	-	ep8

Table 41. The zbcout register

The ep8 bit is set to '1' when zero-byte ISO OUT data packet is received for OUT 8 endpoint in the previous frame.

7.9.8 Endpoints 0 to 5 IN interrupt request register (in_irq)

Address	MSB							LSB
0xC7A9	-	-	in5ir	in4ir	in3ir	in2ir	in1ir	in0ir

Table 42. The in_irq register

inxir is set to '1' when IN packet transmits and ACK receives from the host. Firmware sets inxir to '1' to clear interrupt.

7.9.9 Endpoints 0 to 5 OUT interrupt request register (out_irq)

Address	MSB							LSB
0xC7AA	-	-	out5ir	out4ir	out3ir	out2ir	out1ir	out0ir

Table 43. The out_irq register

outxir is set to '1' when OUT packet is received error free. Firmware sets outxir to '1' to clear interrupt.

7.9.10 The USB interrupt request register (usbirq)

Address	Bit	Name	Function
0xC7AB	7:5		Must be zero
	4	uresir	USB reset interrupt request 1: a USB bus reset is detected
	3	suspir	USB suspend interrupt request 1: USB SUSPEND signaling detected
	2	sutokir	SETUP token interrupt request 1: SETUP token detected
	1	sofir	Start of frame interrupt request 1: SOF packet received
	0	sudavir	SETUP data valid interrupt request 1: error free SETUP data packet received

Table 44. The usbirq bit functions

Firmware clears an interrupt request by writing '1' to the corresponding request bit.

7.9.11 Endpoint 0 to 5 IN interrupt enables (in_ien)

Address	MSB							LSB
0xC7AC	-	-	in5ien	in4ien	in3ien	in2ien	in1ien	in0ien

Table 45. The in_ien register

Firmware sets inxien to '1' to enable interrupt.

7.9.12 Endpoint 0 to 5 OUT interrupt enables (out_ien)

Address	MSB							LSB
0xC7AD	-	-	out5ien	out4ien	out3ien	out2ien	out1ien	out0ien

Table 46. The out_ien register

Firmware sets outxien to '1' to enable interrupt.

7.9.13 USB interrupt enable (usbien)

Address	Bit	Name	Function
0xC7AE	7:5	-	Must be zero
	4	uresie	USB reset interrupt enable
	3	suspie	USB suspend interrupt enable
	2	sutokie	SETUP token interrupt enable
	1	sofie	Start of frame interrupt enable
	0	sudavie	SETUP data valid interrupt enable

Table 47. The usbien register

7.9.14 Endpoint 0 control and status register (ep0cs)

Address	Bit	Name	Function
0xC7B4	7:6	-	Must be zero
	5	chgset	Setup Buffer content was changed. Chgset=1 - setup buffer was changed. Chgset=0 - setup buffer was not changed. The MCU clears the chgset bit by writing a '1' to it. The chgset bit is automatically set when USB controller receives setup data packet.
	4	dstall	Send STALL in the data stage. If dstall bit is set to '1', the USB controller sends a STALL handshake for any IN or OUT token in the data stage. When dstall is set and USB controller sends STALL in the data stage, the ep0stall is automatically set to '1' and USB controller sends STALL handshake also in the status stage. dstall is automatically cleared when a SETUP token arrives. The MCU sets this bit by writing '1' to it. The MCU should set dstall bit after last successful transaction in the data stage. When there were not excessive transactions in the data stage and the next transaction is in the correct status stage the USB controller will answer based on hsnak and ep0stall settings.
	3	outbsy	OUT0 endpoint busy bit. Outbsy is a read only bit that is automatically cleared when a SETUP token arrives. The MCU sets this bit by writing a dummy value to the out0bc register. 1: USB controller controls the OUT 0 endpoint buffer. 0: the MCU controls of the OUT 0 endpoint buffer.
	2	inbsy	IN0 endpoint busy bit. inbsy is a read only bit that is automatically cleared when a SETUP token arrives. The MCU sets this bit by reloading the in0bc register. 1: USB controller controls the IN 0 endpoint buffer. 0: the MCU controls the IN 0 endpoint buffer.
	1	hsnak	If hsnak bit is set to '1', the USB controller responds with a NAK handshake for every packet in the status stage. hsnak bit is automatically set when a SETUP token arrives. The MCU clears the hsnak bit by writing a '1' to it.
	0	ep0stall	Endpoint 0 stall. 1: the USB controller sends a STALL handshake for any IN or OUT token. This is done in the data or handshake phases of the CONTROL transfer. Ep0stall is automatically cleared when a SETUP token arrives. The MCU sets this bit by writing '1' to it.

Table 48. ep0cs register

7.9.15 Endpoint 0 to 5 IN byte count registers (inxbc)

Address	Name	Function
0xC7B5	in0bc	IN 0 endpoint byte count register
0xC7B7	in1bc	IN 1 endpoint byte count register
0xC7B9	in2bc	IN 2 endpoint byte count register
0xC7BB	in3bc	IN 3 endpoint byte count register
0xC7BD	in4bc	IN 4 endpoint byte count register
0xC7BF	in5bc	IN 5 endpoint byte count register

Table 49. Endpoint 0 to 5 IN byte count register locations

After loading the IN x endpoint buffer, the MCU writes to the inxbc register with the number of loaded bytes. Writing to the inxbc register causes the arming of IN x endpoint by setting the inxbsy bit to '1'.

When the host sends IN token for IN x endpoint and inxbsy bit is set, the USB controller responds with an inxbc size data packet.

7.9.16 Endpoint 1 to 5 IN control and status registers (inxcs)

Address	Name	Function
0xC7B6	in1cs	IN 1 endpoint control and status register
0xC7B8	in2cs	IN 2 endpoint control and status register
0xC7BA	in3cs	IN 3 endpoint control and status register
0xC7BC	in4cs	IN 4 endpoint control and status register
0xC7BE	in5cs	IN 5 endpoint control and status register

Table 50. Endpoint 1 to 5 IN control and status register locations

Bit	Symbol	Function
7:2	-	Not used.
1	inxbsy	IN x endpoint busy bit. 1: the USB controller takes control of the IN x endpoint buffer. 0: the MCU takes control of the IN x endpoint buffer. When the host sends an IN token for IN x endpoint and the inxbsy bit is set, the USB controller responds with inxbc size data packet and clears the inxbsy bit. 0: the IN x endpoint is empty and ready for loading by the MCU. 1: the MCU does not access the IN x endpoint buffer. A '1' to '0' transition of the inxbsy bit generates an interrupt request for the IN x endpoint. The MCU sets the inxbsy bit by reloading the inxbc register.
0	inxstl	IN x endpoint stall bit. 1: the USB controller returns a STALL handshake for all requests to the endpoint x.

Table 51. The inxcs register description

7.9.17 Endpoint 0 to 5 OUT byte count registers (outxbc)

Address	Name	Function
0xC7C5	out0bc	OUT 0 endpoint byte count register
0xC7C7	out1bc	OUT 1 endpoint byte count register
0xC7C9	out2bc	OUT 2 endpoint byte count register
0xC7CB	out3bc	OUT 3 endpoint byte count register
0xC7CD	out4bc	OUT 4 endpoint byte count register
0xC7CF	out5bc	OUT 5 endpoint byte count register

Table 52. Endpoint 0 to 5 OUT byte count register locations

The outxbc register contains the number of bytes sent during the last OUT transfer from the host to an OUT x endpoint. The outxbc is a read only register that is updated by the USB controller.

7.9.18 Endpoint 1 to 5 OUT control and status registers (outxcs)

Address	Name	Function
0xC7C6	out1cs	OUT 1 endpoint control and status register
0xC7C8	out2cs	OUT 2 endpoint control and status register
0xC7CA	out3cs	OUT 3 endpoint control and status register
0xC7CC	out4cs	OUT 4 endpoint control and status register
0xC7CE	out5cs	OUT 5 endpoint control and status register

Table 53. Endpoint 1 to 5 OUT control and status register locations

Bit	Symbol	Function
	-	Not used
1	outxbsy	OUT x endpoint busy bit. 1: the USB controller takes control of the OUT x endpoint buffer. 0: the MCU takes control of the OUT x endpoint buffer. When the host sends an OUT token for an OUT x endpoint and the outxbsy bit is set, the USB controller receives an OUT data packet and clears the outxbsy bit. If outxbsy='1', the OUT x endpoint is empty and ready to receive the next data packet from the host. When outxbsy='1', the MCU does not read the OUT x endpoint buffer. A '1' to '0' transition of the outxbsy bit generates an interrupt request for the OUT x endpoint. The MCU sets the outxbsy bit by reloading the outxbc register with a dummy value.
0	outxstl	OUT x endpoint stall bit. If outxstl='1', the USB controller returns a STALL handshake for all requests to the endpoint x.

Table 54. The outxcs register description

7.9.19 USB control and status register (usbcs)

Address	Bit	Name	Function
0xC7D6	7	wakesrc	Wakeup source. This bit indicates that a wakeup pin resumed the USB controller. The MCU resets this bit by writing a '1' to it.
	6	-	Not used.
	5	sofgen	Sofgen= 1 - internal SOF timer is used to generate SOF interrupt in case when SOF issued by USB host was missed. Sofgen= 0 - internal SOF timer is disabled. Default value (after reset) is '0'.
	4	-	Not used.
	3	discon	1: Disconnect the 1.5 kohm internal pull-up resistor on D+ line, 0: Normal
	2	-	Not used.
	1	forcej	Forcej should be used only in the suspend state. The MCU should set forcej bit to drive J state on the USB lines and then clear forcej and set sigrsume to drive resume-K state on the USB lines. Forcing J state between idle and K state can be done to raise the crossover voltage and eliminate any false SE0.
	0	sigrsume	Signal remote device resume. If the MCU sets this bit to '1', the USB controller sets K state on the USB.

Table 55. The usbcs bit functions

7.9.20 Data toggle control register (togctl)

Address	Bit	Name	Function
0xC7D7	7	q	Data toggle value q='1' means that data toggle for endpoint selected by ep2,ep1,ep0 and io bits is set to DATA1. q='0' means that data toggle for endpoint selected by ep2,ep1,ep0 and io bits is set to DATA0. Before reading this bit, the MCU writes the ep2, ep1, ep0 and io bits.
	6	s	Set data toggle to DATA1. Writing '1' to this bit when endpoint is selected (ep2, ep1, ep0, io bits) causes setting the data toggle to DATA1.
	5	r	Reset data toggle to DATA0. Write '1' to this bit when endpoint is selected (ep2, ep1, ep0, io bits) causes setting data toggle to DATA0.
	4	io	Select IN or OUT endpoint io='1' selects IN endpoint, i0='0' selects OUT endpoint.
	3	-	Not used.
	2	ep2	Select number of endpoint. Valid values are 0 to 5 (000 – 101).
	1	ep1	
	0	ep0	

Table 56. The togctl bit functions

7.9.21 USB frame count low (usbframel/usbframeh)

Address	Name	Function
0xC7D8	usbframel	USB frame count low
0xC7D9	usbframeh	USB frame count high

Table 57. USB frame count low (usbframel/usbframeh)

The USB controller copies the frame count into the usbframel and usbframeh registers at every SOF (Start Of Frame). These registers are read only.

Note: Frame count wraps from 3ffh to 0x000.

7.9.22 Function address register (fnaddr)

Address	Name	Function
0xC7DB	fnaddr	USB function address (1-127)

Table 58. Function address register (fnaddr)

The USB controller copies the “function address” which was sent by the host into the fnaddr register. The USB controller responds only with its assigned address. The fnaddr is a read only register.

7.9.23 USB endpoint pairing register (usbpair)

Address	Bit	Name	Function
0xC7DD	7	isosend0	ISO endpoints send zero length data packet. If the USB controller receives IN token for the isochronous endpoint and IN endpoint FIFO is empty, the USB controller response depends on the isosend0 bit. If isosend0=’1’, the USB controller sends a zero length data packet. If isosend0=’0’, the USB controller does not respond.
	6:5		Not used.
	4	pr4out	1: Pair bulk OUT 4 and bulk OUT 5 endpoints.
	3	pr2out	1: Pair bulk OUT 2 and bulk OUT 3 endpoints.
	2	pr6in	1: Pair bulk IN 6 and bulk IN 7 endpoints.
	1	pr4in	1: Pair bulk IN 4 and bulk IN 5 endpoints.
	0	pr2in	1: Pair bulk IN 2 and bulk IN 3 endpoints.

Table 59. The usbpair bit functions

7.9.24 Endpoints 0 to 5 IN valid bits (Inbulkval)

Address	MSB							LSB
0xC7DE	0	0	in5val	in4val	in3val	in2val	in1val	1

Table 60. The inbulkval register

If inxval=’1’, the IN x endpoint is active. When inxval=’0’, the IN x endpoint is inactive and the USB controller does not respond if IN x endpoint is addressed.

7.9.25 Endpoints 0 to 5 OUT valid bits (outbulkval)

Address	MSB							LSB
0xC7DF	0	0	out5val	out4val	out3val	out2val	out1val	1

Table 61. The outbulkval register

If outxval='1', the OUT x endpoint is active. When outxval='0', the OUT x endpoint is inactive and the USB controller does not respond if OUT x endpoint is addressed.

7.9.26 Isochronous IN endpoint valid bits (inisoval)

Address	MSB							LSB
0xC7E0	0	0	0	0	0	0	0	in8val

Table 62. The inisoval register

If in8val='1', the IN 8 endpoint is active. When in8val='0', the IN 8 endpoint is inactive and the USB controller does not respond if IN 8 endpoint is addressed.

7.9.27 Isochronous OUT endpoint valid bits (outisoval)

Address	MSB							LSB
0xC7E1	0	0	0	0	0	0	0	out8val

Table 63. The outisoval register

If out8val='1', the OUT 8 endpoint is active. When out8val='0', the OUT 8 endpoint is inactive and the USB controller does not respond if OUT 8 endpoint is addressed.

7.9.28 SETUP data buffer (setupbuf)

Address	MSB							LSB
0xC7E8-0xC7EF	D7	D6	D5	D4	D3	D2	D1	D0

Table 64. The setupbuf buffer

The setupbuf contains the 8 bytes of the SETUP data packet from the latest CONTROL transfer.

7.9.29 ISO OUT endpoint start address (out8addr)

Address	MSB							LSB
0xC7F0	A9	A8	A7	A6	A5	A4	0	0

Table 65. The out8addr start address

7.9.30 ISO IN endpoint start address (in8addr)

Address	MSB							LSB
0xC7F8	A9	A8	A7	A6	A5	A4	0	0

Table 66. The in8addr start address

8 Encryption/Decryption Unit

The nRF24LU1+ has dedicated HW for data encryption or decryption according to the AES (Advanced Encryption Standard) algorithm. An AES encryption/decryption consists of the transformation of a 128-bit block into an encrypted 128-bit block.

8.1 Features

The AES block supports both encryption and decryption in ECB, CBC, CFB, OFB and CTR modes using a 128-bit key and optionally a 128-bit initialization vector.

8.1.1 ECB – Electronic Code Book

ECB is the most basic AES encryption/decryption mode. In encryption E the plaintext on DI is converted to a ciphertext on DO. In decryption D the ciphertext on DI is converted to plaintext on DO. ECB must use the last expanded key to decrypt. Decryption reverses encryption operations and is identical to the encryption function.

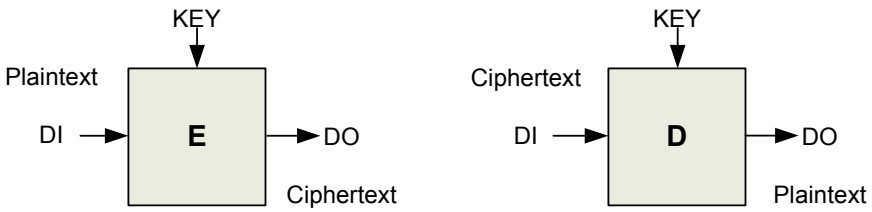


Figure 41. ECB – Electronic Code Book

8.1.2 CBC – Cipher Block Chaining

CBC adds a feedback mechanism to a block cipher. The result of the previous encryption operation is XOR'ed with incoming data. An initialization vector IV is used for the first iteration. CBC must use the last expanded key to decrypt. Decryption reverses encryption operations and is identical to the encryption function.

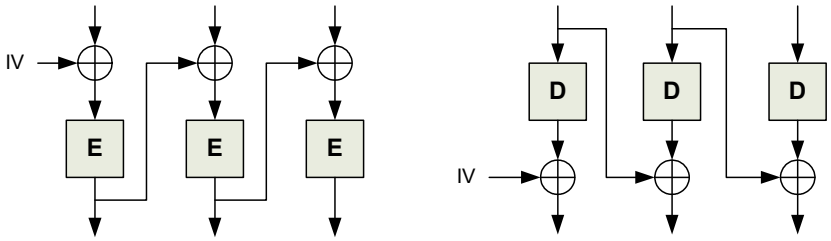


Figure 42. CBC – Cipher Block Chaining mode

8.1.3 CFB – Cipher FeedBack

In CFB the output of an encryption operation is fed back to the input of the AES block. An initialization vector IV is used for the first iteration. Input data is encrypted by XORing it with the output of the encryption module. Decryption reverses encryption operations and is identical to the encryption function.

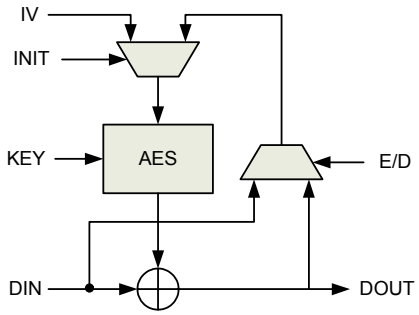


Figure 43. CFB – Cipher FeedBack mode

8.1.4 OFB – Output FeedBack mode

In OFB the output of an encryption operation is fed back to the input of the AES core. An initialization vector IV is used for the first iteration. Input data is encrypted by XORing it with the output of the encryption module. Decryption reverses encryption operations and is identical to the encryption function.

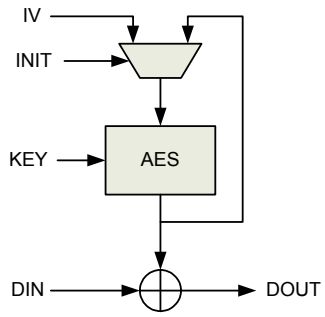


Figure 44. OFB – Output FeedBack mode

8.1.5 CTR – Counter mode

In CTR the output of a counter is the input of the AES core. When entering CTR mode, an initialization vector IV is used to initialize the counter. Input data is encrypted by XORing it with the output of the encryption module. Decryption reverses encryption operations and is identical to the encryption function.

Note: The counter cannot be reinitialized without changing mode.

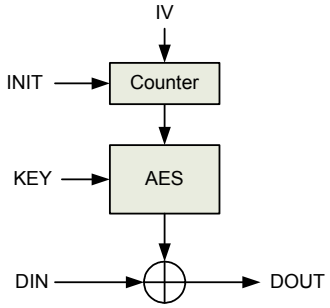


Figure 45. CTR – Counter mode

8.2 Functional description

The MCU initializes the AES block with key (AESKIN) and optionally the initialization vector (AESIV). Then the MCU loads a 128-bit block into AESD, selects mode and starts the operation with AESCS.

When the result is ready, the AESIRQ is asserted and the MCU reads out the AESD register. Alternatively, the AESCS[0] can be polled instead of using interrupts.

Address	Reset value	Bit	Name	Description
0xE8	0x00	7:5	-	Not used
		4:2	mode	Type of AES encryption/decryption 000: CBC, 001: CFB, 010: OFB, 011: CTR, 100: ECB
		1	decr	0: Encrypt, 1: Decrypt
		0	go	Set by SW to '1' to start operation. SW can poll this signal to check if AES block is busy. Automatically reset by HW when operation is completed. An encrypt or decrypt operation takes about 45 Cclk cycles.

Table 67. AESCS register

The AESKIN, AESIV and AESD are 128-bit registers. In order to update or read one of these registers, 16 consecutive write (or read) operations to a single register are required.

The order of writing (or reading) to update a register in nRF24LU1+ begins with the last (16th) operation and works back to the first.

AES data sets are normally organized in two dimensional arrays (ai,j):

a0,0	a0,1	a0,2	a0,3
a1,0	a1,1	a1,2	a1,3
a2,0	a2,1	a2,2	a2,3
a3,0	a3,1	a3,2	a3,3

Table 68. A two dimensional array

The array position to write to is decided by AESIA1 which contains two 4 bit pointers for the AESKIN register and the AESIV register. These pointers are incremented by each read/write to AESKIN or AESIV. After 16 reads/writes, the pointers wrap around to the starting value.

The relation of the byte pointer address n and the position in the AES array (ai,j) is given by the following definition:

i = n mod 4; j = [n / 4]; n = 15 - i - 4 * j

The relationship between the byte pointer address and the AES array position in nRF24LU1+ is shown in Table 69. below:

n = 15	n = 11	n = 7	n = 3
n = 14	n = 10	n = 6	n = 2
n = 13	n = 9	n = 5	n = 1
n = 12	n = 8	n = 4	n = 0

Table 69. Relation between the byte pointer address and AES data array

Address	Reset value	Bit	R/W	Description
0xF1	0x00	7:0	RW	AESKIN
0xF2	0x00	7:0	RW	AESIV
0xF3	0x00	7:0	RW	AESD

Table 70. AESKIN, AESIV and AESD registers

A partial update can be done by using the AESIA1 or AESIA2 registers (see Table 71. and Table 72.).

By setting AESIA1 a single byte or a smaller block can be updated.

Address	Reset value	Bit	Name	Description
0xF5	0x00	7:4	ia_kin	AESKIN byte pointer address
		3:0	ia_iv	AESIV byte pointer address

Table 71. AESIA1 register

The AESIA2 works like AESIA1, but only contains a 4-bit pointer for AESD.

Address	Reset value	Bit	Name	Description
0xF6	0x00	7:4	-	not used
		3:0	ia_data	AESD byte pointer address

Table 72. AESIA2 register

9 SPI master

The system features a simple single buffered SPI (Serial Peripheral Interface) master working in mode 0, that is, capture MISO in rising SCK, and changing MOSI on falling SCK.

The SPI bus (MMISO, MSCK and MMOSI) are available at the programmable digital I/O. The SPI hardware does not generate any chip select signal. Another programmable digital I/O must be used to act as chip selects for one or more external SPI devices, see Table 98. on page 117 for details.

9.1 Block diagram

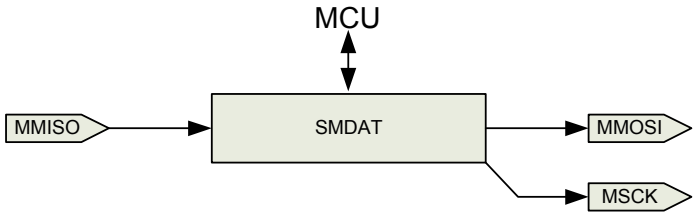


Figure 46. Master SPI block diagram

9.2 Functional description

The SPI hardware is controlled by SFR registers SMDAT and SMCTRL.

Address	Reset value	Bit	R/W	Function
0xB2	0x00		R/W	SPI data input/output

Table 73. SMDAT register

Address	Reset value	Bit	R/W	Function
0xB3	0x00	7:5	-	Not used
		4	RW	00: disable, 01: enable
		3:0	RW	Divider factor from MCU clock (Cclk) to SPI clock frequency 0000: 1/2 of Cclk frequency 0001: 1/2 of Cclk frequency 0010: 1/4 of Cclk frequency 0011: 1/8 of Cclk frequency 0100: 1/16 of Cclk frequency 0101: 1/32 of Cclk frequency 0110: 1/64 of Cclk frequency other: 1/64 of Cclk frequency

Table 74. SMCTRL register.

9.3 SPI operation

A sequence of 8 pulses is started on MSCK every time you write to the SMDAT register. Also, the 8 bits of SMDAT register are clocked out on MMOSI with MSB clocking out first. Simultaneously, 8 bits from MMISO are clocked into SMDAT register. Output data is shifted on the negative edge of MSCK, and input data is read on the positive edge of MSCK. This is illustrated in Figure 47.

When the 8 bits from MMISO are finished, MSDONE interrupt goes active, and the 8 bits from MMISO can be read from SMDAT register. The interrupt bit must be cleared, by writing to SMDAT register again, before starting another SPI transaction.

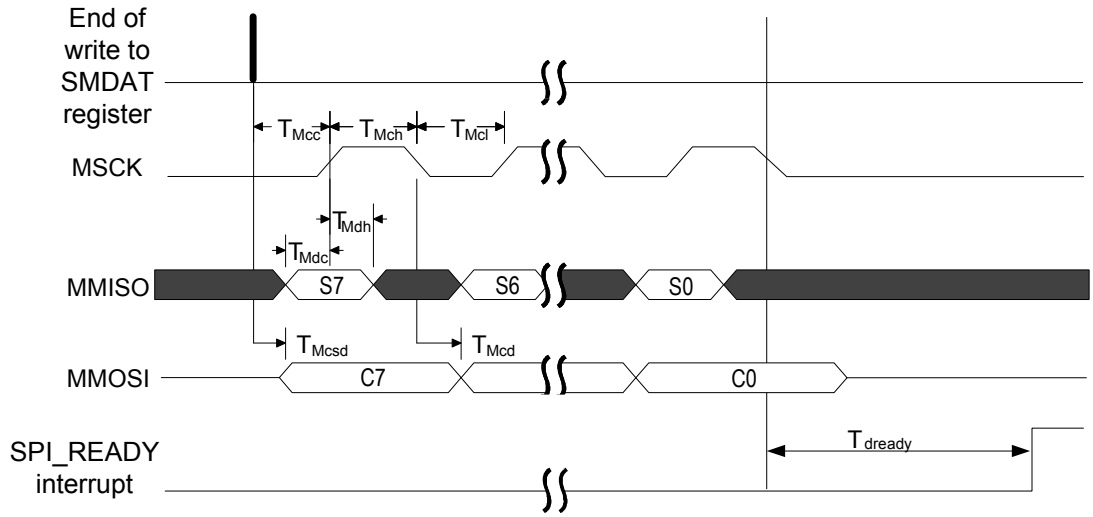


Figure 47. Master SPI timing

Value	Description
$T_{Msck} = T_{Mch} + T_{Mcl}$	SCK cycle time, as defined by SMCTRL register.
T_{Mcc}	Time from writing to SMDAT register to first MSCK pulse, T_{Msck} / 2.
T_{Mcd}	Delay from negative edge of MSCK to new MMOSI output data, may vary from -10ns to 10ns.
T_{Mdc}	MMISO setup time to positive edge of MSCK, T_{Mdc} > 45ns.
T_{Mdh}	MMISO hold time to positive edge of MSCK, T_{Mdh} > 0ns.
Tdready	Time from last MSCK pulse to MSDONE interrupt goes active. 7 MCU clock cycles.

Conditions: Output load= 10pF, MMISO rise/fall time=5ns.

Table 75. Master SPI timing values

Minimum time between two consecutive SPI transactions is: 8.5 T_{Msck} + Tdready + Tsw where Tsw is the time taken by the software to process MSDONE interrupt and write to SMDAT register.

10 SPI slave

The system features a simple single buffered SPI (Serial Programmable Interface) slave working in mode 0, that is, capturing SMOSI on rising SSCK, and changing SMISO in falling SSCK.

The slave SPI monitors the **SCSN**, **SMOSI** and **SSCK** pins, and controls the **SMISO** pin. They are available on **P0.3** to **P0.0**, see chapter 13 on page 116 for details.The SPI slave may also be used for flash programming when **PROG**=1, see section 17.7.

10.1 Block diagram

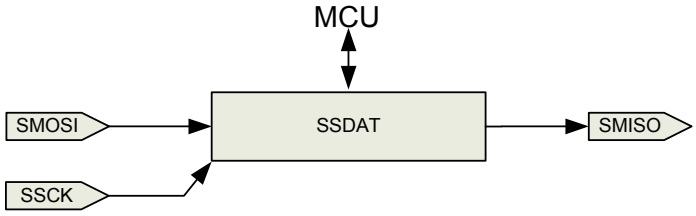


Figure 48. Slave SPI block diagram

10.2 Functional description

The following registers control the slave SPI.

Address	Reset value	Bit	R/W	Function
0xBC	0x00	7:6	-	Not used
		5	RW	1: Disable interrupt when SCSN goes high
		4	RW	1: Disable interrupt when SCSN goes low
		3	RW	1: Disable slave SPI interrupts
		2:1	-	Not used
		0	RW	1: Enable slave SPI

Table 76. SSCONF register

Address	Reset value	Bit	R/W	Function
0xBE	0x00	7:3	-	Not used
		2	R	1: Interrupt caused by positive edge of SCSN
		1	R	1: Interrupt caused by negative edge of SCSN
		0	R	1: Interrupt caused by data-byte sent or received[a]

a. If SPI is polled (no interrupts), then IRCON.2 (see chapter 22.4.4 on page 174) should be polled, since reading SSSTAT clears the status-flags.

Table 77. SSSTAT register

Address	Reset value	Bit	R/W	Function
0xBD	0x00		RW	Data register

Table 78. SSDAT register

10.3 SPI timing

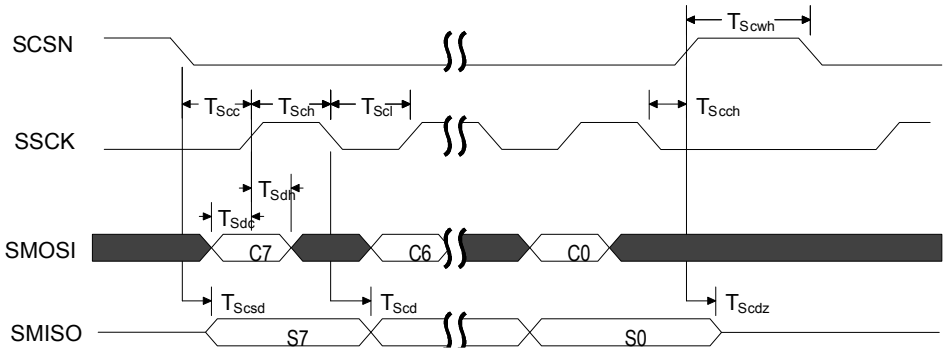


Figure 49. SPI NOP timing diagram)

Parameter	Symbol	Min.	Max	Units
SMOSI to SSCK Setup	T_{Sdc}	5		ns
SSCK to SMOSI Hold	T_{Sdh}	2		ns
SCSN to SMISO Active	T_{Scsd}		35	ns
SSCK to SMISO Valid	T_{Scd}		42	ns
SSCK Low Time	T_{Scl}	50		ns
SSCK High Time	T_{Sch}	50		ns
SSCK Frequency	F_{SSCK}	0	8	MHz
SCSN to SSCK Setup	T_{Scc}	8		ns
SSCK to SCSN Hold	T_{Scch}	2		ns
SCSN inactive time	T_{Scwh}	130		ns
SCSN to SMISO High Z	T_{Scdz}		25	ns

Conditions: SMISO load= 10pF, SSCK rise/fall time=2ns, other inputs rise/fall time=5ns.

Table 79. SPI timing parameters

11 Timer/Counters

The nRF24LU1+ contains a set of counters used for timing up important system events.

11.1 Features

nRF24LU1+ includes the following set of timers/counters:

- Three 16-bit timers/counters (Timer 0, Timer 1 and Timer 2) which can operate as either a timer with a clock rate based on the MCU clock, or as an event counter clocked by signals from the programmable digital I/O.
- In addition there is a RTC2 wakeup timer which is described in section 19.3.3 on page 161.

11.2 Block diagram

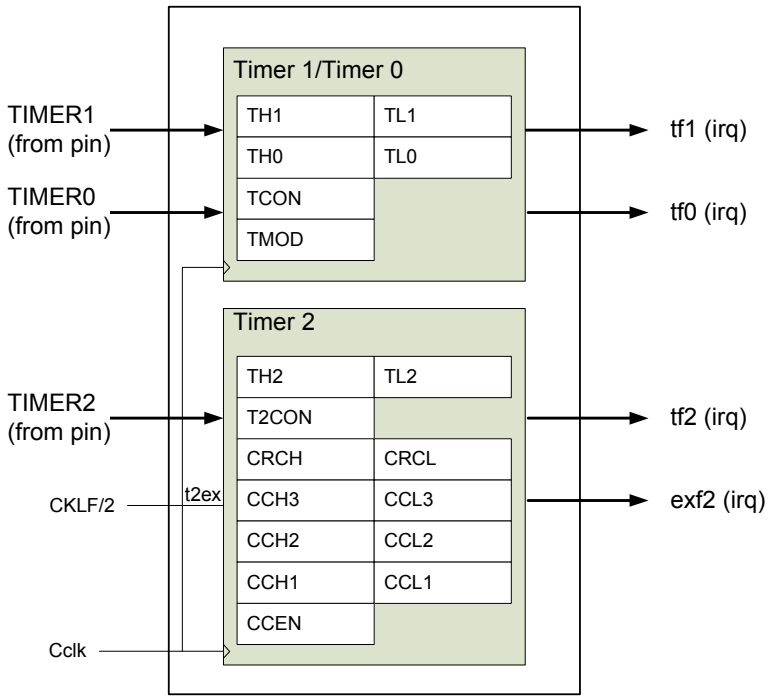


Figure 50. Block diagram of timers/counters

11.3 Functional description

11.3.1 Timer 0 and Timer 1

In timer mode, Timer 0/1 is incremented every 12 clock cycles.

In the counter mode, the Timers 1 and 0 are incremented when the falling edge is detected at the corresponding input pin T0 for Timer 0, or T1 for Timer 1.

Note: Timer input pins TO, T1, and T2 must be configured as described in section 13.2 on page 118.

Since it takes two clock cycles to recognize a 1-to-0 event, the maximum input count rate is ½ of the oscillator frequency. There are no restrictions on the duty cycle, however to ensure proper recognition of 0 or 1 state, an input should be stable for at least 1 clock cycle.

Timer 0 and Timer 1 status and control are in `TCON` and `TMOD` register. The actual 16-bit Timer 0 value is in TH0 (8 msb) and TL0 (8 lsb), while Timer 1 use TH1 and TL1.

Four operating modes can be selected for Timer 0 and Timer 1. Two Special Function Registers, `TMOD` and `TCON`, are used to select the appropriate mode.

11.3.1.1 Mode 0 and Mode 1

In mode 0, Timer 0 and Timer 1 are configured as 13-bit registers (TL0/TL1 = 5 bits, TH0/TH1 = 8 bits). The upper three bits of TL0/TL1 are unchanged and should be ignored. In mode 1 Timer 0 and Timer 1 are configured as 16-bit registers.

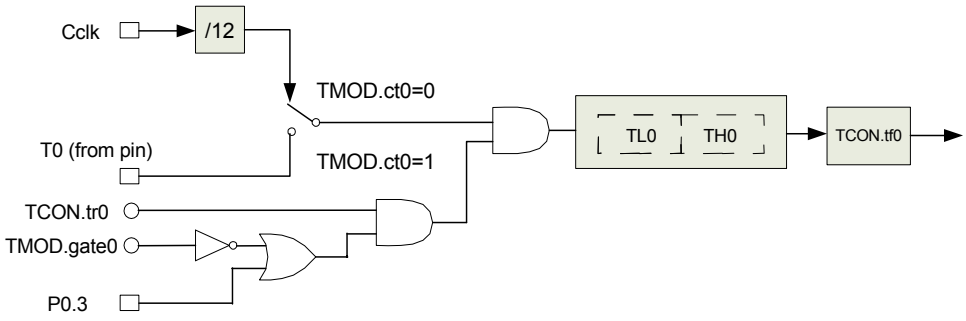


Figure 51. Timer 0 in mode 0 and 1

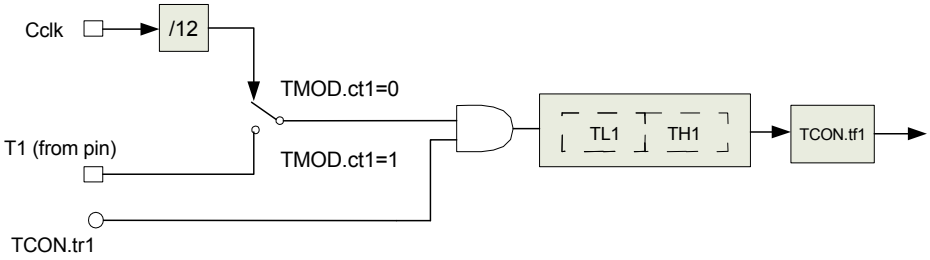


Figure 52. Timer 1 in mode 0 and 1

11.3.1.2 Mode 2

In this mode, Timer 0 and Timer 1 are configured as 8-bit registers with auto reload.

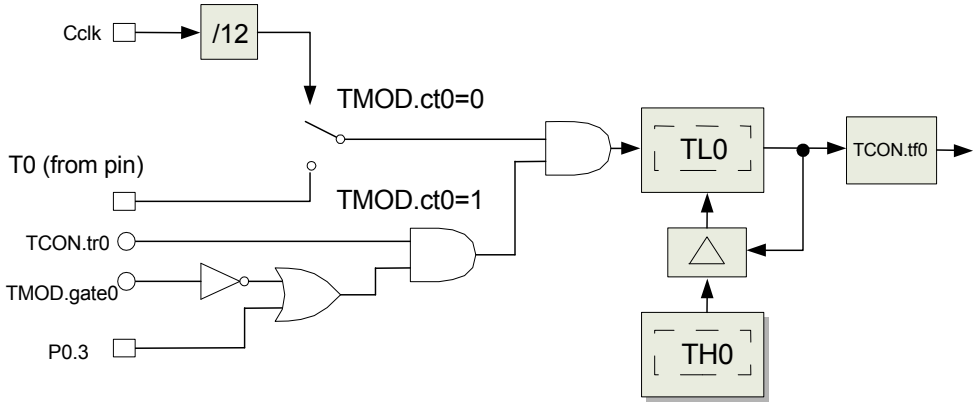


Figure 53. Timer 0 in mode 2

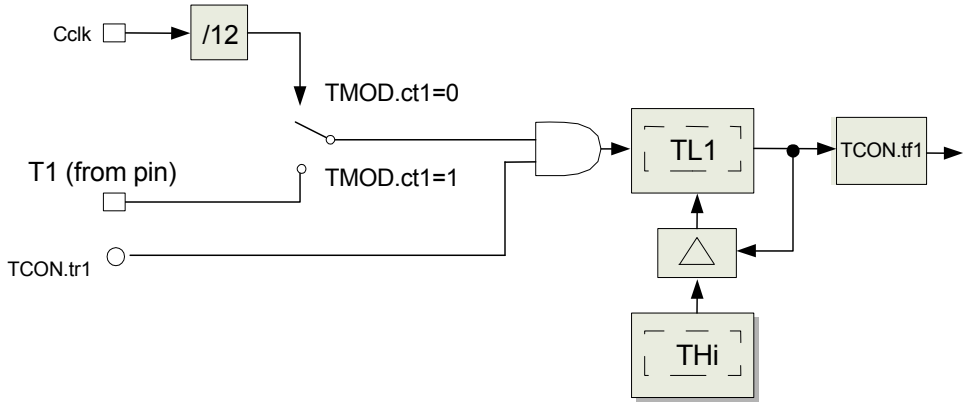


Figure 54. Timer 1 in mode 2

11.3.1.3 Mode 3

In mode 3 Timer 0 and Timer 1 are configured as one 8-bit timer/counter and one 8-bit timer, but timer 1 in this mode holds its count. When Timer 0 works in mode 3 Timer 1 can still be used in other modes by the serial port as a baud rate generator, or as an application not requiring an interrupt from Timer 1.

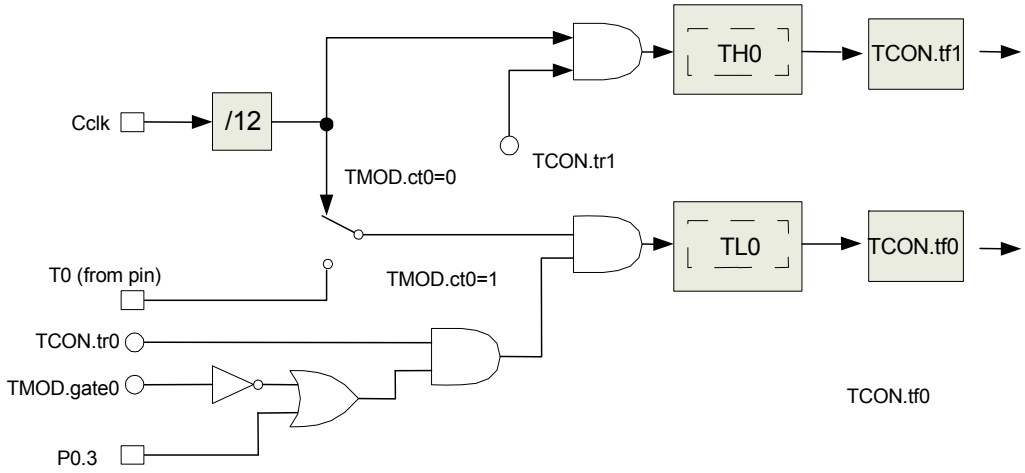


Figure 55. Timer 0 in mode 3

11.3.2 Timer 2

Timer 2 is controlled by T2CON while the value is in TH2 and TL2. Timer 2 also has four capture and one compare/reload registers which can read a value without pausing or reload a new 16-bit value when Timer 2 reaches zero, see chapter 11.4.7 on page 111 and chapter 11.4.8 on page 111.

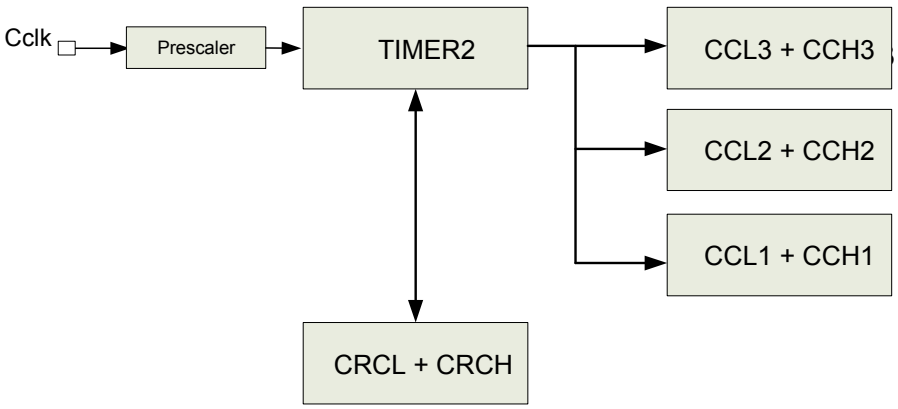


Figure 56. Timer 2 block diagram

11.3.2.1 Timer 2 description

Timer 2 can operate as a timer, event counter, or gated timer.

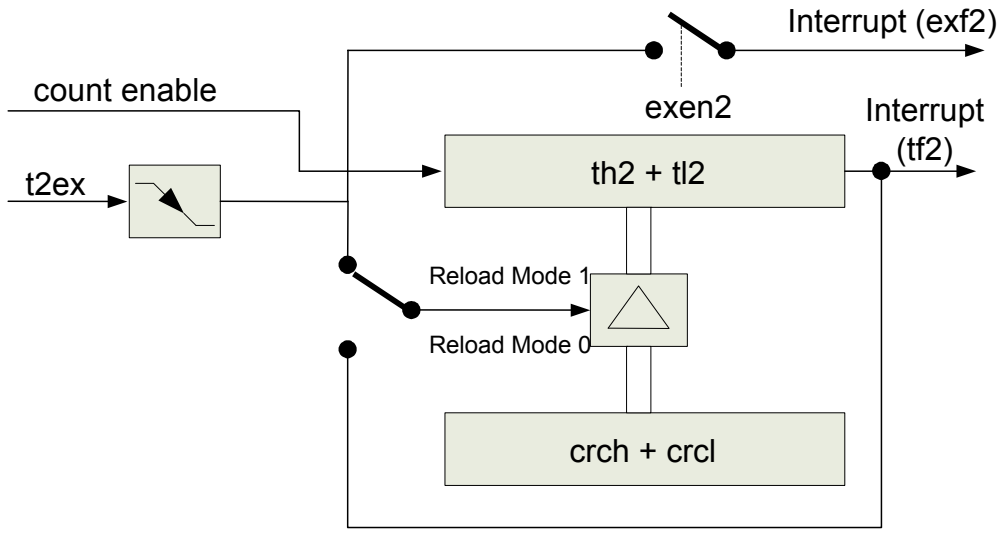


Figure 57. Timer 2 in Reload Mode

11.3.2.2 Timer mode

Timer mode is invoked by setting the t2i0=1 and t2i1=0 in the T2CON register. In this mode, the count rate is derived from the clk input.

Timer 2 is incremented every 12 or 24 clock cycles depending on the 2:1 prescaler. The prescaler mode is selected by bit t2ps of T2CON register. When t2ps=0, the timer counts up every 12 clock cycles, otherwise every 24 cycles.

11.3.2.3 Event counter mode

This mode is invoked by setting the t2i0=0 and t2i1=1 in the T2CON register.

In this mode, Timer 2 is incremented when external signal T2 (P0.5) (see section 13.2 on page 118 for more information on T2) changes its value from 1 to 0. The T2 input is sampled at every rising edge of the clock. Timer 2 is incremented in the cycle following the one in which the transition was detected. The maximum count rate is ½ of the clock frequency.

11.3.2.4 Gated timer mode

This mode is invoked by setting the t2i0=1 and t2i1=1 in the T2CON register.

In this mode, Timer 2 is incremented every 12 or 24 clock cycles (depending on T2CON t2ps flag). Additionally, it is gated by the external signal T2 (P0.5). When T2=0, Timer 2 is stopped.

11.3.2.5 Timer 2 reload

A 16-bit reload from the CRC register can be done in two modes:

- Reload Mode 0: Reload signal is generated by Timer 2 overflow (auto reload).
- Reload Mode 1: Reload signal is generated by negative transition at t2ex.

Note: t2ex is connected to an internal clock signal which is half frequency of CKLF (see section 19.3.1 on page 161.

11.4 SFR registers

11.4.1 Timer/Counter control register – TCON

TCON register reflects the current status of MCU Timer 0 and Timer 1 and it is used to control the operation of these modules.

Address	Reset value	Bit	Name	Auto clear	Description
0x88	0x00	7	tf1	Yes	Timer 1 overflow flag. Set by hardware when Timer1 overflows.
		6	tr1	No	Timer 1 Run control. If cleared, Timer 1 stops.
		5	tf0	Yes	Timer 0 overflow flag. Set by hardware when Timer 0 overflows.
		4	tr0	No	Timer 0 Run control. If cleared, Timer 0 stops.
		3	ie1	Yes	External interrupt 1 flag. Set by hardware.
		2	it1	No	External interrupt 1 type control. 1: falling edge, 0: low level
		1	ie0	Yes	External interrupt 0 flag. Set by hardware.
		0	it0	No	External interrupt 0 type control. 1: falling edge, 0: low level

Table 80. TCON register

The tf0, tf1 (timer 0 and timer 1 overflow flags), ie0 and ie1 (external interrupt 0 and 1 flags) are automatically cleared by hardware when the corresponding service routine is called.

11.4.2 Timer mode register - TMOD

TMOD register is used for configuration of Timer 0 and Timer1.

Address	Reset value	Bit	Name	Description
0x89	0x00	7	reserved	Must be 0
		6	ct1	Timer 1 counter/timer select. 1: Counter, 0: Timer
		5-4	mode1	Timer 1 mode 00 – Mode 0: 13-bit counter/timer 01 – Mode 1: 16-bit counter/timer 10 – Mode 2: 8-bit auto-reload timer 11 – Mode 3: Timer 1 stopped
		3	gate0	Timer 0 gate control
		2	ct0	Timer 0 counter/timer select. 1: Counter, 0: Timer
		1-0	mode0	Timer 0 mode 00 – Mode 0: 13-bit counter/timer 01 – Mode 1: 16-bit counter/timer 10 – Mode 2: 8-bit auto-reload timer 11 – Mode 3: two 8-bit timers/counters

Table 81. TMOD register

11.4.3 Timer0 – TH0, TL0

Address	Register name
0x8A	TL0
0x8C	TH0

Table 82. Timer 0 register (TH0:TL0)

These registers reflect the state of Timer 0. TH0 holds higher byte and TL0 holds lower byte. Timer 0 can be configured to operate as either a timer or a counter.

11.4.4 Timer1 – TH1, TL1

Address	Register name
0x8B	TL1
0x8D	TH1

Table 83. Timer 1 register (TH1:TL1)

These registers reflect the state of Timer 1. TH1 holds higher byte and TL1 holds lower byte. Timer 1 can be configured to operate as either timer or counter.

11.4.5 Timer 2 control register – T2CON

T2CON register reflects the current status of Timer 2 and is used to control the Timer 2 operation.

Address	Reset value	Bit	Name	Description
0xC8	0x00	7	t2ps	Prescaler select. 0: timer 2 is clocked with 1/12 of the Cclk frequency. 1: timer 2 is clocked with 1/24 of the Cclk frequency.
		6	i3fr	INT3 edge select. 0: falling edge, 1: rising edge
		5	i2fr	INT2 edge select: 0: falling edge, 1: rising edge
		4:3	t2r	Timer 2 reload mode. 0X – reload disabled, 10 – Mode 0, 11 – Mode 1
		2	t2cm	Timer 2 compare mode. 0: Mode 0, 1: Mode 1
		1-0	t2i	Timer 2 input select. 00: stopped, 01: f/12 or f/24, 10: falling edge of t2, 11: f/12 or f/24 gated by t2.

Table 84. T2CON register

11.4.6 Timer 2 – TH2, TL2

Address	Register name
0xCC	TL2
0xCD	TH2

Table 85. Timer 2 (TH2:TL2)

The TL2 and TH2 registers reflect the state of Timer 2. TH2 holds higher byte and TL2 holds lower byte. Timer 2 can be configured to operate in compare, capture or, reload modes.

11.4.7 Compare/Capture enable register – CCEN

The CCEN register serves as a configuration register for the Compare/Capture Unit associated with the Timer 2.

Address	Reset value	Bit	Name	Description
0xC1	0x00	7:6	coca3	compare/capture mode for CC3 register 00: compare/capture disabled 01: reserved 10: reserved 11: capture on write operation into register CCL3
		5:4	coca2	compare/capture mode for CC2 register 00: compare/capture disabled 01: reserved 10: reserved 11: capture on write operation into register CCL2
		3:2	coca1	compare/capture mode for CC1 register 00: compare/capture disabled 01: reserved 10: reserved 11: capture on write operation into register CCL1
		1:0	coca0	compare/capture mode for CRC register 00: compare/capture disabled 01: reserved 10: compare enabled 11: capture on write operation into register CRCL

Table 86. CCEN register

11.4.8 Capture registers – CC1, CC2, CC3

The Compare/Capture registers (CC1, CC2, CC3) are 16-bit registers used by the Compare/Capture Unit associated with the Timer 2. CCHn holds higher byte and CCLn holds lower byte of the CCn register.

Address	Register name
0xC2	CCL1
0xC3	CCH1
0xC4	CCL2
0xC5	CCH2
0xC6	CCL3
0xC7	CCH3

Table 87. Capture Registers - CC1, CC2 and CC3

11.4.9 Compare/Reload/Capture register – CRCH, CRCL

Address	Reset value	Register name
0xCA	0x00	CRCL
0xCB	0x00	CRCH

Table 88. Compare/Reload/Capture register - CRCH, CRCL

CRC (Compare/Reload/Capture) register is a 16-bit wide register used by the Compare/Capture Unit associated with Timer 2. CRCH holds higher byte and CRCL holds lower byte.

12 Serial Port (UART)

The MCU system is configured with one serial port that is identical in operation to the standard 8051 serial port (Serial interface 0). The two serial port signals RXD and TXD are available at the programmable digital I/O. See Chapter 13 on page 116.

The serial port (UART) derives its clock from the MCU clock; Cclk. See chapter 20.4.1 on page 167 for more information.

12.1 Features

- Synchronous mode, fixed baud rate
- 8-bit UART mode, variable baud rate
- 9-bit UART mode, variable baud rate
- 9-bit UART mode, fixed baud rate
- Additional baud rate generator

12.2 Block diagram

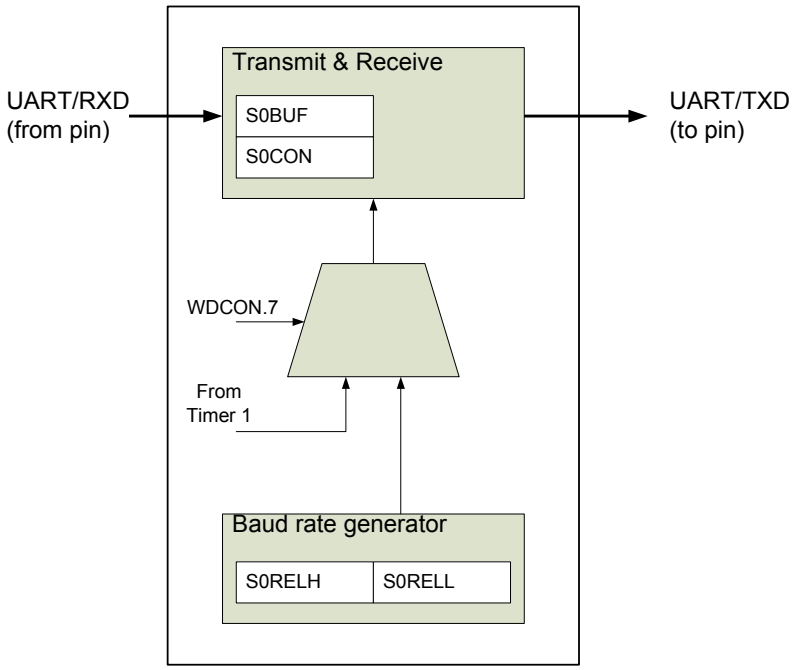


Figure 58. Block diagram of serial port

12.3 Functional description

The serial port is controlled by `S0CON`, while the actual data transferred is read or written in the `S0BUF` register. Transmission speed (baud rate) is selected using the `S0RELL`, `S0RELH` and `WDCON` registers.

12.4 SFR registers

12.4.1 Serial Port 0 control register – S0CON

The S0CON register controls the function of Serial Port 0.

Address	Reset value	Bit	Name	Description
0x98	0x00	7:6	sm0: sm1	Serial Port 0 mode select 0 0: Mode 0 – Shift register at baud rate Cclk / 12 0 1: Mode 1 – 8-bit UART. Baud rate see Figure 59. on page 114 1 0: Mode 2 – 9-bit UART at baud rate Cclk /32 or Cclk/64[a] 1 1: Mode 3 – 9 bit UART. Baud rate see Figure 59. on page 114
		5	sm20	Multiprocessor communication enable
		4	ren0	Serial reception enable: 1: Enable Serial Port 0.
		3	tb80	Transmitter bit 8. This bit is used while transmitting data through Serial Port 0 in Modes 2 and 3. The state of this bit corresponds with the state of the 9th transmitted bit (for example, parity check or multi-processor communication). It is controlled by software.
		2	rb80	Received bit 8. This bit is used while receiving data through Serial Port 0 in Modes 2 and 3. It reflects the state of the 9th received bit.
		1	ti0	Transmit interrupt flag. It indicates completion of a serial transmission at Serial Port 0. It is set by hardware at the end of bit 8 in mode 0 or at the beginning of a stop bit in other modes. It must be cleared by software.
		0	ri0	Receive interrupt flag. It is set by hardware after completion of a serial reception at Serial Port 0. It is set by hardware at the end of bit 8 in mode 0 or in the middle of a stop bit in other modes. It must be cleared by software.

a. If smod = 0 baud rate is Cclk/64, if smod = 1 then baud rate is Cclk/32.

Table 89. S0CON register

The baud rate for Mode 1 or Mode 3 is:

$$for\ bd\ (wdcon.7) = 0:$$

$$baud\ rate = \frac{2^{smod} * Cclk}{32} * (Timer\,1\ overflow\ rate)$$

$$for\ bd\ (wdcon.7) = 1:$$

$$baud\ rate = \frac{2^{smod} * Cclk}{64 * \left(2^{10} - s0rel\right)}$$

Figure 59. Equation of baud rate settings for Serial Port 0

Below is an explanation of some of the values used in Figure 59. :

Value	Definition
smod (PCON.7)	Serial Port 0 baud rate select flag
s0rel	The contents of S0REL registers (s0relh, s0rell) see chapter 12.4.3 on page 115.
bd (wdcon.7)	The MSB of WDCON register see chapter 12.4.4 on page 115

Table 90. Values of S0CON equation

12.4.2 Serial port 0 data buffer – S0BUF

Address	Reset value	Register name
0x99	0x00	S0BUF

Table 91. S0BUF register

Writing data to the SOBUF register sets data in serial output buffer and starts the transmission through Serial Port 0. Reading from the S0BUF reads data from the serial receive buffer.

12.4.3 Serial port 0 reload register – S0RELH, S0RELL

Serial Port 0 Reload register is used for Serial Port 0 baud rate generation. Only 10 bits are used, 8 bits from the S0RELL, and 2 bits from the S0RELH.

Address	Reset value	Register name
0xAA	0xD9	S0RELL
0xBA	0x03	S0RELH

Table 92. S0RELL/S0RELH register

12.4.4 Serial Port 0 baud rate select register - WDCON

The MSB of this register is used by Serial Port 0 for baud rate generation

Address	Reset value	Bit	Name	Description
0xD8	0x00	7	bd	Serial Port 0 baud rate select (in modes 1 and 3) When 1, additional internal baud rate generator is used, otherwise Timer 1 overflow is used.[a]
		6-0		Not used

Table 93. WDCON register

a. It is not recommended to use Timer1 overflow as baud generator.

13 Input/Output port (GPIO)

Six general purpose I/O lines are available on the nRF24LU1+. These can be used for general I/O with selectable direction for each bit, or these lines can be used for specialized functions.

13.1 Normal IO

When PROG=0, the GPIO pins are controlled by the registers P0ALT, P0DIR and P0EXP, when PROG=1 the GPIO pins are configured as a slave SPI port, see pins **SCSN**, **SMISO**,**SMOSI**, **SSCK** below. The P0ALT register selects between the default and the alternate functions for each of the six port pins when P0EXP=0. If P0ALT=0 then the default function is selected, port data is set with the P0 register, and pin direction is set with P0DIR register.

Address	Reset value	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bbit 0
0x80	0xFF	-	-	D5	D4	D3	D2	D1	D0

Table 94. P0 register

Address	Reset value	Bit	Description
0x94	0xFF	7:6	Not used
		5:0	0: P0.x is output, 1: P0.x is input

Table 95. P0DIR register

The P0ALT and P0EXP registers are used to select alternate or expanded functions, see Table 98. on page 117 for details.

Address	Reset value	Bit	Description
0x95	0x00	7:6	Not used
		5:0	1: Alternate function, 0: General I/O, see Table 99. on page 118

Table 96. P0ALT register

Address	Reset value	Bit	Description
0xC9	0x00	7:6	Controls P0.5
		5:4	Controls P0.4
		3:2	Not used
		1:0	Controls **P0.3-P0.0**, see Table 99. on page 118 for details

Table 97. P0EXP register

The relationship between the P0EXP and P0ALT registers is shown in Table 98. on page 117.

Pin	Normal		Expanded 1	Expanded 2	Expanded 3
	Default function	Alternate			
	P0EXP[7:6]				
	00		01	10	11
	P0ALT[5]		P0ALT[5]		
	0	1	x		
P0.5	D5	T0 (timer0 input)	T2 (timer2 input)		
	P0EXP[5:4]				
	00		01	10	11
	P0ALT[4]		P0ALT[4]		
	0	1	x		
P0.4	D4		T1 (timer1 input)		
	P0EXP[1:0]				
	00		01	10	11
	P0ALT[3-0]		P0ALT[3-0]		
	0	1	x		
P0.3	D3	INT0 (interrupt)	P0.3[a]	SCSN	
P0.2	D2	TXD (UART)	MMISO	SMISO	
P0.1	D1	RXD (UART)	MMOSI	SMOSI	
P0.0	D0	GTIMER[b]	MSCK	SSCK	

a. Configured as output, typically used for Master SPI, see chapter 9 on page 99.
b. GTIMER is an RTC output controlled by WGTIMER, see chapter 19.3.6 on page 162.

Table 98. Port functions

13.2 Expanded IO

The combined effect of the P0ALT and P0EXP register is shown in Table 99. on page 118. The content of both P0ALT and P0EXP is shown in binary. An 'X' in Table 104 means that the bit can both be '0' or '1'.

P0ALT	P0EXP	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
00000000	00000000	D5	D4	D3	D2	D1	D0
001XXXXX	00XXXXXX	T0[a]					
001XXXXX	01XXXXXX	T2[b]					
00X1XXXX	XX00XXXX						
00X1XXXX	XX01XXXX		T1[c]				
00X1XXXX	XX11XXXX						
00XX1XXX	XXXXXX00			INT0[d]			
00XX1XXX	XXXXXX01			MCSN			
00XX1XXX	XXXXXX10			SCSN			
00XX1XXX	XXXXXX11			OCITDO			
00XXX1XX	XXXXXX00				TXD		
00XXX1XX	XXXXXX01				MMISO		
00XXX1XX	XXXXXX10				SMISO		
00XXX1XX	XXXXXX11				OCITDI		
00XXXX1X	XXXXXX00					RXD	
00XXXX1X	XXXXXX01					MMOSI	
00XXXX1X	XXXXXX10					SMOSI	
00XXXX1X	XXXXXX11					OCITMS	
00XXXXX1	XXXXXX00						GTIMER[e]
00XXXXX1	XXXXXX01						MSCK
00XXXXX1	XXXXXX10						SSCK
00XXXXX1	XXXXXX11						OCITCK

a. Timer0 input
b. Timer2 input
c. Timer1 input
d. Interrupt input INT0
e. GTIMER is an RTC output controlled by WGTIMER, see chapter 19.3.6 on page 162.

Table 99. Alternative functions of Port 0

14 MCU

The nRF24LU1+ contains a fast 8-bit MCU, which executes the normal 8051 instruction set.

The architecture eliminates redundant bus states and implements parallel execution of fetch and execution phases. Most of the one-byte instructions are performed in one single cycle. The MCU uses 1 clock per cycle. This leads to a performance improvement rate of 8.0 (in terms of MIPS) with respect to legacy 8051 devices.

The original 8051 had a 12-clock architecture. A machine cycle needed 12 clocks and most instructions were either one or two machine cycles. Except for MUL and DIV instructions, the 8051 used either 12 or 24 clocks for each instruction. Each cycle in the 8051 also used two memory fetches. In many cases, the second fetch was a dummy, and extra clocks were wasted.

Table 100. shows the speed advantage compared to a legacy 8051. A speed advantage of 12 implies that the instruction is executed twelve times faster. The average speed advantage is 8.0. However, the real speed improvement seen in any system depends on the instruction mix.

Speed advantage	Number of instructions	Number of opcodes
24	1	1
12	27	83
9.6	2	2
8	16	38
6	44	89
4.8	1	2
4	18	31
3	2	9
Average: 8.0	Sum: 111	Sum: 255

Table 100. Speed advantage summary

14.1 Features

- Control Unit
  - 8-bit Instruction decoder
  - Reduced instruction cycle time (up to 12 times in respect to standard 80C51)
- Arithmetic-Logic Unit
  - 8-bit arithmetic and logical operations
  - Boolean manipulations
  - 8 x 8 bit multiplication and 8 / 8 bit division
- Multiplication-Division Unit
  - 16 x 16 bit multiplication
  - 32 / 16 bit and 16 / 16 bit division
  - 32-bit normalization
  - 32-bit L/R shifting
- Three 16-bit Timers/Counters
  - 80C51-like Timer 0 & 1
  - 80515-like Timer 2
- Compare/Capture Unit, dedicated to Timer 2
  - Four 16-bit Compare registers used for Pulse Width Modulation
  - Four external Capture inputs used for Pulse Width Measuring
  - 16-bit Reload register used for Pulse Generation

- Full Duplex Serial Interfaces
  - Serial 0 (80C51-like)
  - Synchronous mode, fixed baud rate
  - 8-bit UART mode, variable baud rate
  - 9-bit UART mode, fixed baud rate
  - 9-bit UART mode, variable baud rate
  - Baud Rate Generator
- Interrupt Controller
  - Four Priority Levels with 13 interrupt sources
- Memory interface
  - addresses up to 64 kB of External Program/Data Memory
  - Dual Data Pointer for fast data block transfer
- Hardware support for software debug

14.2 Block diagram

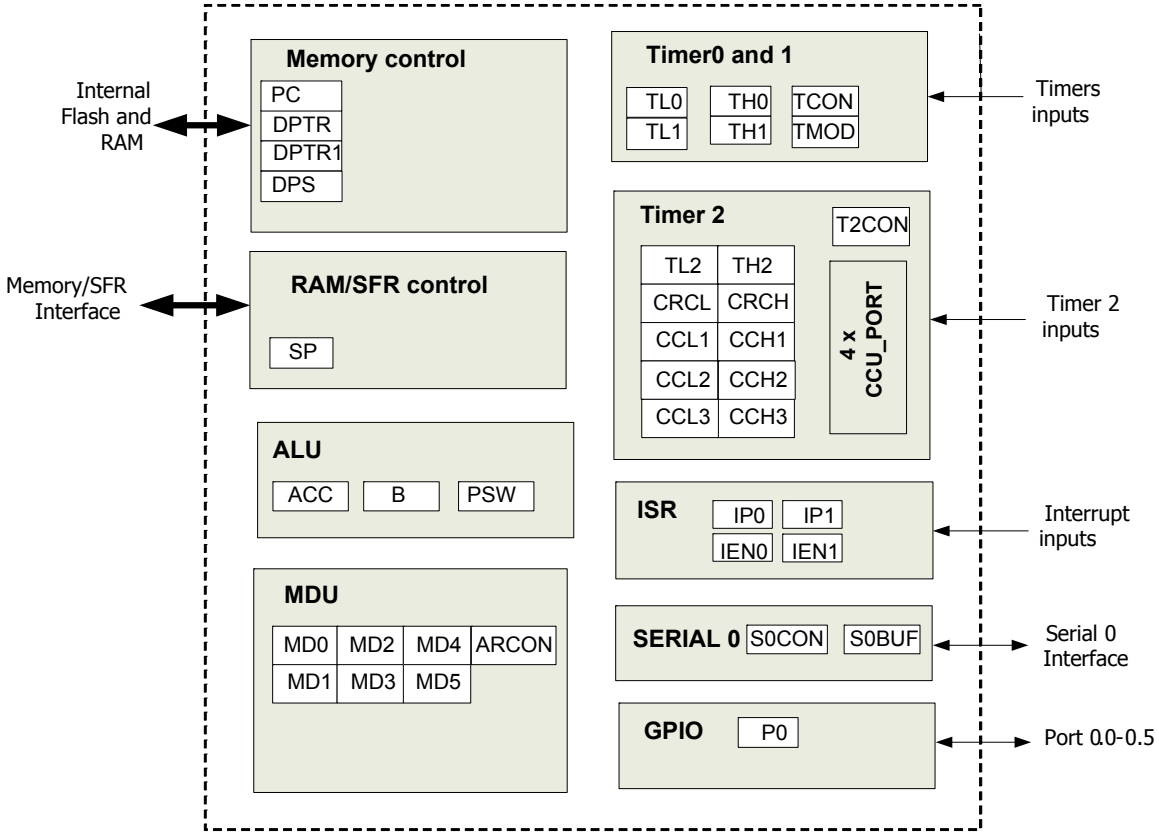


Figure 60. MCU block diagram

14.3 Arithmetic Logic Unit (ALU)

The Arithmetic Logic Unit (ALU) provides 8-bit division, 8-bit multiplication, and 8-bit addition with or without carry. The ALU also provides 8-bit subtraction with borrow and some bitwise logic operations, that is, logical AND, OR, Exclusive OR or NOT.

All operations are unsigned integer operations. Additionally, the ALU can increment or decrement 8-bit registers. For accumulator only, it can rotate left or right through carry or not, swap nibbles, clear or complement bits and perform a decimal adjustment.

The ALU is handled by three registers, which are memory mapped as special function registers. Operands for operations may come from accumulator `ACC`, register `B` or from outside of the unit. The result may be stored in accumulator `ACC` or may be driven outside of the unit. The control register, that contains flags such as carry, overflow or parity, is the `PSW` (Program Status Word) register.

The nRF24LU1+ also contains an on-chip co-processor MDU (Multiplication Division Unit). This unit enables 32-bit division, 16-bit multiplication, shift and normalize operations, see chapter 18 on page 156 for details.

14.4 Instruction set summary

All instructions are binary code compatible and perform the same functions as they do within the legacy 8051 processor. The following tables give a summary of instruction cycles of the MCU core.

Mnemonic	Description	Code	Bytes	Cycles
ADD A,Rn	Add register to accumulator	0x28-0x2F	1	1
ADD A,direct	Add directly addressed data to accumulator	0x25	2	2
ADD A,@Ri	Add indirectly addressed data to accumulator	0x26-0x27	1	2
ADD A,#data	Add immediate data to accumulator	0x24	2	2
ADDC A,Rn	Add register to accumulator with carry	0x38-0x3F	1	1
ADDC A, direct	Add directly addressed data to accumulator with carry	0x35	2	2
ADDC A,@Ri	Add indirectly addressed data to accumulator with carry	0x36-0x37	1	2
ADDC A,#data	Add immediate data to accumulator with carry	0x34	2	2
SUBB A,Rn	Subtract register from accumulator with borrow	0x98-0x9F	1	1
SUBB A, direct	Subtract directly addressed data from accumulator with borrow	0x95	2	2
SUBB A, @Ri	Subtract indirectly addressed data from accumulator with borrow	0x96-0x97	1	2
SUBB A, #data	Subtract immediate data from accumulator with borrow	0x94	2	2
INC A	Increment accumulator	0x04	1	1
INC Rn	Increment register	0x08-0x0F	1	2
INC direct	Increment directly addressed location	0x05	2	3
INC @Ri	Increment indirectly addressed location	0x06-0x07	1	3
INC DPTR	Increment data pointer	0xA3	1	1
DEC A	Decrement accumulator	0x14	1	1
DEC Rn	Decrement register	0x18-0x1F	1	2
DEC direct	Decrement directly addressed location	0x15	2	3
DEC @Ri	Decrement indirectly addressed location	0x16-0x17	1	3
MUL AB	Multiply A and B	0xA4	1	5
DIV	Divide A by B	0x84	1	5
DA A	Decimal adjust accumulator	0xD4	1	1

Table 101. Arithmetic operations

Mnemonic	Description	Code	Bytes	Cycles
ANL A, Rn	AND register to accumulator	0x58-0x5F	1	1
ANL A,direct	AND directly addressed data to accumulator	0x55	2	2
ANL A,@Ri	AND indirectly addressed data to accumulator	0x56-0x57	1	2
ANL A,#data	AND immediate data to accumulator	0x54	2	2
ANL direct,A	AND accumulator to directly addressed location	0x52	2	3
ANL direct,#data	AND immediate data to directly addressed location	0x53	3	4
ORL A,Rn	OR register to accumulator	0x48-0x4F	1	1
ORL A,direct	OR directly addressed data to accumulator	0x45	2	2
ORL A,@Ri	OR indirectly addressed data to accumulator	0x46-0x47	1	2
ORL A,#data	OR immediate data to accumulator	0x44	2	2
ORL direct,A	OR accumulator to directly addressed location	0x42	2	3
ORL direct,#data	OR immediate data to directly addressed location	0x43	3	4
XRL A,Rn	Exclusive OR register to accumulator	0x68-0x6F	1	1
XRL A, direct	Exclusive OR indirectly addressed data to accumulator	0x66-0x67	2	2
XRL A,@Ri	Exclusive OR indirectly addressed data to accumulator	0x66-0x67	2	2
XRL A,#data	Exclusive OR immediate data to accumulator	0x64	2	2
XRL direct,A	Exclusive OR accumulator to directly addressed location	0x62	2	3
XRL direct,#data	Exclusive OR immediate data to directly addressed location	0x63	3	4
CLR A	Clear accumulator	0xE4	1	1
CPL A	Complement accumulator	0xF4	1	1
RL A	Rotate accumulator left	0x23	1	1
RLC A	Rotate accumulator left through carry	0x33	1	1
RR A	Rotate accumulator right	0x03	1	1
RRC A	Rotate accumulator right through carry	0x13	1	1
SWAP A	Swap nibbles within the accumulator	0xC4	1	1

Table 102. Logic operations

Mnemonic	Description	Code	Bytes	Cycles
MOV A,Rn	Move register to accumulator	0xE8-0xEF	1	1
MOV A,direct	Move directly addressed data to accumulator	0xE5	2	2
MOV A,@Ri	Move indirectly addressed data to accumulator	0xE6-0xE7	1	2
MOV A,#data	Move immediate data to accumulator	0x74	2	2
MOV Rn,A	Move accumulator to register	0xF8-0xFF	1	2
MOV Rn,direct	Move directly addressed data to register	0xA8-0xAF	2	4
MOV Rn,#data	Move immediate data to register	0x78-0x7F	2	2
MOV direct,A	Move accumulator to direct	0xF5	2	3
MOV direct,Rn	Move register to direct	0x88-0x8F	2	3
MOV directl,direct2	Move directly addressed data to directly addressed location	0x85	3	4
MOV direct,@Ri	Move indirectly addressed data to directly addressed location	0x86-0x87	2	4

Mnemonic	Description	Code	Bytes	Cycles
MOV direct,#data	Move immediate data to directly addressed location	0x75	3	3
MOV @Ri,A	Move accumulator to indirectly addressed location	0xF6-0xF7	1	3
MOV @Ri,direct	Move directly addressed data to indirectly addressed location	0xA6-0xA7	2	5
MOV @Ri,#data	Move immediate data to indirectly addressed location	0x76-0x77	2	3
MOV DPTR,#datal6	Load data pointer with a 16-bit immediate	0x90	3	3
MOVC A,@A+DPTR	Load accumulator with a code byte relative to DPTR	0x93	1	3
MOVC A,@A+PC	Load accumulator with a code byte relative to PC	0x83	1	3
MOVX A,@Ri	Move[a] external RAM (8-bit addr) to accumulator	0xE2-0xE3	1	4
MOVX A,@DPTR	Move[a] external RAM (16-bit addr) to accumulator	0xE0	1	4
MOVX @Ri,A	Move[a] accumulator to external RAM (8-bit addr)	0xF2-0xF3	1	5
MOVX @DPTR,A	Move[a] accumulator to external RAM (16-bit addr)	0xF0	1	5
PUSH direct	Push directly addressed data onto stack	0xC0	2	4
POP direct	Pop directly addressed location from stack	0xD0	2	3
XCH A,Rn	Exchange register with accumulator	0xC8-0xCF	1	2
XCH A,direct	Exchange directly addressed location with accumulator	0xC5	2	3
XCH A,@Ri	Exchange indirect RAM with accumulator	0xC6-0xC7	1	3
XCHD A,@Ri	Exchange low-order nibbles of indirect and accumulator	0xD6-0xD7	1	3

a. The MOVX instructions perform one of two actions depending on the state of pmw bit (pcon.4).

Table 103. Data transfer operations

Mnemonic	Description	Code	Bytes	Cycles
ACALL addr11	Absolute subroutine call	xxx10001b	2	6
LCALL addr16	Long subroutine call	0x12	3	6
RET	Return from subroutine	0x22	1	4
RETI	Return from interrupt	0x32	1	4
AJMP addr11	Absolute jump	xxx00001b	2	3
LJMP addrl6	Long jump	0x02	3	4
SJMP rel	Short jump (relative address)	0x80	2	3
JMP @A+DPTR	Jump indirect relative to the DPTR	0x73	1	2
JZ rel	Jump if accumulator is zero	0x60	2	3
JNZ rel	Jump if accumulator is not zero	0x70	2	3
JC rel	Jump if carry flag is set	0x40	2	3
JNC rel	Jump if carry flag is not set	0x50	2	3
JB bit, rel	Jump if directly addressed bit is set	0x20	3	4
JNB bit, rel	Jump if directly addressed bit is not set	0x30	3	4

Mnemonic	Description	Code	Bytes	Cycles
JBC bit, rel	Jump if directly addressed bit is set and clear bit	0x10	3	4
CJNE A, direct, rel	Compare directly addressed data to accumulator and jump if not equal	0xB5	3	4
CJNE A,#data,rel	Compare immediate data to accumulator and jump if not equal	0xB4	3	4
CJNE Rn, #data, rel	Compare immediate data to register and jump if not equal	0xB8-0xBF	3	4
CJNE @Ri, #data, rel	Compare immediate data to indirect addressed value and jump if not equal	0xB6-B7	3	4
DJNZ Rn, rel	Decrement register and jump if not zero	0xD8-DF	2	3
DJNZ direct, rel	Decrement directly addressed location and jump if not zero	0xD5	3	4
NOP	No operation	0x00	1	1

Table 104. Program branches

Mnemonic	Description	Code	Bytes	Cycles
CLR C	Clear carry flag	0xC3	1	1
CLR bit	Clear directly addressed bit	0xC2	2	3
SETB C	Set carry flag	0xD3	1	1
SETB bit	Set directly addressed bit	0xD2	2	3
CPL C	Complement carry flag	0xB3	1	1
CPL bit	Complement directly addressed bit	0xB2	2	3
ANL C,bit	AND directly addressed bit to carry flag	0x82	2	2
ANL C,/bit	AND complement of directly addressed bit to carry	0xB0	2	2
ORL C,bit	OR directly addressed bit to carry flag	0x72	2	2
ORL C,/bit	OR complement of directly addressed bit to carry	0xA0	2	2
MOV C,bit	Move directly addressed bit to carry flag	0xA2	2	2
MOV bit,C	Move carry flag to directly addressed bit	0x92	2	3

Table 105. Boolean manipulation

14.5 Opcode map

Opcode	Mnemonic	Opcode	Mnemonic	Opcode	Mnemonic
00H	NOP	56H	ANL A,@R0	ACH	MOV R4,direct
01H	AJMP addr11	57H	ANL A,@R1	ADH	MOV R5,direct
02H	JUMP addrl6	58H	ANL A,R0	AFH	MOV R6,direct
03H	RRA	59H	ANL A,R1	AFH	MOV R7,direct
04H	INCA	5AH	ANL A,R2	B0H	ANL C,/bit
05H	INC direct	5BH	ANL A,R3	B1H	ACALL addr11
06H	INC @R0	5CH	ANL A,R4	B2H	CPL bit
07H	INC @R1	5DH	ANL A,R5	B3H	CPLC
08H	INC R0	5EH	ANL A,R6	B4H	CJNE A,#data,rel
09H	INC R1	5FH	ANL A,R7	B5H	CJNE A, direct, rel
0AH	INC R2	60H	JZ rel	B6H	CJNE @R0,#data,rel
0BH	INC R3	61H	AJMP addr11	B7H	CJNE @R1, #data,rel
0CH	INC R4	62H	XRL direct, A	B8H	CJNE R0, #data,rel
0DH	INC R5	63H	XRL direct, #data	B9H	CJNE R1,#data,rel
0EH	INC R6	64H	XRL A, #data	BAH	CJNE R2,#data,rel
0FH	INC R7	65H	XRL A,direct	BBH	CJNE R3,#data,rel
10H	JBC bit, rel	66H	XRLA,@R0	BCH	CJNE R4,#data,rel
11H	ACALL addr11	67H	XRL A,@R1	BDH	CJNE R5,#data,rel
12H	LCALL add r16	68H	XRL A,R0	BEH	CJNE R6,#data,rel
13H	RRC A	69H	XRL A,R1	BFH	CJNE R7,#data,rel
14H	DEC A	6AH	XRL A,R2	C0H	PUSH direct
15H	DEC direct	6BH	XRL A,R3	C1H	AJMP addr11
16H	DEC @R0	6CH	XRL A,R4	C2H	CLR bit
17H	DEC @R1	6DH	XRL A,R5	C3H	CLR C
18H	DEC R0	6EH	XRL A,R6	C4H	SWAP A
19H	DEC R1	6FH	XRL A,R7	C5H	XCH A, direct
1AH	DEC R2	70H	JNZ rel	C6H	XCH A,@R0
1BH	DECR3	71H	ACALL addr11	C7H	XCH A,@R1
1CH	DECR4	72H	ORL C, bit	C8H	XCH A,R0
1DH	DECR5	73H	JMP @A+DPTR	C9H	XCH A,R1
1EH	DECR6	74H	MOV A, #data	CAH	XCH A,R2
1FH	DECR7	75H	MOV direct, #data	CBH	XCHA,R3
20H	JB bit, rel	76H	MOV @R0,#data	CCH	XCH A,R4
21H	AJMP addr11	77H	MOV @R1, #data	CDH	XCH A,R5
22H	RET	78H	MOV R0, #data	CEH	XCH A,R6
23H	RL A	79H	MOV R1, #data	CFH	XCHA,R7
24H	ADD A, #data	7AH	MOV R2, #data	D0H	POP direct
25H	ADD A, direct	7BH	MOV R3, #data	D1H	ACALL addr11
26H	ADD A,@R0	7CH	MOV R4, #data	D2H	SETB bit
27H	ADD A,@R1	7DH	MOV R5, #data	D3H	SETB C
28H	ADD A,R0	7EH	MOV R6, #data	D4H	DAA
29H	ADD A,R1	7FH	MOV R7, #data	D5H	DJNZ direct, rel
2AH	ADD A,R2	80H	SJMP rel	D6H	XCHDA,@R0
2BH	ADD A,R3	81H	AJMP addr11	D7H	XCHD A,@R1
2CH	ADD A,R4	82H	ANL C, bit	D8H	DJNZ R0,rel
2DH	ADD A,R5	83H	MOVC A,@A+PC	D9H	DJNZ R1,rel
2EH	ADD A,R6	84H	DIV AB	DAH	DJNZ R2,rel
2FH	ADD A,R7	85H	MOV direct, direct	DBH	DJNZ R3,rel
30H	JNB bit, rel	86H	MOV direct,@R0	DCH	DJNZ R4,rel
31H	ACALL addr11	87H	MOV direct,@R1	DDH	DJNZ R5,rel

Opcode	Mnemonic	Opcode	Mnemonic	Opcode	Mnemonic
32H	RETI	88H	MOV direct,R0	DFH	DJNZ R6,rel
33H	RLC A	89H	MOV direct,R1	DFH	DJNZ R7,rel
34H	ADDC A,#data	8AH	MOV direct,R2	F0H	MOVX A,@DPTR
35H	ADDC A, direct	8BH	MOV direct,R3	F1H	AJMP addr11
36H	ADDC A,@R0	8CH	MOV direct,R4	E2H	MOVX A,@R0
37H	ADDC A,@R1	8DH	MOV direct, R5	F3H	MOVX A,@R1
38H	ADDC A,R0	8EH	MOV direct,R6	E4H	CLR A
39H	ADDC A,R1	8FH	MOV direct,R7	F5H	MOVA, direct
3AH	ADDC A,R2	90H	MOV DPTR, #datal6	E6H	MOVA,@R0
3BH	ADDC A,R3	91H	ACALL addr11	F7H	MOV A,@R1
3CH	ADDC A,R4	92H	MOV bit, C	E8H	MOV A,R0
3DH	ADDC A,R5	93H	MOVCA,@A+DPTR	F9H	MOV A,R1
3EH	ADDC A,R6	94H	SUBB A, #data	EAH	MOV A,R2
3FH	ADDC A,R7	95H	SUBB A, direct	FRH	MOV A,R3
40H	JC rel	96H	SUBB A,@R0	ECH	MOV A,R4
41H	AJMP addr11	97H	SUBB A,@R1	FDH	MOV A,R5
42H	ORL direct, A	98H	SUBB A, R0	EEH	MOV A,R6
43H	ORL direct, #data	99H	SUBB A,R1	EFH	MOV A,R7
44H	ORL A, #data	9AH	SUBB A,R2	F0H	MOVX @DPTR,A
45H	ORL A, direct	9BH	SUBB A,R3	F1H	ACALL addr11
46H	ORL A,@R0	9CH	SUBB A,R4	F2H	MOVX @R0,A
47H	ORL A,@R1	9DH	SUBB A,R5	F3H	MOVX @R1,A
48H	ORL A,R0	9EH	SUBB A,R6	F4H	CPL A
49H	ORL A,R1	9FH	SUBB A,R7	F5H	MOV direct, A
4AH	ORL A,R2	A0H	ORL C,/bit	F6H	MOV @R0,A
4BH	ORLA,R3	A1H	AJMP addr11	F7H	MOV @R1,A
4CH	ORL A,R4	A2H	MOV C, bit	F8H	MOV R0,A
4DH	ORL A,R5	A3H	INC DPTR	F9H	MOV R1,A
4EH	ORL A,R6	A4H	MUL AB	FAH	MOV R2,A
4FH	ORLA,R7	A5H	Debug	FBH	MOV R3,A
50H	JNC rel	A6H	MOV @R0,direct	FCH	MOV R4,A
51H	ACALL addr11	A7H	MOV @R1,direct	FDH	MOV R5,A
52H	ANL direct, A	A8H	MOV R0,direct	FEH	MOV R6,A
53H	ANL direct, #data	A9H	MOV R1,direct	FFH	MOV R7,A
54H	ANL A, #data	AAH	MOV R2,direct		
55H	ANL A, direct	ABH	MOV R3,direct		

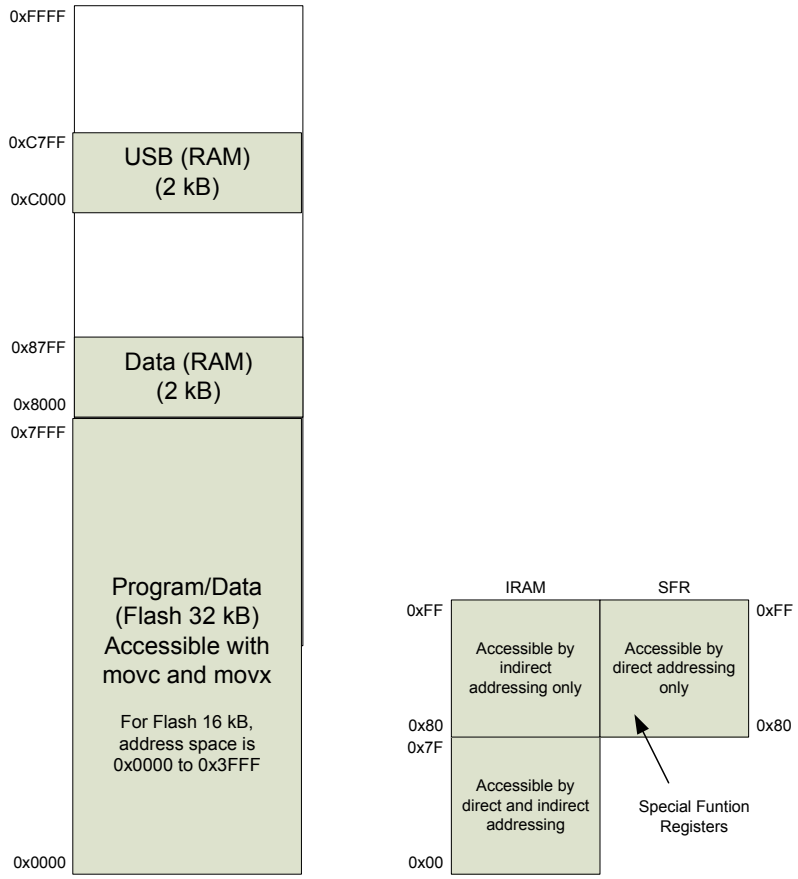
Table 106. Opcode map

15 Memory and I/O organization

The MCU has a 64 kbytes address space for code and data, an area of 256 byte for internal data (IRAM), and an area of 128 byte for Special Function Registers (SFR).

The nRF24LU1+ has 16 or 32 kbytes of flash program memory, 2 kbytes of SRAM data memory and a dedicated internal RAM of 256 byte for MCU internal data, see Figure 61. To allow erase and write flash operations, the MCU must run the sequence described in section 17.5.1.

In addition, an area of 2 kbytes is reserved for the USB buffer RAM and the USB configuration registers.



Note: In a program running in a protected flash area, movc may not be used to access addresses 0x00 to 0x03.

Figure 61. nRF24LU1+ memory map

15.1 Special function registers

15.1.1 Special function registers locations

The map of Special Function Registers is shown in Table 107.

Address	X000	X001	X010	X011	X100	X101	X110	X111
0xF8-0xFF	FSR	FPCR	FCR					
0xF0-0xF7	B	AESKIN	AESIV	AESD		AESIA1	AESIA2	
0xE8-0xEF	AESCS	MD0	MD1	MD2	MD3	MD4	MD5	ARCON
0xE0-0xE7	ACC					RFDAT	RFCTL	
0xD8-0xDF	WDCON	USBSLP						
0xD0-0xD7	PSW							
0xC8-0xCF	T2CON	P0EXP	CRCL	CRCH	TL2	TH2		
0xC0-0xC7	IRCON	CCEN	CCL1	4CCH1	CCL2	CCH2	CCL3	CCH3
0xB8-0xBF	IEN1	IP1	S0RELH		SSCONF	SSDAT	SSSTAT	
0xB0-0xB7		RSTRES	SMDAT	SMCTRL		TICKDV		
0xA8-0xAF	IEN0	IP0	S0RELL	REGXH	REGXL	REGXC		
0xA0-0xA7	USBCON			CLKCTL	PWRDWN	WUCONF	INTEXP	
0x98-0x9F	S0CON	S0BUF						
0x90-0x97	RFCON		DPS		P0DIR	P0ALT		
0x88-0x8F	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	
0x80-0x87	P0	SP	DPL	DPH	DPL1	DPH1		PCON

Table 107. Special Function Registers locations

Note: Undefined locations are reserved and must not be read or written.

The registers in the X000 column in Table 107. above are both byte and bit addressable. The other registers are only byte addressable.

15.1.2 Special function registers reset values

Register name	Address	Reset value	More information	Description
ACC	0xE0	0x00	Section 15.1.3 on page 131	Accumulator
AESCS	0xE8	0x00	Section 8.2 on page 96	AES Command/Status
AESD	0xF3	0x00	Section 8.2 on page 96	AES Data In/Out
AESIA1	0xF5	0x00	Section 8.2 on page 96	AES Indirect Address register 1
AESIA2	0xF6	0x00	Section 8.2 on page 96	AES Indirect Address register 2
AESIV	0xF2	0x00	Section 8.2 on page 96	AES Initialization Vector
AESKIN	0xF1	0x00	Section 8.2 on page 96	AES Key In
ARCON	0xEF	0x00	Section 18.3 on page 156	Arithmetic Control register
B	0xF0	0x00	Section 15.1.4 on page 131	B register
CCEN	0xC1	0x00	Section 11.4.7 on page 111	Compare/Capture Enable register
CCH1	0xC3	0x00	Section 11.4.8 on page 111	Compare/Capture register 1, high byte
CCH2	0xC5	0x00	Section 11.4.8 on page 111	Compare/Capture register 2, high byte
CCH3	0xC7	0x00	Section 11.4.8 on page 111	Compare/Capture register 3, high byte
CCL1	0xC2	0x00	Section 11.4.8 on page 111	Compare/Capture register 1, low byte
CCL2	0xC4	0x00	Section 11.4.8 on page 111	Compare/Capture register 2, low byte
CCL3	0xC6	0x00	Section 11.4.8 on page 111	Compare/Capture register 3, low byte
CKCON	0x8E	0x01	Section 16.1 on page 134	Memory cycle control
CLKCTL	0xA3	0x80	Section 20.4.1 on page 167	
CRCH	0xCB	0x00	Section 11.4.9 on page 112	Compare/Reload/Capture register, high byte
CRCL	0xCA	0x00	Section 11.4.9 on page 112	Compare/Reload/Capture register, low byte
DPH	0x83	0x00	Section 15.1.7 on page 132	Data Pointer High
DPL	0x82	0x00	Section 15.1.7 on page 132	Data Pointer Low
DPS	0x92	0x00	Section 15.1.9 on page 133	Data Pointer Select register
FCR	0xFA	0x00	Section 17.3.6 on page 139	Flash Command register
FPCR	0xF9	NA	Section 17.3.6 on page 139	Flash Protect Configuration register
FSR	0xF8	NA	Section 17.3.6 on page 139	Flash Status register
IEN0	0xA8	0x00	Section 22.4.1 on page 172	Interrupt Enable register 0
IEN1	0xB8	0x00	Section 22.4.2 on page 173	Interrupt Priority register / Enable register 1
INTEXP	0xA6	0x01	Section 22.4.2 on page 173	

Register name	Address	Reset value	More information	Description
IP0	0xA9	0x00	Section 22.4.3 on page 173	Interrupt Priority register 0
IP1	0xB9	0x00	Section 22.4.3 on page 173	Interrupt Priority register 1
IRCON	0xC0	0x00	Section 22.4.4 on page 174	Interrupt Request Control register
MD0	0xE9	0x00	Section 18.3 on page 156	Multiplication/Division register 0
MD1	0xEA	0x00	Section 18.3 on page 156	Multiplication/Division register 1
MD2	0xEB	0x00	Section 18.3 on page 156	Multiplication/Division register 2
MD3	0xEC	0x00	Section 18.3 on page 156	Multiplication/Division register 3
MD4	0xED	0x00	Section 18.3 on page 156	Multiplication/Division register 4
MD5	0xEE	0x00	Section 18.3 on page 156	Multiplication/Division register 5
P0	0x80	0xFF	Section 13.1 on page 116	Port 0 (only P0.0 – P0.5 available externally)
P0ALT	0x95	0x00	Section 13.1 on page 116	GPIO port functions
P0DIR	0x94	0xFF	Section 13.1 on page 116	GPIO pin direction control
P0EXP	0xC9	0x00	Section 13.1 on page 116	
PCON	0x87	0x00	Section 20.4.5 on page 169	Power Control
PSW	0xD0	0x00	Section 15.1.5 on page 132	Program Status Word
PWRDWN	0xA4	0x00	Section 20.4.2 on page 168	
REGXC	0xAD	0x00	Section 19.3.6 on page 162	Control register for watchdog and wakeup functions
REGXH	0xAB	0x00	Section 19.3.6 on page 162	High byte of 16-bit watchdog/ wakeup register
REGXL	0xAC	0x00	Section 19.3.6 on page 162	Low byte of 16-bit watchdog/ wakeup register
RFCON	0x90	0x02	Section 6.5.1 on page 53	RF Transceiver configuration register
RFCTL	0xE6	0x00	Section 6.5.1 on page 53	RF Transceiver control register
RFDAT	0xE5	0x00	Section 6.5.1 on page 53	RF data register
RSTRES	0xB1	0x00	Section 20.4.3 on page 168	
S0BUF	0x99	0x00	Section 12.4.2 on page 115	Serial Port 0, Data Buffer
S0CON	0x98	0x00	Section 12.4.1 on page 114	Serial Port 0, Control register
S0RELH	0xBA	0x03	Section 12.4.3 on page 115	Serial Port 0, Reload register, high byte
S0RELL	0xAA	0xD9	Section 12.4.3 on page 115	Serial Port 0, Reload register, low byte
SMCTRL	0xB3	0x00	Section 9.2 on page 99	SPI Master Control register
SMDAT	0xB2	0x00	Section 9.2 on page 99	SPI Master data register
SP	0x81	0x07	Section 15.1.6 on page 132	Stack Pointer
SSCONF	0xBC	0x00	Section 10.2 on page 101	SPI Slave configuration
SSDAT	0xBD	0x00	Section 10.2 on page 101	SPI Slave Data register
SSSTAT	0xBE	0x00	Section 10.2 on page 101	SPI Slave Status register
T2CON	0xC8	0x00	Section 11.4.5 on page 110	Timer 2 Control register

Register name	Address	Reset value	More information	Description
TCON	0x88	0x00	Section 11.4.1 on page 108	Timer/Counter Control register
TH0	0x8C	0x00	Section 11.4.3 on page 109	Timer 0, high byte
TH1	0x8D	0x00	Section 11.4.4 on page 109	Timer 1, high byte
TH2	0xCD	0x00	Section 11.4.6 on page 110	Timer 2, high byte
TICKDV	0xB5	0x03	Section 19.3.2 on page 161	Divider for watchdog and wakeup functions
TL0	0x8A	0x00	Section 11.4.3 on page 109	Timer 0, low byte
TL1	0x8B	0x00	Section 11.4.4 on page 109	Timer 1, low byte
TL2	0xCC	0x00	Section 11.4.6 on page 110	Timer 2, low byte
TMOD	0x89	0x00	Section 11.4.2 on page 109	Timer Mode register
USBCON	0xA0	0xFF	Section 7.3 on page 65	USB configuration/status register
USBSLP	0xD9	0x00	Section 7.3 on page 65	USB sleep
WDCON	0xD8	0x00	Section 12.4.4 on page 115	Serial Port 0 Baud Rate Select register (only wdcon.7 bit used)
WUCONF	0xA5	0x00	Section 20.4.4 on page 168	Wakeup configuration register

Table 108. Special Function Registers reset values

15.1.3 Accumulator - ACC

Accumulator is used by most of the MCU instructions to hold the operand and to store the result of an operation.

Note: The mnemonics for accumulator specific instructions refer to accumulator as A, not ACC.

Address	Reset value	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xE0	0x00	acc.7	acc.6	acc.5	acc.4	acc.3	acc.2	acc.1	acc.0

Table 109. ACC register

15.1.4 B register – B

The B register is used during multiplying and division instructions. It can also be used as a scratch-pad register to hold temporary data.

Address	Reset value	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xF0	0x00	b.7	b.6	b.5	b.4	b.3	b.2	b.1	b.0

Table 110. B register

15.1.5 Program Status Word register - PSW

The PSW register contains status bits that reflect the current state of the MCU.

Address	Reset value	Bit	Name	Description
0xD0	0x00	7	cy	Carry flag: Carry bit in arithmetic operations and accumulator for Boolean operations.
		6	ac	Auxiliary Carry flag: Set if there is a carry-out from 3rd bit of Accumulator in BCD operations
		5	f0	General purpose flag 0
		4-3	rs	Register bank select, bank 0..3 (0x00-0x07, 0x08-0x0f, 0x10-0x17, 0x18-0x1f)
		2	ov	Overflow flag: Set if overflow in Accumulator during arithmetic operations
		1	f1	General purpose flag 1
		0	p	Parity flag: Set if odd number of '1' in ACC.

Table 111. PSW register

Note: The Parity bit can only be modified by hardware in the ACC register state.

15.1.6 Stack Pointer – SP

This register points to the top of the stack in internal data memory space. It is used to store the return address of a program before executing an interrupt routine or subprograms. The SP register is incremented before executing PUSH or CALL instruction and it is decremented after executing POP or RET(I) instruction (it always points to the top of the stack).

Address	Reset value	Register name
0x81	0x07	SP

Table 112. SP register

15.1.7 Data Pointer – DPH, DPL

Address	Reset value	Register name
0x82	0x00	DPL
0X83	0x00	DPH

Table 113. Data Pointer register (DPH:DPL)

The Data Pointer registers can be accessed through DPL and DPH. The actual data pointer is selected by DPS register.

The Data Pointer registers are intended to hold a 16-bit address in the indirect addressing mode used by MOVX (move external memory), MOVC (move program memory) or JMP (computed branch) instructions. They may be manipulated as 16-bit register or as two separate 8-bit registers. DPH holds a higher byte and DPL holds a lower byte of an indirect address.

These registers are used to access external code or data space (for example, MOVC A, @A+DPTR or MOV A, @DPTR).

15.1.8 Data Pointer 1 – DPH1, DPL1

Address	Register name
0x84	DPL1
0X85	DPH1

Table 114. Data Pointer 1 register (DPH1:DPL1)

The Data Pointer register 1 can be accessed through DPL1 and DPH1. The actual data pointer is selected by DPS register.

These registers are intended to hold a 16-bit address in the indirect addressing mode used by MOVX (move external memory), MOVC (move program memory) or JMP (computed branch) instructions. They can be manipulated as a 16-bit register or as two separate 8-bit registers. DPH1 holds a higher byte and DPL1 holds a lower byte of an indirect address.

These registers are used to access external code or data space (for example, MOVC A,@A+DPTR or MOV A,@DPTR respectively).

The Data Pointer 1 is an extension to the standard 8051 architecture to speed up block data transfers.

15.1.9 Data Pointer Select register – DPS

The MCU contains two Data Pointer registers. Both Data Pointer registers can be used as 16-bits address source for indirect addressing. The DPS register serves for selecting the active data pointer register.

Address	Reset value	Bit	Name	Description
0x92	0x00	7:1	-	Not used
		0	dps	Data Pointer Select. 0: select DPH:DPL, 1: select DPH1:DPL1

Table 115. DPS register

16 Random Access Memory (RAM)

The nRF24LU1+ contains two separate RAM blocks. These blocks are used to save temporary data or programs.

The RAM blocks are 256x8 IRAM bits and 2048x8 bits.

Note: The information in these blocks is lost when power to the device is removed.

As described in chapter 15 on page 127, the RAM resides in different maps, that is, different instructions are used to access them.

The smallest RAM-block (256 bytes) resides in the “internal” RAM-area, called IRAM, and contains scratch-pad data, subroutine stacks, register files, and so on.

Note: The lower 128 bytes can be addressed direct or indirectly, while the upper 128 bytes can only be accessed using indirect addressing.

The largest RAM (2048 bytes) resides in the XDATA space and is a fixed block located from address 0x8000 to 0x87FF. This block is used for data or program code.

16.1 Cycle control

The MCU has a programmable SFR register that controls timing to on-chip memory. Since this memory is fast, the default values are recommended.

Address	Reset value	Bit	Description
0x8E	0x01	7	Not used
		6-4	Program memory wait state control 000: No wait-states (default at power-up) Other values are reserved and should not be used
		3	Not used
		2-0	External data memory stretch control. 001: One stretch cycle (default) Other values are reserved and should not be used

Table 116. CKCON register

17 Flash Memory

This section describes the operation of the embedded flash memory. The MCU can read and write the memory and perform erase page operations. You can configure and program the flash memory through an external SPI slave interface and you can also configure it to inhibit readback or modification of the memory content. You can also program the flash memory through the USB by using the USB bootloader.

17.1 Features

- 16 or 32 kB Flash memory
- Page size 512 bytes
- 32 or 64 pages of MainBlock + 1 InfoPage
- Endurance minimum 1000 write/erase cycles
- Direct SPI programmable
- Read and write accessible from MCU
- Configurable MCU write and erase protection
- Configurable SPI readback protection
- HW support for FW upgrade

17.2 Block diagram

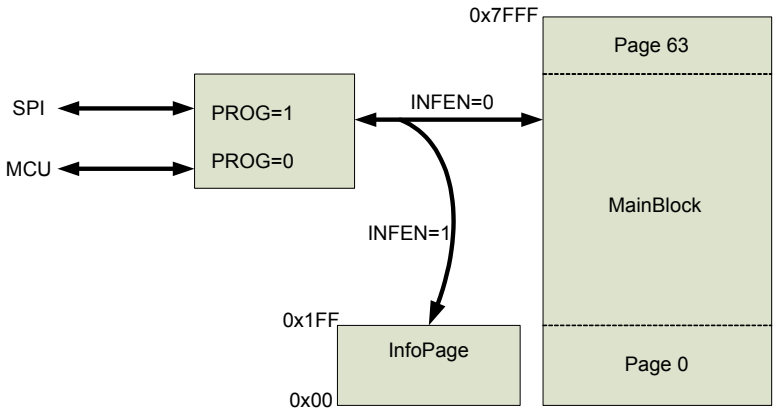


Figure 62. Flash memory block diagram

17.3 Functional description

17.3.1 Flash memory configuration

The flash memory is divided into two blocks, the MainBlock and the information block (one InfoPage).

At the chip interface the flash behaves as an SPI programmable flash memory. All configuration and setup of the behavior during normal mode (that is, when MCU is active and running) is defined through the SPI

and the configuration data is stored in the InfoPage. During the chip reset/start-up sequence the configuration data is read and stored in a set of registers that control flash memory behavior.

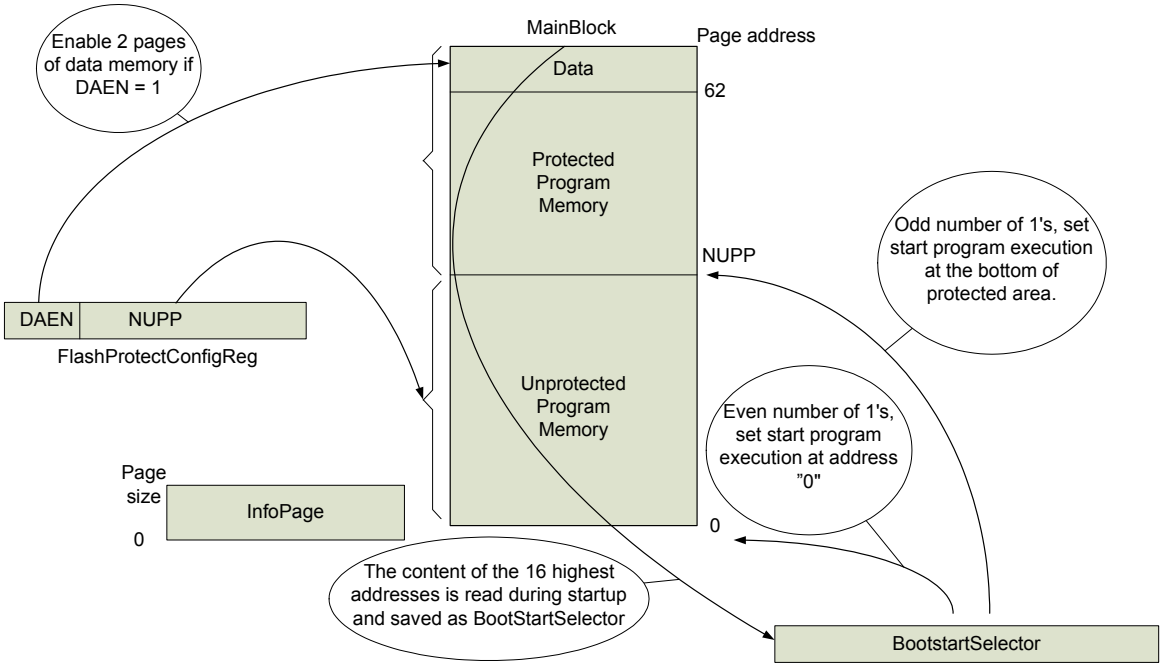


Figure 63. Flash memory organization

The MainBlock of flash memory can be configured through the SPI (Figure 63.) in the following ways:

- Unprotected program memory; can be read, written and erased through the SPI and by the MCU. With no configuration the whole MainBlock is unprotected memory.
- Protected program memory; can be read, written and erased through the SPI but only read by the MCU. Any number of pages can be configured as protected memory.
- Data memory; can be read, written and erased through the SPI and by the MCU. Two data pages 62 and 63 may be enabled. If enabled, the number of protected pages is reduced by two.

You can program the InfoPage through the SPI only, because the MCU has only read access to the InfoPage, except for byte 0x23, which is writable from the MCU, see Table 117.

Note: You can write to a flash byte twice between each erase. A page (or full) erase sets the value of all erased bytes to 0xFF. A write to a flash byte only programs the 0-bits of the byte, leaving the 1-bits as they are (after the erase). Thus for the second write, only the remaining 1-bits can be programmed. This results in an "and" function of the first and second write. Example: Writing 0x46 and 0x27 to the same byte (without erase in between) will result in a stored value 0x06.

17.3.2 InfoPage content

The content of the InfoPage is given in Table 117. below. The InfoPage is erased at factory so that all bytes, except CHIPID, initially have the value 0xFF.

InfoPage data	Size	Address	Comment
Reserved	11 bytes	0x00	Reserved.
CHIPID	5 bytes	0x0B	ID number for each individual device. The ID is generated by a pseudo random process. No ID will violate the rules specificed in section 6.4.3.2
Reserved	16 bytes	0x10	Reserved.
Page address for start of protected area (that is, number of unprotected pages)	1 byte	0x20	Read out during reset/start-up sequence of the chip to register FPCR.NUPP Byte value: 0xFF: all pages are unprotected Other value: page address
Enable flash data memory	1 byte	0x21	Read out during reset/start-up sequence of the chip. Byte value: 0xFF: no data memory, FPCR.DAEN=0 Other value: data memory exists, FPCR.DAEN=1
Readback blocking byte for InfoPage	1 byte	0x22	Read out during reset/start-up sequence of chip. Byte value: 0xFF: FSR.RDISIP=0 Other value: FSR.RDISIP=1
Readback blocking byte for MainBlock	1 byte	0x23	Read out during reset/start-up sequence of chip. Byte value: 0xFF: FSR.RDISMB=0 Other value: FSR.RDISMB=1 **Note:** This byte may be written (to 0x00) by the MCU, but the InfoPage cannot be erased by the MCU.
Enable debug	1 byte	0x24	Read out during reset/start-up sequence of chip. Byte value: 0xFF: FSR.DBG=0 Other value: FSR.DBG=1 if FSR.RDISMB=0, enables HW debug. See chapter 23 on page 175
Reserved	219 bytes	0x25	Reserved.
For user data	256 bytes	0x100	Free to use.

Note: The InfoPage can only be erased if FSR.RDISMB= 0. See also section 17.7.4.1 on page 154.

Table 117. InfoPage content

17.3.3 Protected pages and data pages

The flash memory can be split into unprotected, protected and data areas. When you protect an area of flash memory it becomes read only for the MCU, but it can still be read, written and erased through the SPI interface. The protected area of the flash memory is safe from illegal erase/write operations from the MCU and can typically be used for firmware upgrade functions, see section 17.5.2 on page 141.

The flash MainBlock consists of 32 or 64 pages each 512 bytes long. The factory configuration FPCR.NUPP (Number of Unprotected Pages)=0xFF leaves all pages unprotected, that is, the MCU can erase and write to any page. FPCR.DAEN=1 reserves the two highest pages (62-63) as data pages. The FPCR.NUPP value is the page number of the first protected page, see Figure 62. on page 135. For exam-

ple, FPCR.NUPP=12 and FPCR.DAEN=1 gives 12 unprotected pages (0-11) and 50 protected pages (12-61) and two data pages (62-63), see Figure 63. on page 136.

If you have a protected and unprotected area, the value of the 16 topmost flash memory addresses decide from which area the MCU will start code execution. During the reset/start up sequence these 16 bytes are read automatically. If there is an even number of logic 1 databits in the 16 topmost addresses of the flash memory, FSR.STP = 0, otherwise FSR.STP = 1. The factory configuration of FSR.STP=0 starts the code execution at address 0x0000 which is the beginning of the unprotected area. If FSR.STP=1, the code execution starts from the beginning of the protected area.

To use this "start from protected" feature, data pages must be enabled (FPCR.DAEN=1), which will allow MCU programming access to the highest page (63).

17.3.4 16 kB Flash memory size option

nRF24LU1+ has two flash memory size options, 32 k bytes and 16 k bytes, see also chapter 27.1.1 on page 181.

For the 16 kB Flash version the data pages, if configured, always reside in pages 62-63.

Any MCU program, with a size less than 16 kB, running on a 32 kB Flash version, will run unmodified in a 16 kB Flash version. Likewise any MCU program running on a 16 kB Flash version will run unmodified on a 32 kB Flash version.

Example configurations

FPCR. NUPP	FPCR. DAEN	Flash memory size	Description
255	0	32 kB	All memory unprotected (pages 0-63)
20	0	32 kB	NA because FSR.STP cannot be set
0	0	32 kB	All memory protected (pages 0-63)
255	1	32 kB	Pages 0-61 unprotected, data pages 62-63
40	1	32 kB	Pages 0-39 unprotected, pages 40-61 protected, data pages 62-63
20	1	32 kB	Pages 0-19 unprotected, pages 20-61 protected, data pages 62-63
0	1	32 kB	Pages 0-61 protected, data pages 62-63
255	0	16 kB	All memory unprotected (pages 0-31)
20	0	16 kB	NA because FSR.STP cannot be set
0	0	16 kB	All memory protected (pages 0-31)
255	1	16 kB	Pages 0-29 unprotected, data pages 62-63
40	1	16 kB	Illegal (values >29 and < 64 are illegal)
20	1	16 kB	Pages 0-19 unprotected, pages 20-29 protected, data pages 62-63
0	1	16 kB	Pages 0-29 protected, data pages 62-63

Table 118. Example of Flash configurations for 32 kB and 16 kB size

17.3.5 Software compatibility with nRF24LU1

nRF24LU1 programs with no protected flash memory configured (FPCR.NUPP=0xFF) can be ported unchanged to nRF24LU1+.

For nRF24LU1 programs with protected flash memory configured (FPCR.NUPP < 30), the addresses of the data pages must be changed. This is because the data pages in nRF24LU1 reside in pages 30-31 while the data pages in nRF24LU1+ reside in pages 62-63. This means that all addresses for write or erase of the data pages in nRF24LU1 must be translated from pages 30-31 to pages 62-63 to run on nRF24LU1+. However, this address translation is not required if no protected memory is configured.

17.3.6 SFR registers for flash memory operations

Addr	Reset value	Bit	Name	RW	Function
0xF8	Read from Flash, see Table 117.	7	DBG	RW	FSR – Flash Status Register 1: Enable HW debugger, 0: Disable HW debugger, if RDISMB = 0, DBG may be set (to one) by the MCU/SFR write, but it will return back to reset value after a subsequent reset.
	Read from Flash	6	STP	R	1: Start from protected program memory, 0: start from address 0
	0	5	WEN	RW	Flash write (or erase) enable. 1: enable
	0	4	RDYN	R	Flash interface ready. 0: ready
	0	3	INFEN	RW	InfoPage enable. 1: enable
	Read from Flash, see Table 117.	2	RDISMB	R	SPI read-back disable of MainBlock. 1: read back disable and also inhibits page erase and MainBlock write. Writable by SPI command RDISMB and is cleared automatically by the ERASE_ALL command. Can also be written indirectly by MCU as described in Table 117. on page 137.
	Read from Flash	1	RDISIP	R	SPI read-back disable of InfoPage. 1: read back disable. Only writable by SPI command RDISIP and is cleared automatically if InfoPage is erased.
	0	0	-		Reserved, read as 0.
0xF9	N/A	7 6:0	DAEN NUPP	R R	FPCR – Flash Protect Configuration Register 1: The two upper flash pages reserved as data memory, 0: no data memory enabled. Number of unprotected pages. **Note:** NUPP < 0xFF reserves the 16 highest bytes of the flash MainBlock.
0xFA	0x00	7:0	EPA	RW	FCR – Flash Command Register Byte value < 64: Erase page address, otherwise used for secure MCU flash write, see section 17.5.1 on page 140.

Table 119. FSR, FPCR and FCR registers

17.4 Brown-out

There is an on-chip power-fail detector, see chapter 21 on page 170, which ensures that any flash memory program or erase access will be ignored when the 'Power Fail' signal (see Figure 83.) is active. Both the micro controller and the flash memory still function according to specification, and any write operation that was started will be completed. Flash erase operations will be aborted. If the supply voltage drops further, that is when the signal "Reset" (see Figure 83.) is active, the chip will be reset. If the power supply rises again before reaching the reset threshold, there will be no reset, and there is no status indication to show

that this has happened.

To ensure proper programming of the flash in the cases where power supply may be unreliable, the user should take the following precautions:

- Make sure there is no partial erase.
  - If the device is reset during an erase cycle, always assume that the erase was unsuccessful.
  - If there is no reset, make sure that the erase duration is longer than 20 ms. A sample firmware code for such a check may be found in nRFGo SDK.
- Make sure the data read back from the flash is identical to what is written to flash. The mechanism above will guarantee that the data is safely stored to flash if the value does compare. If the compare fails, the write has been ignored due to a power supply event.
- Using the VBUS supply, the time from “Power Fail” to “Reset” is longer than one flash byte write operation (around 46 µs), as this is assured by the 10µF capacitor on the VBUS pin. If using the VDD supply, make sure that this requirement is met by sufficient reservoir on the supply.

17.5 Flash programming from the MCU

This section describes how you can write and erase the flash memory using the MCU. Note that all flash write and erase operations require that Cclk ≥12 MHz, see Table 133. on page 167.

17.5.1 MCU write and erase of the MainBlock

When a flash write is initiated, the MCU is halted for 740 clock cycles (46µs @16 MHz) for each byte written. When a page erase is initiated, the MCU can be halted for up to 360,000 clock cycles (22.5 ms @16 MHz). During this time the MCU does not respond to any interrupts. Firmware must assure that page erase does not interfere with normal operation of the nRF24LU1+.

The MCU can perform erase page and write operations to the unprotected part and the data part of the flash MainBlock. To prevent unwanted/harmful erase and write operations a security mechanism is implemented.

To allow erase and write flash operations the MCU must run the following sequence:

1. Write 0xAA to the `FCR` register. This starts an internal 7 bit down counter. The counter counts down from 127 to 0.
2. Before the count down period has expired (8 µs @16 MHz), write 0x55 to the `FCR`. This restarts the internal 7 bit down counter. Then the counter again counts down from 127 to 0. In the count down period (8 µs) the `FSR.WEN` bit is writeable from the MCU.
3. Set `FSR.WEN` high before count down period has expired.
4. The flash is now open for erase and write from the MCU until `FSR.WEN` is set low again. `FSR.WEN` can be set low directly (no security mechanism applies).
5. To erase a page, write page address (range 0-63) to the `FCR` register. Bytes are written individually (there is no auto increment) to the flash using the specific memory address. When the programming code executes from the flash, any erase or write operation is self timed and the MCU stops until the operation is finished. If the programming code executes from the XDATA RAM the code must wait until the operation has finished. This can be done either by polling the `FSR.RDYN` bit to go low or by a wait loop. Do not set `FSR.WEN` low before the write or erase operation is finished. Memory address is identical to the flash address, see Chapter 15 on page 127 for memory mapping.

17.5.2 Hardware support for firmware upgrade

If the FSR.STP bit is high the MCU starts a program execution from the lowest address in the protected part of the flash memory (FPCR.NUPP<<9). If the FSR.STP bit is low (as it is in a normal case) the MCU starts an execution from address “0” of the flash memory.

The FSR.STP bit can only be set indirectly by programming one of the 16 last bytes of flash MainBlock. Upon the next Reset these 16 bytes will be read, and if the data area is enabled (FPCR.DAEN = 1) and the content of the 16 highest addresses of the flash memory (MainBlock) have an odd number of 1’s then FSR.STP is set high. Otherwise, FSR.STP is set low. If the data area is not enabled (FPCR.DAEN = 0) FSR.STP is always low. This mechanism may be used to obtain a safe upgrade for the unprotected area of the flash.

There is only one set of interrupt vectors (see Table 138.) and they always point to the first page of flash memory, regardless of whether the page is defined as protected or unprotected memory. So, if a program is placed in the protected part of flash memory starting at a higher page, the corresponding interrupt vectors still point to the unprotected part of the memory. Unless the implications of this are clearly understood, it is recommended not to use interrupt in programs intended to run from the protected area.

Here is an example of this mechanism in use:

- The application is running in an unprotected area and the program doing the upgrade resides in a protected area.
- You must configure flash with FPCR.DAEN=1 to allow firmware control over the FSR.STP bit.
- The host initiates a firmware upgrade over the USB interface.
- Set FSR.WEN as described in section 17.5.1 on page 140.
- A bit in one of the 16 highest addressed bytes is programmed to 0.
- You can now restart the system. The system restarts from the protected area.
- You can now perform erase and write operations safely in the unprotected area, that is, you can update the unprotected area through the USB interface.
  In case of a power failure or a restart, the MCU starts an execution in the protected area.
- When the upgrade is finished a new bit in one of the 16 highest addressed bytes is programmed to 0 which gives an even number of 1s in these bytes, implying FSR.STP=0, after next reset.
- You can now restart the system, and it restarts from the unprotected area.

  Note: For program in protected area, see the restriction for movc in Figure 61. on page 127

17.6 Flash programming through USB

The nRF24LU1+ bootloader allows you to program the nRF24LU1+ through the USB interface. The bootloader is pre-programmed into the nRF24LU1+ flash memory and automatically starts when power is applied. After start-up the bootloader copies the flash programming code to the internal SRAM from where the complete flash memory can be programmed. The bootloader occupies the topmost 2K bytes (KB) of the flash and is not deleted unless the user program extends into this area. If the program is larger than 30KB the bootloader is overwritten and lost. In addition to the topmost 2KB of the flash, the bootloader also uses the 3 byte reset vector at address 0. If your application needs to re-execute the bootloader; you must restore the reset vector so that the bootloader executes after power on reset.

17.6.1 Flash Layout

The 32 kB flash is divided into 64 pages of 512 bytes each. Since the maximum USB packet size in nRF24LU1+ is 64 bytes the bootloader divides each flash page into 8 blocks of 64 bytes each as shown in the following figure:

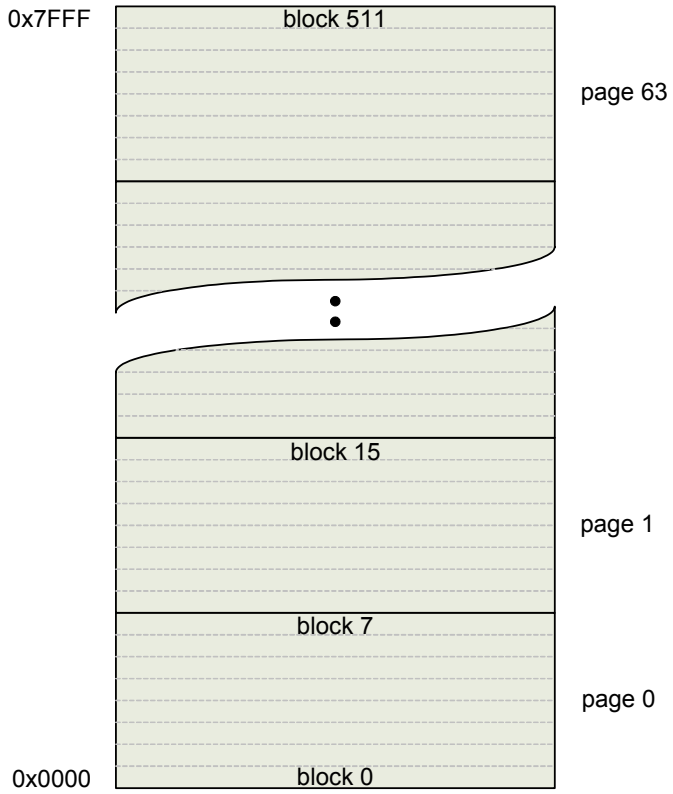


Figure 64. Relation between address, pages and blocks

17.6.2 USB Protocol

nRF24LU1+ begins enumerating when connected to a USB host and is available to the operating system. A driver, or application, communicating with the bootloader must use the following parameters:

USB parameters	Value
VID (Vendor Identification)	0x1915
PID (Product Identification)	0x0101
IN endpoint address	0x81
OUT endpoint address	0x01

Table 120. Driver/application USB parameters for communication with bootloader

The USB host communicates with the bootloader by writing commands to the IN endpoint. All USB commands to the bootloader start with a 1 byte identifier (cmd id) and return a packet. If the command takes parameters, the parameters are written as one or two bytes after the one byte identifier, for example, command 0x02 takes the parameter pn. Each command returns a value that must be read from the OUT endpoint.

17.6.2.1 Firmware Version (cmd id 0x01)

Figure 65. Command 0x01

This command returns firmware information of the bootloader. **hb** is the major version number and **lb** is the minor version number.

17.6.2.2 Flash Write Init (cmd 0x02)

This command is sent to the bootloader to start writing a flash page to the page indicated by the parameter **pn**. After this command is completed the host must send the 8 blocks that constitutes the page. This is done by sending 64 byte packets to the USB IN endpoint with the contents of the block. The bootloader responds with a one-byte packet (containing 0x00) after each packet has been sent.

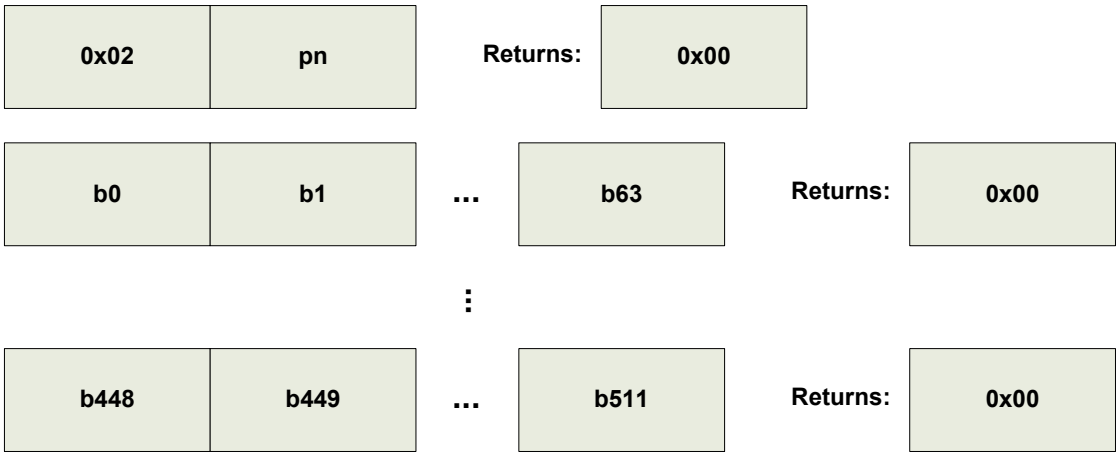


Figure 66. Command 0x02

17.6.2.3 Read Flash (cmd 0x03)

This command is sent by the host to read one of the 64 byte flash blocks. The block is indicated by the parameter **bn** (bits 0-7 only. Bit 8 is set by cmd 6 below). If the flash MainBlock readback disable is effective this command returns 0x00 in all bytes in the flash page containing the block is used (programmed) and 0xff if it is empty.

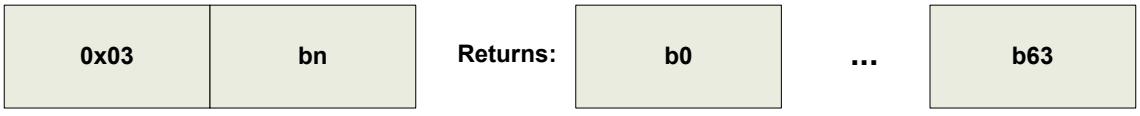


Figure 67. Command 0x03

17.6.2.4 Flash Page Erase (cmd 0x04)

This command is used mainly for debugging purposes since the Flash Page Write command above automatically erases the page if needed prior to programming. Using this command erases page **pn**.

Figure 68. Command 0x04

17.6.2.5 Turn on flash MainBlock readback disable (cmd 0x05)

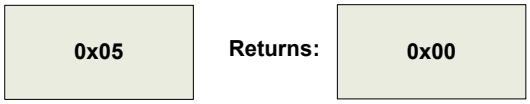


Figure 69. Command 0x05

This command returns 0x00 if successful or 0x01 if the bootloader failed to turn on flash MainBlock readback disable. The device must be reset for the readback disable to be effective.

17.6.2.6 Select flash half (cmd 0x06)

Figure 70. Command 0x06

This command is used to select which flash half command 0x03 works against (bit 8 of the block number). When **n** = 0 the lower 16 kB flash is selected and when **n** = 1 the upper 16 kB is selected. The reason for having this command is to make sure the interface and commands for the nRF24LU1 bootloader are the same on the nRF24LU1+.

17.7 Flash programming through SPI

The on-chip flash is designed to interface a standard SPI device for programming. The interface uses an 8 bit instruction register and a set of instructions/commands to program and configure the flash memory. Note that all flash write and erase operations require that Cclk ≥12 MHz, see Table 133. on page 167.

17.7.1 SPI commands

To allow access through the SPI the external **PROG** pin must be set high during all flash operation commands. After activation of the **PROG** pin you must wait at least 1.5 ms before you input the first flash command. When the **PROG** pin is set, the **GPIO** pins are automatically configured as slave SPI (see chapter 13 on page 116. Further description of SPI slave is found in chapter 10 on page 101). Before each flash write or erase command, FSR.WEN must be set, because this bit is automatically cleared after any write or erase command. The value of FSR.INFEN always decides if access goes to the flash MainBlock or the InfoPage.

Command	Command format	Address	Number of databytes	Command operation
WREN	0x06	NA	0	Set flash write enable,FSR.WEN
WRDIS	0x04	NA	0	Reset flash write enable, FSR.WEN
RDSR	0x05	NA	1 (or more)	Read Flash Status Register (FSR)
WRSR	0x01	NA	1	Write Flash Status Register (FSR). Only bits 5 and 3 (WEN and INFEN) are writable by this command.
READ	0x03	Start address, 2 bytes	1-32768	Read data from flash
PROGRAM	0x02	Start address, 2 bytes	1-256	Write data to flash
ERASE PAGE	0x52	page number, 1 byte	0	Erase addressed flash page
ERASE ALL	0x62	NA	0	Erase all pages of flash MainBlock
RDFPCR	0x89	NA	1	Read Flash Protect Configuration Register (FPCR)
RDISIP	0x84	NA	0	Set flash InfoPage read-back disable, FSR.RDISIP, and write 0x00 to InfoPage byte 0x22.
RDISMB	0x85	NA	0	Set flash MainBlock read-back disable, FSR.RDISMB and write 0x00 to InfoPage byte 0x23.
ENDEBUG	0x86	NA	0	Write 0x00 to InfoPage byte 0x24, and after next reset, the HW debugger will be enabled, see chapter 23 on page 175 for details, and FSR.DBG will be set.

Table 121. Flash SPI operation commands

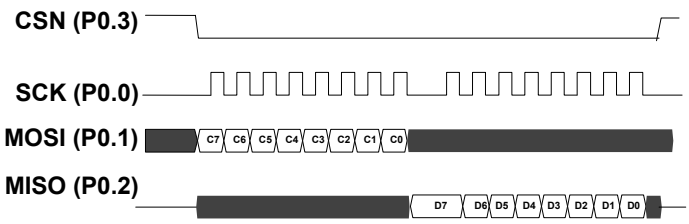


Figure 71. SPI read command without address

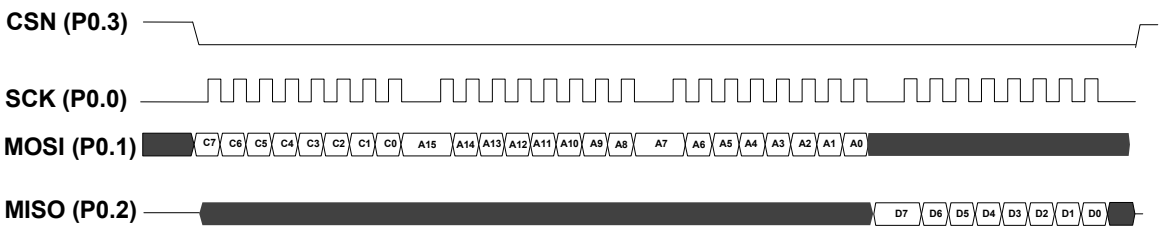


Figure 72. SPI flash read operation, shown with 1 databyte

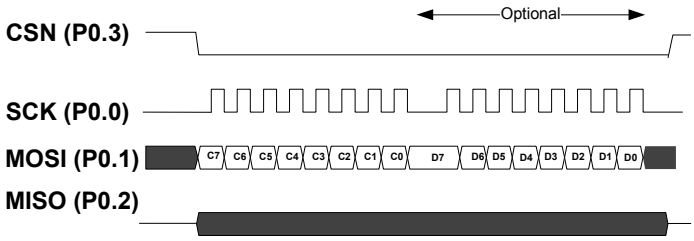


Figure 73. SPI write command without address

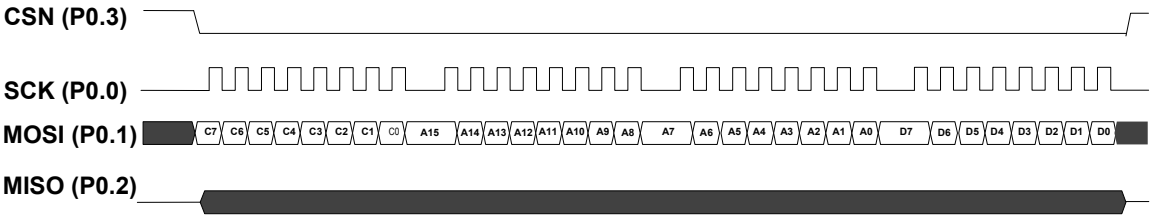


Figure 74. SPI flash write operation, shown with 1 databyte

An SPI command always starts with the external master sending a command byte to the flash slave, followed by a variable number of address and data bytes. The number of address and data bytes are specific to each command, as shown in Table 121. on page 145. In Figure 71. to Figure 74. Cn is the SPI command bit, An is the address bit and Dn is the data bit (note: MSBit in each byte first). After CSN is deactivated, a flash write or erase command requires the chip to do the flash programming or erase, and this will take some time to complete therefore, it is advised not to issue a new command until the specified amount of time has elapsed. Alternatively, you can repeatedly issue RDSR commands until the FSR.RDYN bit reads back as 0.

17.7.1.1 WREN/WRDIS

SPI command WREN sets the flash write enable bit FSR.WEN, and SPI command WRDIS resets FSR.WEN. This bit enables all SPI write and erase operations to the flash memory. The device powers up in write disable state and will automatically return to write disable state after any SPI flash write or erase command. Each SPI flash write and erase instruction must therefore be preceded by a WREN command. Both WREN and WRDIS are 1-byte commands with no data.

17.7.1.2 RDSR

The SPI command RDSR reads out the content of the flash status register FSR, and consists of 1-command byte and 1-data byte as shown in Figure 71. on page 146. By keeping the CSN line active after the first data byte, FSR will be repeatedly re-read until CSN is set inactive.

17.7.1.3 WRSR

The SPI command WRSR writes to the flash status register FSR, and consists of a 1-command byte and 1-data byte as shown in Figure 73. on page 146.

17.7.1.4 READ

The SPI command READ reads out the content of the flash memory, starting from the given address. If FSR.INFEN=0, flash MainBlock will be read. If FSR.INFEN=1, flash InfoPage will be read. The following sequence is required:

1. The CSN line is activated (that is, pulled low) to enable/ activate the SPI slave.
2. The READ command is transmitted through the MOSI line followed by the two byte address to the byte to be read as shown in Figure 72. on page 146.
3. The addressed data byte is shifted out on the MISO line.

If the CSN line is kept active after the first byte is read out the read command can be extended, the address is auto incremented and data continues to be shifted out. The internal address counter rolls over when the highest address is reached, allowing the complete memory to be read in one continuous read command.

A readback of the flash content is only possible if the respective read disable bit FSR.RDISMB or FSR.RDISIP is not set.

17.7.1.5 PROGRAM

The SPI command PROGRAM writes to (or programs) the flash memory, starting from the given address. If FSR.INFEN=0, flash MainBlock will be written to. If FSR.INFEN=1, flash InfoPage will be written to. The following sequence is required:

1. Enable the device for writing (set FSR.WEN) using the WREN or WRSR command.
2. The CSN line is pulled low to enable the SPI slave.
3. The PROGRAM command is sent on the MOSI line followed by the two-byte address (address of the first byte) and the data to be programmed/ written, as shown in Figure 74. on page 146.
4. The on-chip driven program sequence is started when you set the CSN pin high/ deactivated.
5. Programming n bytes takes (n + 1)*365 clock cycles (XC1) after CSN is deactivated. During the program sequence all SPI commands are ignored except the RDSR command.

The CSN line can be kept active to write up to 256 bytes (with address auto increment) in one PROGRAM command. The first byte can be anywhere in a page. Normally, a byte can not be reprogrammed without erasing the whole page, see also the Note: on page 136.

Note: FSR.RDISMB = 1 inhibits the write to flash MainBlock.

The device returns to write disable after completion of a PROGRAM command.

17.7.1.6 ERASE PAGE

The SPI command ERASE PAGE erases 1 addressed page in the flash memory, so that the content of the page will be 0xFF. If FSR.INFEN=0, a page in flash MainBlock will be erased. If FSR.INFEN=1, flash InfoPage will be erased. The following sequence is required:

1. Enable the device for writing (set FSR.WEN) using the WREN or WRSR command.
2. The CSN line is pulled low to enable the SPI slave.
3. The ERASE PAGE command is sent on the MOSI line followed by the page number (0-63) to be erased.
4. The on-chip driven erase sequence is started when the CSN pin is set high.
5. Erasing a page takes about 360000 clock cycles (XC1) after CSN is deactivated. During the erase sequence all SPI commands are ignored except the RDSR command.

Note: FSR.RDISMB = 1 inhibits page erase of both flash MainBlock and the InfoPage. The device returns to write disable after completion of an ERASE PAGE command.

17.7.1.7 ERASE ALL

The SPI command ERASE ALL erases all content of the flash MainBlock memory to value 0xFF. The following sequence is required:

1. Enable the device for writing (set FSR.WEN) using the WREN or WRSR command.
2. The CSN line is pulled low to enable the SPI slave.
3. The ERASE ALL command is sent on the MOSI line.
4. The on-chip driven erase sequence is started when the CSN pin is set high.
5. ERASE_ALL takes about 360000 clock cycles (XC1) after CSN is deactivated. During the erase sequence all SPI commands are ignored except the RDSR command.

The ERASE_ALL command cannot be used to erase the flash InfoPage, but is always allowed to erase flash MainBlock. The device returns to write disable after completion of an ERASE ALL command.

17.7.1.8 RDFPCR

The SPI command RDFPCR reads out the content of the flash project configuration register FPCR, and consists of 1 command byte and 1 data byte as shown in Figure 71. on page 146.

17.7.1.9 RDISIP

Flash InfoPage readback disables and writes 0x00 to byte 0x22 in flash InfoPage. The command disables all read access to the flash InfoPage from the external SPI interface. This is a single byte command with no data.The following sequence is required:

1. Enable the device for writing (set `FSR.WEN`) using the `WREN` or `WRSR` command.
2. The CSN line is pulled low to enable the SPI slave.
3. The `RDISIP` command is sent on the MOSI line.
4. The on-chip driven program sequence is started when the CSN pin is high/deactivated.
5. The program sequence takes 730 clock cycles (XC1) after CSN is deactivated. During this sequence all SPI commands are ignored except the `RDSR` command.

The device returns to write disable after completion of an RDISIP command. `FSR.RDISIP` can only be cleared by erasing flash InfoPage with an ERASE PAGE command, which is only allowed if `FSR.RDISMB`=0.

17.7.1.10 RDISMB

Flash MainBlock readback disables and writes 0x00 to byte 0x23 in flash InfoPage. The command disables all read, write and page erase access to the flash MainBlock from any external interface (SPI or HW debugger). It also prevents erase of flash InfoPage and enabling of HW debugger. This will protect the content of the flash MainBlock from being read out over the external SPI or HW debugger interface. This is a single byte command with no data. The following sequence is required:

1. Enable the device for writing (set `FSR.WEN`) using the `WREN` or `WRSR` command.
2. The CSN line is pulled low to enable the SPI slave.
3. The `RDISMB` command is sent on the MOSI line.
4. The on-chip driven program sequence is started when the CSN pin is high/deactivated.
5. The program sequence takes 730 clock cycles (XC1) after CSN is deactivated. During this sequence all SPI commands are ignored except the `RDSR` command.

The device returns to write disable after completion of an RDISMB command. The only way to clear FSR.RDISMB is to erase the whole flash MainBlock with an ERASE ALL command, and thereafter do an ERASE PAGE of the flash InfoPage to set the value of byte 0x23 to 0xFF.

17.7.1.11 ENDEBUG

The SPI command ENDEBUG writes 0x00 to the flash InfoPage byte 0x24, and after the next reset `FSR.DBG` is set and the HW Debugger is enabled. To clear FSR[7] first erase the flash InfoPage, and then perform a reset. This is a single byte command with no data. This command is only allowed if `FSR.DBG`=0, and is ignored otherwise.

The following sequence is required:

1. Enable the device for writing (set `FSR.WEN`) using the `WREN` or `WRSR` command.
2. The CSN line is pulled low to enable the SPI slave.
3. The ENDEBUG command is sent on the MOSI line.
4. The on-chip driven program sequence is started when the CSN pin is high/deactivated.
5. The program sequence takes 730 clock cycles (XC1) after CSN is deactivated. During this sequence all SPI commands are ignored except the RDSR command.

FSR.DBG may be set (only when FSR.RDISMB=0) without involving the flash InfoPage, by the following sequence:

MCU writes a "1" to FSR.DBG. The HW debugger is then temporarily enabled until next reset, when the flash InfoPage value takes control.

Note: There is no SPI command for directly setting or clearing FSR.DBG.

The device returns to write disable after completion of an ENDEBUG command.

17.7.1.12 SPI Readback disable

A mechanism to prevent readback over the external SPI interface is implemented. Two bytes of the InfoPage are reserved for this. The InfoPage and MainBlock each have their own readback disable signal, FSR.RDISIP and FSR.RDISMB. The InfoPage content is checked whenever the chip is started or restarted. If byte 0x22 of InfoPage=0xFF, FSR.RDISIP=0, otherwise FSR.RDISIP=1. If byte 0x23 of InfoPage=0xFF, FSR.RDISMB=0, otherwise FSR.RDISMB=1.

The InfoPage bytes may be written once, (by using command SPI commands RDISIP or RDISMB) which enables the readback disable function. The InfoPage byte 0x23 (RDISMB) is also writable by MCU. Until the flash memory is erased again, the readback function cannot be enabled. FSR.RDISMB=1 inhibits page erase and write to flash (both MainBlock and InfoPage), but erase all is always allowed, as it is the only way to clear FSR.RDISMB.

17.7.2 Standalone programming requirements

When programming the nRF24LU1+ in a standalone flash/EEPROM/MCU programmer, an adapter board (or socket assembly) with capacitors and resistors and possibly an oscillator are required. The following table describes how the device pins are used:

Signal	Pins	Disposition	Further information
VDD	1, 3, 9, 19, 24, 27	Connect together to supply and decouple	See 17.7.2.2
VBUS	3	Open	See 17.7.2.2
D+	4	Leave open	
D-	5	Leave open	
VSS	6, 12, 17, 18, 23, 26, 30	Connect to ground net (plane)	See 17.7.2.2
PROG	7	Connect to pin electronics or strap to VDD	See 2.2.2
RESET	8	Connect to pin electronics or strap to VDD	See 2.2.7
SCK	10	Connect to pin electronics (nRF24LU1+ in)	See 17.7.1
MOSI	11	Connect to pin electronics (nRF24LU1+ in)	See 17.7.1
MISO	13	Connect to pin electronics (nRF24LU1+ out)	See 17.7.1
CSN	14	Connect to pin electronics (nRF24LU1+ in)	See 17.7.1
P0.4	15	Leave open	
P0.5	16	Leave open	
VDD_PA	20	Leave open	

Signal	Pins	Disposition	Further information
ANT1	21	Leave open	
ANT2	22	Leave open	
IREF	25	Leave open	
DEC1	28	Leave open	
DEC2	29	Decouple to VSS	See 17.7.2.2
XC2	31	leave open (optionally connect to XTAL)	See 17.7.2.1
XC1	32	Connect to clock source	See 17.7.2.1

Table 122. Device pins

17.7.2.1 Clock requirements

The XC1 requires a clock between 9 MHz and 16 MHz during the entire programming sequence. The programming speed is directly proportional to the speed of the clock so, we recommend a clock speed of 16 MHz +/- 60ppm. The clock source can be a crystal between `XC1` and `XC2` or an external clock source connected to the `XC1` pin:

- From an oscillator module on the adapter board or a pin driver in the programmer.
  - Required amplitude is at least 0.5V p-p, maximum 3.3V p-p.
  - Waveform is sine or square.
  - Required duty cycle is 40% to 60% (V/2 in the sine case).

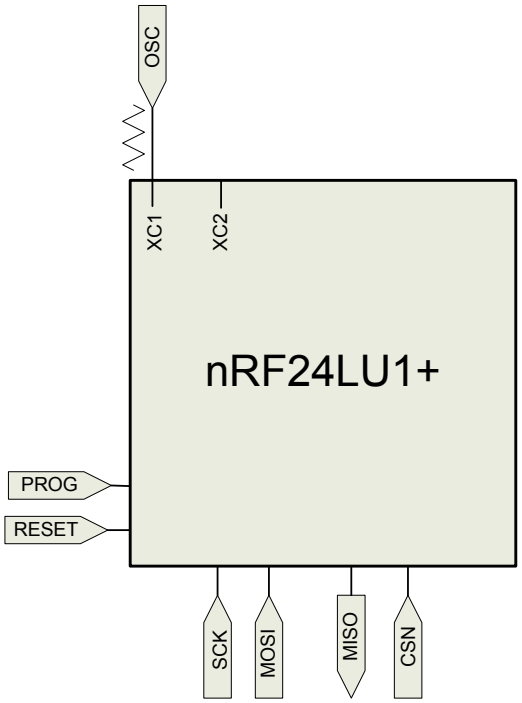


Figure 75. External clock source

- From a crystal between XC1 (pin 32) and XC2 (pin 31)

- See chapter 24 on page 176 for oscillator circuitry details.
- Make sure the socket solution does not add significant parasitic to the circuit.

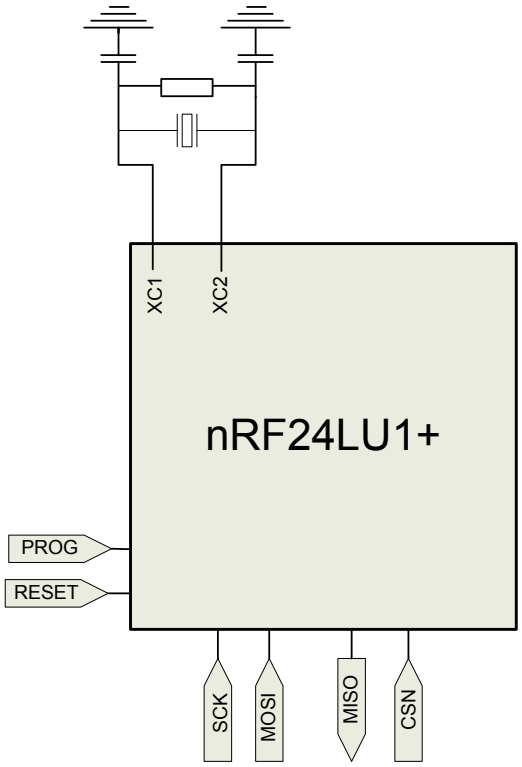


Figure 76. Clock source from a crystal between XC1 and XC2 pins

17.7.2.2 Power supply requirements

All VSS pins should be well connected to the adapter board, preferably using a ground plan. All VDD pins should be connected together and decoupled with at least three capacitors (two 10nF capacitors and one 100nF capacitor), to VSS. The **DEC2** pin should be decoupled with 33nF to VSS.

Using 3.3V supply (preferred)

Leave the **VBUS** pin open and connect a 3.3V +/- 5% power supply to the VDD pins.

17.7.2.3 Signal pin requirements

SPI Port

The pins **CSN**, **SCK** and **MOSI** must be driven by programmer pin electronics. All write operations to the FLASH are controlled by these three signals. The pin **MISO** is an output of the nRF24LU1+ that must be read to validate flash contents, and can be read to check status during write or erase operations.

The clock frequency of the SPI port is not correlated to the XC1 clock (it does not need to be synchronous). The SPI data signals have no defined relation to the XC1 clock (can be any phase). For more information see chapter 5 on page 19.

Control pins PROG and RESET

If power on sequence is clean (that is, nRF24LU1+ is well seated when power is ramped and power ramp is monotonous), the program and reset signals may be strapped to VDD. However, if possible, these signals should be controlled by pin drivers so that they are independent of power ramp quality.

17.7.3 In circuit programming over SPI

You can program a finished PCB with nRF24LU1+ that has all parts mounted. There are similar requirements to a standalone programmer with the following exceptions:

- All pins must be connected according to application requirements.
- **PROG** pin needs a pull-down on the PCB.
- **PROG** pin should be under control of the programmer over the edge of the PCB or through a pogo pin.
- **RESET** pin needs a pull-up on the PCB.
- **RESET** pin should be under control of the programmer over the edge of the PCB or through a pogo pin (to restart device after programming if required).
- The SPI input pins (**CSN**, **SCK**, **MOSI**) should be under control of the programmer over the edge of the PCB or through a pogo pin.
- The SPI output pin (**MISO**) should be readable by programmer over the edge of the PCB or through a pogo pin.
- The applications use of the SPI port pins should not conflict with the use of these pins as a SPI port.
- nRF24LU1+ can be powered effectively by a 5V connected to VBUS (over USB plug or pogo pin) or by a 3.3V +/- 5% connected to VDD over the edge of the PCB or through a pogo pin.

17.7.4 SPI programming sequences

The details of SPI timing are described in section 10.3 on page 102. With limit track length (and other loading on **MISO**) it is possible to operate the SPI up to 8 MHz. Reducing that to 4 MHz (or even 2 MHz) does not significantly impact the overall programming time.

The sequences of command and data in an SPI command are found in Figure 72. on page 146 and Figure 74. on page 146. In these figures only 1 byte of data is shown. Typically for read and write data transfers the block should be as long as possible (64 to 256 data bytes gives the best performance).

The typical production line sequence of commands is:

1. Pulse **RESET** pin low and return to high.
2. Pull **PROG** pin high and wait for 2ms.
3. Issue `WREN` command
4. Issue `ERASE_ALL` command
5. Wait for the `ERASE_ALL` command to finish (this takes 360,000 clock cycles (XC1) after the positive edge of CSN).
6. Repeat the following 128 times for 32 kB flash memory, (64 times for 16 kB):
   - Issue `WREN` command
   - Issue `PROGRAM` command followed by the next address and then the next 256 data bytes.
   - Wait until the `PROGRAM` command is finished (this is 256 + 1 times 365 clock cycles (XC1) after the positive edge of CSN).

7. Repeat the following 128 times for 32 kB flash memory:
   - Issue `READ` command followed by next address then read out 256 bytes on **MISO**
   - Compare read bytes against expected

The following are optional steps that update the InfoPage fields:

8. Issue `WFSR` to set `FSR.INFEN` bit to 1.

IF "Page address for start of protected area" is specified (not equal to 0xFF):

9. Issue SPI command WREN
10. Issue SPI command PROGRAM with address 0x0020 followed by the offset byte
11. Wait until PROGRAM command is finished

IF "Enable flash data memory" is specified:

12. Issue SPI command WREN
13. Issue SPI command PROGRAM with address 0x0021 followed by a byte that is not 0xFF
14. Wait until PROGRAM command has completed (this is 365 clock cycles (XC1) after the positive edge of CSN).

IF "Readback blocking for MainBlock" is specified:

15. Issue SPI command RDISMB
16. Wait until PROGRAM command has completed (this is 365 clock cycles (XC1) after the positive edge of CSN).

IF "Readback blocking for InfoPage" is specified:

17. Issue SPI command RDISIP
18. Wait until PROGRAM command has completed (This is 365 clock cycles (XC1) after the positive edge of CSN).

Note: The completion of the `ERASE_ALL` and/or the `PROGRAM` command may be ensured by waiting the specified amount of time, or alternatively repeatedly issuing `RDSR` commands until the `FSR.RDYN` bit reads back as 0.

17.7.4.1 Erasing and programming the InfoPage

Our devices have all user defined fields of the InfoPage pre-erased. The above programming sequence does not erase the InfoPage. If InfoPage needs to be erased due to re-programming of a part, the following steps must be added:

1. Issue `WFSR` to set `FSR.INFEN` bit to 1 and `FSR.WEN` bit to 1
2. Issue `ERASE_PAGE` 0

Note: To avoid losing the `CHIP_ID` (at address 0x0B to 0x0F), these bytes should be saved before erasing, and programmed back afterwards.

3. Wait until `ERASE_PAGE` command is finished (this will be 360,000 clock cycles (XC1) after the positive edge of CSN).

The InfoPage is now erased and ready for programming. If the flash MainBlock is read protected (`FSR.RDISMB`=1), the InfoPage can only be erased if the MainBlock has been erased. This means that

step 1 of this sequence must come after step 5 (ERASE_ALL) in the programming sequence in section 17.7.4 on page 153.

Note: The completion of the ERASE_PAGE and/or the PROGRAM command may be ensured by waiting the specified amount of time, or alternatively repeatedly issuing RDSR commands until the FSR.RDYN bit reads back as 0.

18 MDU – Multiply Divide Unit

The MDU – Multiplication Division Unit, is an on-chip arithmetic co-processor which enables the MCU to perform additional extended arithmetic operations like 32-bit division, 16-bit multiplication, shift and, normalize operations.

18.1 Features

The MDU is controlled by the SFR registers MD0 .. MD5 and ARCON.

18.2 Block diagram

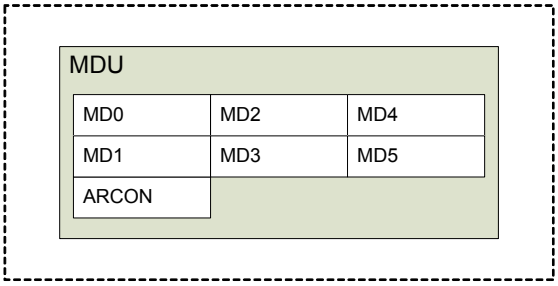


Figure 77. Block diagram of MDU

18.3 Functional description

All operations are unsigned integer operations. The MDU is handled by seven registers, which are memory mapped as Special Function Registers. The arithmetic unit allows concurrent operations to be performed independent of the MCU's activity.

Operands and results are stored in MD0..MD5 registers. The module is controlled by the ARCON register. Any calculation of the MDU overwrites its operands.

The MDU does not allow reentrant code and cannot be used in multiple threads of the main and interrupt routines at the same time. Use the NOMDU_R515 directive to disable MDU operation in possible conflicting functions.

18.4 SFR registers

The MD0 .. MD5 are registers used in the MDU operation.

Address	Register name
0xE9	MD0
0xEA	MD1
0xEB	MD2
0xEC	MD3
0xED	MD4
0xEE	MD5

Table 123. Multiplication/Division registers MD0..MD5

The ARCON register controls the operation of MDU and informs you about its current state.

Address	Reset value	Bit	Name	Description
0xEF	0x00	7	mdef	MDU Error flag MDEF. Indicates an improperly performed operation (when one of the arithmetic operations has been restarted or interrupted by a new operation).
		6	mdov	MDU Overflow flag MDOV. Overflow occurrence in the MDU operation.
		5	slr	Shift direction, 0: shift left, 1: shift right.
		4-0	sc	Shift counter. When set to '0's, normalize operation is selected. After normalization, the "sc.0" … "sc.4" contains the number of normalizing shifts performed. Shift operation is selected when at least one of these bits is set high. The number of shifts performed is determined by the number written to "sc.4".., "sc.0", where "sc.4" is the MSB.

Table 124. ARCON register

The operation of the MDU consists of the following phases:

18.4.1 Loading the MDx registers

The type of calculation the MDU has to perform is selected in accordance with the order in which the MDx registers are written.

Operation	32 bit/16 bit		16 bit / 16 bit		16 bit x 16 bit		Shift/normalize	
first write	MD0 (lsb) MD1 MD2 MD3 (msb)	Dividend	MD0 (lsb) MD1 (msb)	Dividend	MD0 (lsb) MD4 (lsb)	Num1 Num2	MD0 (lsb) MD1 MD2 MD3 (msb)	Number
last write	MD4 (lsb) MD5 (msb)	Ddivisor	MD4 (lsb) MD5 (msb)	Ddivisor	MD1 (msb) MD5 (msb)	Num1 Num2	ARCON	

Table 125. MDU registers write sequence

1. Write MD0 to start any operation.
2. Write operations, as shown in Table 125. to determine appropriate MDU operation.
3. Write (to MD5 or ARCON) starts selected operation.

The SFR Control detects some of the above sequences and passes control to the MDU. When a write access occurs to MD2 or MD3 between write accesses to MD0 and finally to MD5, then a 32/16 bit division is selected.

When a write access to MD4 or MD1 occurs before writing to MD5, then a 16/16 bit division or 16x16 bit multiplication is selected. Writing to MD4 selects 16/16 bit division and writing to MD1 selects 16x16 bit multiplication, that is, Num1 x Num2.

18.4.2 Executing calculation

During executing operation, the MDU works on its own in parallel with the MCU.

Operation	Number of clock cycles	
Division 32bit/16bit	17 clock cycles	
Division 16bit/16bit	9 clock cycles	
Multiplication	11 clock cycles	
Shift	min. 3 clock cycles (sc = 01h)	max 18 clock cycles (sc = 1Fh)
Normalize	min. 4 clock cycles (sc <- 01h)	max 19 clock cycles (sc <- 1Fh)

Table 126. MDU operations execution times

18.4.3 Reading the result from the MDx registers

Operation	32 bit/16 bit		16 bit / 16 bit		16 bit x 16 bit		Shift/normalize	
first read	MD0 (lsb) MD1 MD2 MD3 (msb)	Quotient	MD0 (lsb) MD1 (msb)	Quotient	MD0 (lsb) MD1 MD2	Product	MD0 (lsb) MD1 MD2	Number
last read	MD4 (lsb) MD5 (msb)	Remainder	MD4 (lsb) MD5 (msb)	Remainder	MD3 (msb)		MD3 (msb)	

Table 127. MDU registers read sequence

The Read out sequence of the first MDx registers is not critical but the last read (from MD5 - division and MD3 - multiplication, shift or normalize) determines the end of a whole calculation (end of phase three).

18.4.4 Normalizing

All leading zeroes of 32-bit integer variable stored in the `MD0` .. `MD3` registers are removed by shift left operations. The whole operation is completed when the MSB (Most Significant Bit) of `MD3` register contains a '1'. After normalizing, bits ARCON.4 (msb) .. ARCON.0 (lsb) contain the number of shift left operations that were done.

18.4.5 Shifting

In shift operation, 32-bit integer variable stored in the `MD0` ... `MD3` registers (the latter contains the most significant byte) is shifted left or right by a specified number of bits. The slr bit (ARCON.5) defines the shift direction and bits ARCON.4 ... ARCON.0 specify the shift count (which must not be 0). During shift operation, zeroes come into the left end of MD3 for shifting right or they come in the right end of the MD0 for shifting left.

18.4.6 The mdef flag

The mdef error flag (see Table 124. on page 157) indicates an improperly performed operation (when one of the arithmetic operations is restarted or interrupted by a new operation). The error flag mechanism is automatically enabled with the first write operation to MD0 and disabled with the final read instruction from MD3 (multiplication or shift/norm) or MD5 (division) in phase three.

The error flag is set when:

- If you write to MD0 .. MD5 and/or ARCON during phase two of MDU operation (restart or calculations interrupting).
- If any of the MDx registers are read during phase two of MDU operation when the error flag mechanism is enabled. In this case, the error flag is set but the calculation is not interrupted.

The error flag is reset only after read access to the ARCON register. The error flag is read only.

18.4.7 The mdov flag

The mdov overflow flag (see Table 124. on page 157) is set when one of the following conditions occurs:

- division by zero.
- multiplication with a result greater than 0000 FFFFh.
- start of normalizing if the most significant bit of MD3 is set (“md3.7” = ‘1’).

Any operation of the MDU that does not match the above conditions clears the overflow flag.

Note: The overflow flag is exclusively controlled by hardware, it cannot be written.

19 Watchdog and wakeup functions

In order to achieve the lowest possible average current consumption, the processor clock can be stopped under firmware control. Operation can be resumed (wakeup) on external events like toggling of GPIO pins or from the internal RTC wakeup timer, USB or, the RF-module, see chapter 20 on page 165 for details.

In addition, a programmable watchdog timer can be enabled to reset the system if the software hangs.

19.1 Features

- 32 kHz operation
- Programmable 8-bit resoulution
- 16-bit range Watchdog
- Watchdog disabled (reset) only by a system reset
- 24-bit range wakeup timer
- Timer is a possible interrupt source
- Timer reload can be signalled on GPIO

19.2 Block diagram

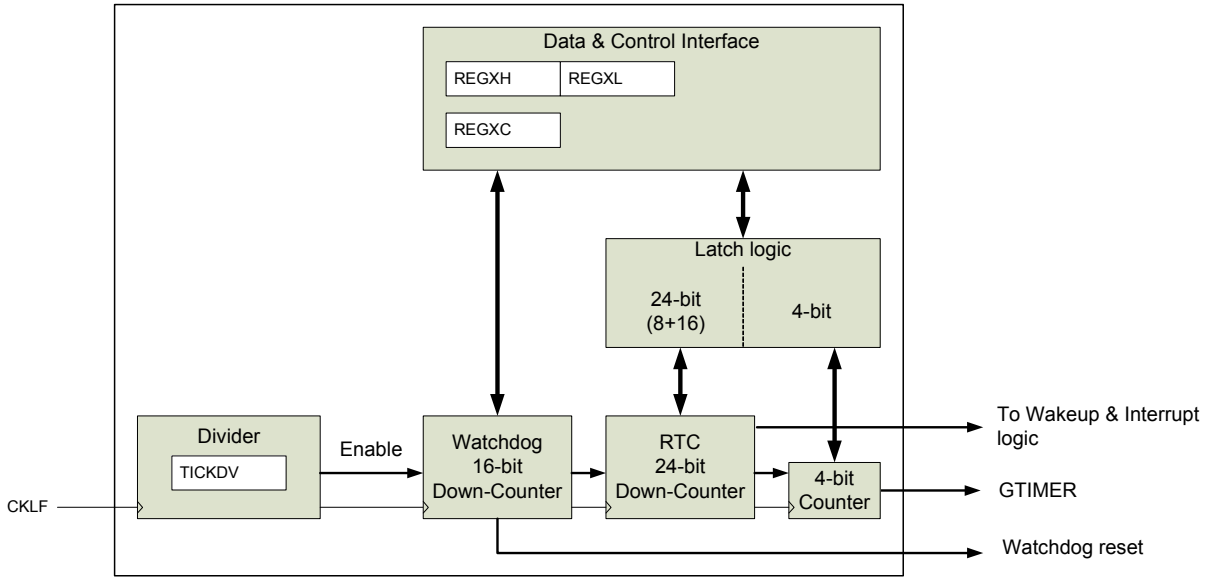


Figure 78. Watchdog and wakeup functions block diagram

19.3 Functional description

19.3.1 The Low Frequency Clock (CKLF)

CKLF frequency f_{CKLF} is 32000 Hz (derived from the crystal oscillator[1] and is used for wakeup functions and the Watchdog. This clock is always running.

19.3.2 Tick calibration

The tick is an interval (in CKLF periods) that determines the resolution of the watchdog and the RTC wakeup timer. By default the tick is set to 125 µs (4 CKLF cycles). The programmable range is from 31.25 µs to 8 ms. The tick is as accurate as the 32 kHz source.

The tick is controlled by the TICKDV register.

Addr	Reset value	bit	R/W	Function
0xB5	0x03	7:0	RW	Divider that is used in generating tick from CKLF frequency. Ttick = (1 + TICKDV) / f_{CKLF}.

Table 128. TICKDV register

19.3.3 RTC wakeup timer

The RTC is a simple 24 bit down counter that produces an optional interrupt and reloads automatically when the count reaches zero. This process is initially disabled, and is enabled with the first write to the lower 16 bit of the timer latch (WRTCLAT). Writing the lower 16 bits of the timer latch is always followed by a reload of the counter. Only write the upper 8 bit of the timer latch when the timer is disabled, see Table 130. on page 164.

The RTC counter may be disabled again by writing a disable opcode to the control register (WRTCDIS). Both the latch and the counter value may be read by giving the respective codes in the control register, see the description in Table 129. on page 163 and Table 130. on page 164.

The RTC counter is used for a wakeup sometime in the future (a relative time wakeup call). If ‘N’ is written to the counter, the first wakeup happens between ‘N+1’ and ‘N+2’ “tick” from the completion of the write. From then on a new wakeup is issued every “N+1” "tick" until the unit is disabled or another value is written to the latch.

The wakeup timer is one of the sources that can generate a WU interrupt (see Table 140. on page 173) to the MCU. You may poll the flag or enable the interrupt. If the MCU is in a power down or standby state, the wakeup forces the device to exit power down or standby regardless of the state of the interrupt enable.

The MCU system does not provide any “absolute time functions”. Absolute time functions can be handled in software since the RAM is continuously powered even when in sleep mode.

1. f_{CKLF} is 1/500 of oscillator frequency.

19.3.4 Programmable GPIO wakeup function

All pins in port 0 can be used as wakeup signals for the MCU system. The device can be programmed to react on rising, falling or, both edges of each pin individually. Additionally, each pin is equipped with a programmable filter that is used for glitch suppression.

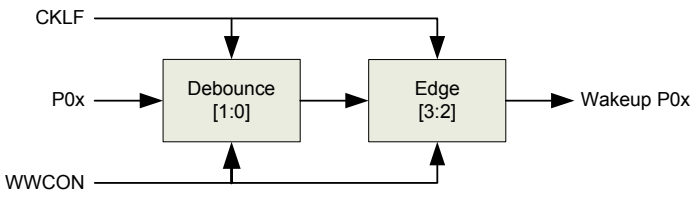


Figure 79. Wakeup filter, each pin for GPIO wakeup function

The debounce logic acts as a low pass filter. The input has to be stable for the number of clock pulses that are given (in WGTIMER) to appear on the output. Edge triggers on positive, negative, or both edges. The edge delay is 2 clock cycles. Please see Table 130. on page 164 and Table 131. on page 164 for filter configuration.

19.3.5 Watchdog

The watchdog is activated on the first write to its control register REGXC. It cannot be disabled by any other means than a reset.

The watchdog register is loaded by writing a 16-bit value (number of ticks) to the two 8-bit data registers (REGXH and REGXL) and then writing the correct opcode to the control register. The watchdog counts down towards 0 and when 0 is reached the complete MCU is reset.

To avoid the reset, the software must regularly load new values into the watchdog register.

19.3.6 Programming interface to watchdog and wakeup functions

Figure 80. on page 163 shows how the blocks that are always active are connected to the MCU.

RTC timer GPIO wakeup and Watchdog are controlled through three SFRs. The three registers, REGXH, REGXL and, REGXC, are used to interface the blocks running on the slow CKLF clock. The 16-bit register REGXH:REGXL can be written or read as two bytes from the MCU.

Typical sequences are:

```
Write:Write REGXH, Write REGXL, Write REGXC
Read: Write REGXC, Read REGXH, Read REGXL
```

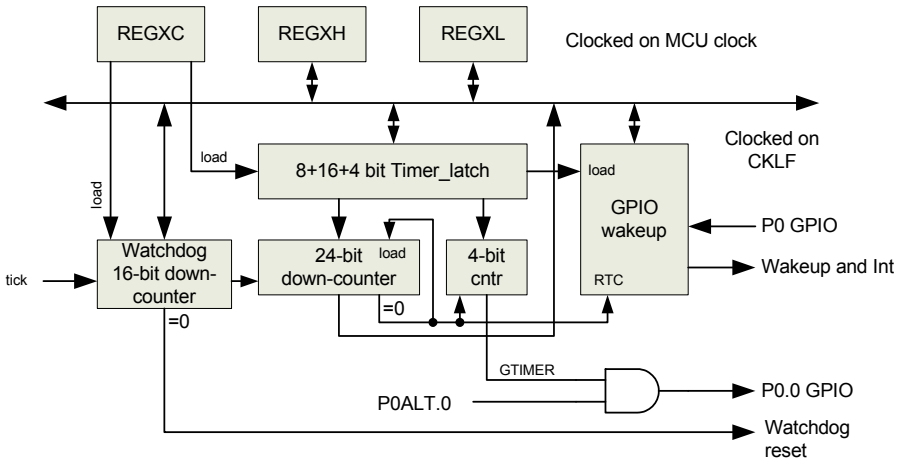


Figure 80. Block diagram of wakeup and watchdog functions

Table 129. on page 163 describes the functions of the SFR registers that control those blocks, and Table 130. on page 164 explains the contents of the individual control registers for watchdog and wakeup functions.

Addr	Reset value	bit	R/W	Init	Name	Function
0xAB	0x00	7:0	RW	0x00	REGXH	Most significant byte of 16-bit data register
0xAC	0x00	7:0	RW	0x00	REGXL	Least significant byte of 16-bit data register
0xAD	0x00	7:5 4 3 2:0	- R RW RW	0x00	REGXC	Control register for 16 bit data register Not used Status of last REGXC write access 0: finished, 1: not finished 0: read, 1: write; see R/W column in Table 130. on page 164. Indirect address, see the far left column in Table 130. on page 164.

Table 129. REGXH, REGXL and REGXC registers

Indirect Address	Data register Bit	R/W[a]	Name	Function
000	15:0	R	RWD	Watchdog register (count)
	15:0	W	WWD	Watchdog register (count)
001	15:8 7:0	R R	RGTIMER	MSB part of RTC counter MSB part of RTC latch
	15:12 11:8 7:0	- W W	WGTIMER	Not used GTIMER latch MSB part of RTC latch
010	15:0	R	RRTCLAT	Least significant part of RTC latch
	15:0	W	WRTCLAT	Least significant part of RTC latch
011	15:0	R	RRTC	RTC counter value
	-	W	WRTCDIS	Disable RTC (data not used)

Indirect Address	Data register Bit	R/W[a]	Name	Function
100 100	15:9 8 5:0	- R R	RWSTA0	Not used Wakeup status for RTC timer Wakeup status for pins P05-P00. RWSTA0 is automatically cleared after read.
	15:14 13:12 11:10 9:8 7:6 5:4 3:2 1:0	W W W W W W W W	WWCON0	Edge selection of P03 Debounce filter for P03 Edge selection of P02 Debounce filter for P02 Edge selection of P01 Debounce filter for P01 Edge selection of P00 Debounce filter for P00, see Table 131. on page 164.
101	15:9 8 7:0	- R R	RWSTA1	Identical to RWSTA0 above
	15:8 7:6 5:4 3:2 1:0	- W W W W	WWCON1	Not used Edge selection of P05 Debounce filter for P05 Edge selection of P04 Debounce filter for P04, see Table 131. on page 164.
110	15:0	-	-	Reserved, do not use
111	15:0	-	-	Reserved, do not use

a. REGXC bit-3 selects between R(ead) and W(rite) operation

Table 130. Indirect addresses and functions

Debounce filter selection		Edge selection	
Code	**Number of clock pulses**	**Code**	**positive/negative trigger**
00	0	00	Off
01	2	01	Positive
10	8	10	Negative
11	64	11	Both

Table 131. GPIO wakeup filter configuration, WWCON

20 Power management

The nRF24LU1+ Power Management function controls the power dissipation through the administration of modes of operation and by controlling clock frequencies.

20.1 Features

- Supports low power modes for MCU, RF Tranceiver, USB and 48 MHz PLL
- Programmable MCU clock frequency from 64 kHz to 16 MHz
- Multi-source MCU wakeup
- Watchdog and wakeup functionality running in low power mode

20.2 Block diagram

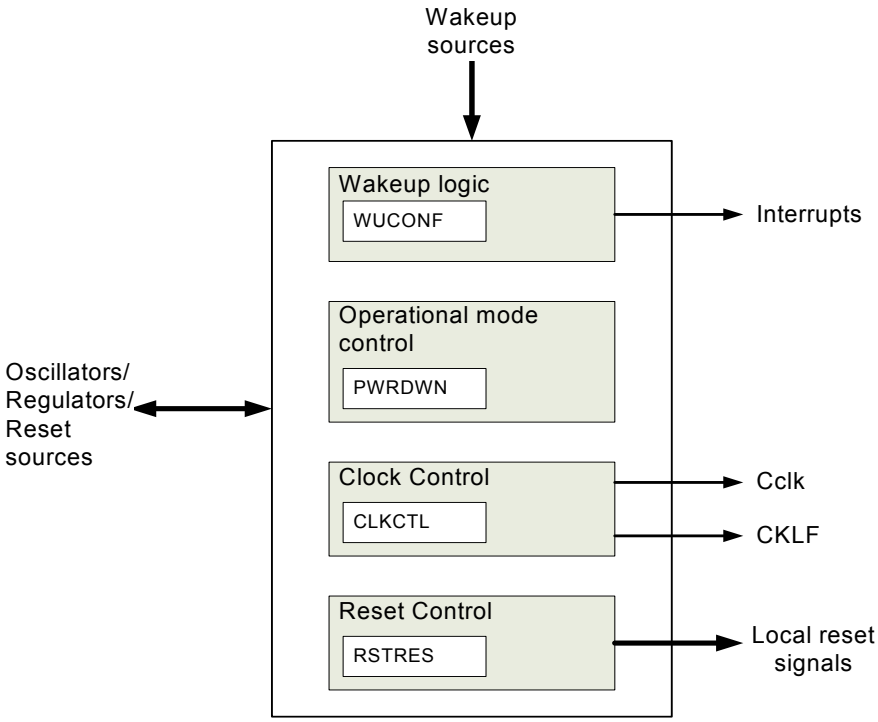


Figure 81. Power management block diagram

20.3 Modes of operation

There are four main power consuming functions on the chip. These can be controlled on and off in different ways depending on the required functionality after the start-up/reset sequence is ended.

These functions are:

- MCU
  - states of operation: active and standby
  - active at the end of the reset sequence
  - Set to standby by software (write **PWRDWN** register = 0x01)
  - Set to active by wakeup sources:
    Interrupt from USB
    Interrupt from RF Transceiver
    Interrupt from external pin
    Interrupt from on-chip RTC

- RF Transceiver
  - states of operation: **power down**, **standby** and **active** (TX or RX)
  - pwrdwn at the end of the reset sequence
  - Set to **standby**, **active** or **power down** by software, see section 6.3.1 on page 27

- USB
  - states of operation: active and suspend
  - active at the end of the reset sequence
  - Set to suspend by software (write USBSLP register = 0x01)
  - Set to active by software or by wakeup from USB host (through the USB bus)

- PLL
  - states of operation: on and off
  - on at the end of the reset sequence
  - Set to on or off by hardware with one exception:
    if the USB is in suspend the PLL may be controlled by software
    (Enable PLL, bit 7 in the CLKCTL register)

Table 132. on page 166 summarizes the available modes of operation after the reset sequence is ended:

- **PROG** is an external pin on the nRF24LU1+.
- RF Transceiver, USB, MCU and PLL represent the functions defined above.

PROG	RF Transceiver	USB	MCU	PLL	Comment
1	-	-	-	-	Flash programming mode via SPI
0	standby	suspend	standby	OFF	
0	standby	suspend	active	software	
0	standby	active	standby	ON	
0	standby	active	active	ON	
0	active	suspend	standby	OFF	
0	active	suspend	active	software	
0	active	active	standby	ON	
0	active	active	active	ON	

Table 132. nRF24LU1+ modes of operation

In nRF24LU1+ the 16 MHz oscillator is always running. An internal PLL can be enabled that multiplies the 16 MHz by three to get an internal 48 MHz clock. This clock is required for USB operation.

The internal 32.000 kHz clock (CKLF) is generated from the 16 MHz oscillator.

To save power when the USB is suspended, the PLL can be turned off, and the clock frequency to the MCU can be reduced. This reduces power consumption, but also reduces performance.

To further reduce power, the MCU clock can be stopped using the `PWRDWN` register. In the `PCON` register stop and idle modes can be selected, but since their effect on power consumption is minor use `PWRDWN`.

The following various internal and external events can resume the MCU clock:

- Interrupt from RF Transceiver, `rfirq`
- Interrupt from USB
- Interrupt from RTC timer or GPIO-pins (see chapter 19 on page 160)

The `WUCONF` register controls how these events are handled.

20.4 Functional description

20.4.1 Clock control – CLKCTL

Addr	Reset value	Bit	R/W	Function
0xA3	0x80	7	RW	Enable PLL, 1: PLL on, 0: PLL off
		6:4	RW	Set Cclk (MCU clock) frequency when PLL is ON 000: 16 MHz 001: 12 MHz 010: 8 MHz 011: 4 MHz 100: 1.6 MHz Other combinations: reserved.
		3:2	-	Not used
		1:0	RW	Set Cclk (MCU clock) frequency when PLL is OFF 00: 4 MHz 01: 1.6 MHz 10: 320 kHz 11: 64 kHz

Table 133. CLKCTL register

20.4.2 Power down control – PWRDWN

Addr	Reset value	Bit	R/W	Function
0xA4	0x00	7:4	-	Not used
		3	R	Read CKLF clock (32 kHz clock, always running)
		2:0	W	Set MCU to standby if different from 000

Note: Any pending interrupt flags in `IRCON` must be cleared before setting MCU to standby.

Table 134.PWRDWN register

20.4.3 Reset result – RSTRES

The following three reset sources initiate the same reset/start-up sequence:

- Reset from the on-chip reset generator.
- Reset from pin.
- Reset generated from the on-chip watchdog function.

The RSTRES register stores the reset cause:

Addr	Reset value	Bit	R/W	Function
0xB1	0x00	7:1	-	Not used
		0	R	Reset cause, 1: Watchdog, 0: other

Table 135. RSTRES register

20.4.4 Wakeup configuration register – WUCONF

Addr	Reset value	Bit	R/W	Function
0xA5	0x00	7:6	RW	00: Enable wakeup on RFIRQ, if IEN1.1=1 01: Reserved, not used 10: Enable wakeup on RFIRQ, regardless of IEN1.1 11: Ignore RFIRQ
		5:4	RW	00: Enable wakeup on WU, if IEN1.5=1[a] 01: Reserved, not used 10: Enable wakeup on WU, regardless of IEN1.5 11: Ignore WU
		3:2	RW	00: Enable wakeup on USBIRQ, if IEN1.4=1 01: Reserved, not used 10: Enable wakeup on USBIRQ, regardless of IEN1.4 11: Ignore USBIRQ
		1:0	RW	00: Enable wakeup on USBWU, if IEN1.3=1 01: Reserved, not used 10: Enable wakeup on USBWU, regardless of IEN1.3 11: Ignore USBWU

a. WU is generated as described in sections 19.3.3 and 19.3.4

Note: IRCON flag will be set upon wakeup, even if interrupt is not enabled in `IEN1`.

Table 136.WUCONF register

20.4.5 Power control register - PCON

The PCON register is used to control the Power Down Modes (IDLE, STOP), the Program Memory Write Mode and Serial Port 0 baud rate doubler.

Address	Reset value	Bit	Name	Description
0x87	0x00	7	smod	Serial Port 0 baud rate select, see Table 89. on page 114 (baud rate doubler).
		6	gf3	General purpose flag 3
		5	gf2	General purpose flag 2
		4	pmw	Program memory write mode. Setting this bit enables the program memory write mode.
		3	gf1	General purpose flag 1
		2	gf0	General purpose flag 0
		1	stop	Stop mode control. Setting this bit activates the Stop Mode. Always read as 0.
		0	idle	Idle mode control. Setting this bit activates the Idle Mode. Always read as 0.

Table 137. PCON register

21 Power supply supervisor

The power supply supervisor initializes the system at power-on, provides an early warning of impending power failure, and puts the system in reset state if the supply voltage is too low for safe operation.

21.1 Features

- Power-on reset
- Brown-out reset
- Early power-fail warning with some hardware protection of data in flash memory

21.2 Functional description

21.2.1 Power-on reset

A Power-on reset generator initializes the system at power-on. The system is held in reset state until VDD has reached around 2.7V or higher.

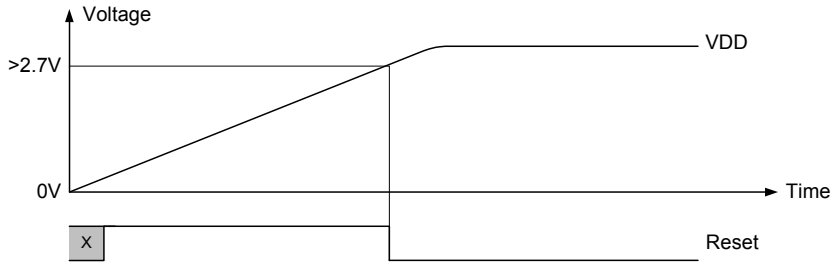


Figure 82. Power-on reset

21.2.2 Brown-out detection

If supply VBUS or VDD drops below around 2.7V (which is outside the operational specification), a power-fail detection signal goes active. If the supply goes below around 1.8V, a brown-out reset signal goes on and the chip is reset. The supply must rise above approximately 2.7V again before the reset signal is released. The power-fail signal is used to prevent flash memory write or erase at low voltage, see section 17.4 on page 139.

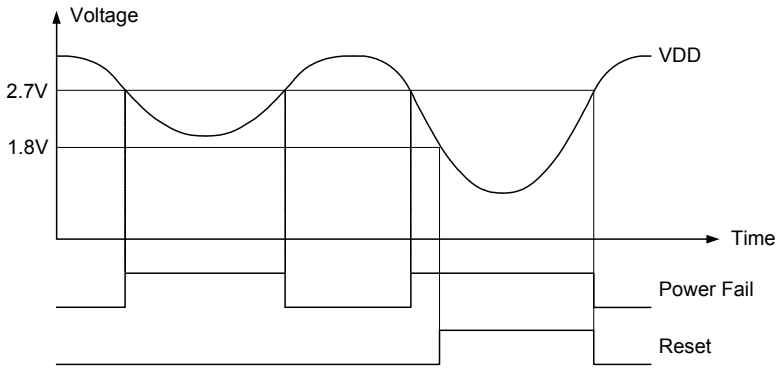


Figure 83. Brown-out detection

22 Interrupts

nRF24LU1+ has an advanced interrupt controller with 15 sources, as shown in Figure 84.. The unit manages dynamic program sequencing based upon important real-time events as signalled from timers, the RF Transceiver, the USB interface or pin activity.

22.1 Features

- Interrupt controller with 15 sources and 4 priority levels
- Interrupt request flags available
- Interrupt from pin with selectable polarity

22.2 Block diagram

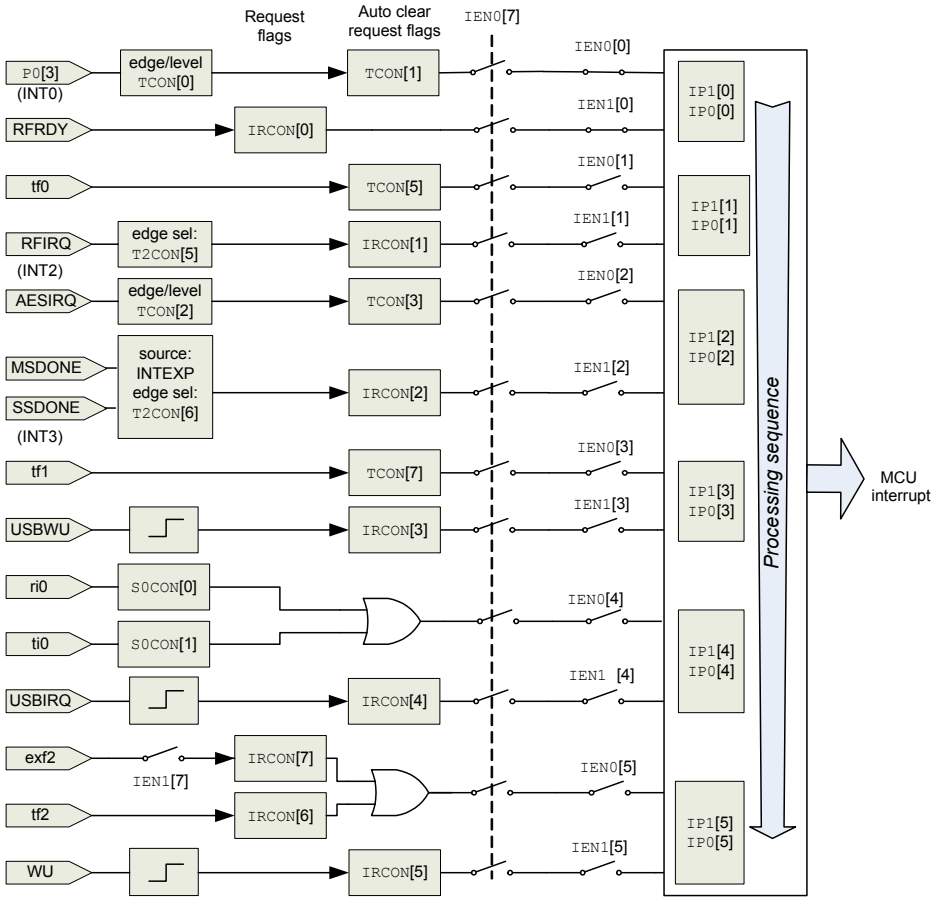


Figure 84. nRF24LU1+ interrupt structure

22.3 Functional description

When an enabled interrupt occurs, the MCU vectors to the address of the interrupt service routine (ISR) associated with that interrupt, as listed in Table 138. on page 172. The MCU executes the ISR to completion unless another interrupt of higher priority occurs.

Source	Vector	Polarity	Description
P0.3	0x0003	low/fall	External pin P0.3
tf0	0x000B	high	Timer 0 interrupt
AESIRQ	0x0013	low/fall	AES ready interrupt
tf1	0x001B	high	Timer 1 interrupt
ri0	0x0023	high	Serial channel receive interrupt
ti0	0x0023	high	Serial channel transmit interrupt
tf2	0x002B	high	Timer 2 interrupt
exf2	0x002B	High	Timer 2 external event (pin P0.5)
RFRDY	0x0043	high	RF SPI ready
RFIRQ	0x004B	fall/rise	RF IRQ
MSDONE	0x0053	fall/rise	Master SPI transaction completed
SSDONE	0x0053	fall/rise	Slave SPI transaction completed
USBWU	0x005B	rise	USB wakeup interrupt
USBIRQ	0x0063	rise	USB interrupt
WU	0x006B	rise	Internal Wakeup interrupt

Table 138. nRF24LU1+ interrupt sources

22.4 SFR registers

Various SFR registers are used to control and prioritize between different interrupts.

The `IRCON`, `SCON`, `IP0`, `IP1`, `IEN0`, `IEN1` and `INTEXP` are described in this section. In addition, a description of the TCON and T2CON registers is found in chapter 11 on page 103.

22.4.1 Interrupt enable 0 register – IEN0

The `IEN0` register is responsible for global interrupt system enabling/disabling as well as Timer0, 1 and 2, Port 0 and Serial Port individual interrupts enabling/disabling.

Address	Reset value	Bit	Description
0xA8	0x00	7	1: Enable interrupts. 0: all interrupts are disabled.
		6	Not used.
		5	1: Enable Timer2 interrupt.
		4	1: Enable Serial Port interrupt.
		3	1: Enable Timer1 overflow interrupt
		2	1: Enable pin P0.4 interrupt.
		1	1: Enable Timer0 overflow interrupt.
		0	1: Enable pin P0.3 interrupt.

Table 139. IEN0 register

22.4.2 Interrupt enable 1 register – IEN1

The `IEN1` register is responsible for RF, SPI, USB and Timer 2 interrupts.

Address	Reset value	Bit	Description
0xB8	0x00	7	1: Enable Timer2 external reload interrupt
		6	Not used
		5	1: Wakeup interrupt enable
		4	1: USB interrupt enable
		3	1: USB wakeup interrupt enable
		2	1: Master or Slave SPI ready interrupt enable
		1	1: RF interrupt enable
		0	1: RF SPI ready enable

Table 140. IEN1 register

Master SPI and Slave SPI share the same interrupt line.

Address	Reset value	Bit	Function
0xA6	0x01	7:2	Not used
		1	1: Enable Master SPI interrupt
		0	1: Enable Slave SPI interrupt

Table 141. INTEXP register.

22.4.3 Interrupt priority registers – IP0, IP1

The 14 interrupt sources are grouped into six priority groups. For each of the groups, one of four priority levels can be selected. It is achieved by setting appropriate values in `IP0` and `IP1` registers.

The contents of the Interrupt Priority Registers define the priority levels for each interrupt source according to the tables below.

Address	Reset value	Bit	Description
0xA9	0x00	7:6	Not used.
		5:0	Interrupt priority. Each bit together with corresponding bit from `IP1` register specifies the priority level of the respective interrupt priority group.

Table 142. IP0 register

Address	Reset value	Bit	Description
0xB9	0x00	7:6	Not used.
		5:0	Interrupt priority. Each bit together with corresponding bit from `IP0` register specifies the priority level of the respective interrupt priority group.

Table 143. IP1 register

Group	Interrupt bits	Priority groups		
0	ip1.0, ip0.0	P0.3 interrupt	RF interrupt	
1	ip1.1, ip0.1	Timer 0 interrupt	RF SPI interrupt	
2	ip1.2, ip0.2	P0.4 interrupt	Master SPI	Slave SPI
3	ip1.3, ip0.3	Timer 1 interrupt	USB wakeup	
4	ip1.4, ip0.4	Serial port receive	Serial port transmit	USB interrupt
5	ip1.5, ip0.5	Timer 2 interrupt	Wakeup interrupt	

Table 144. Priority groups

ip1.x	ip0.x	Priority level
0	0	Level 0 (lowest)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (highest)

Table 145. Priority levels (x is the number of priority group)

22.4.4 Interrupt request control registers – IRCON

The `IRCON` register contains Timer 2, SPI, RF, USB and wakeup interrupt request flags.

Address	Reset value	Flag	Bit	Auto clear[a]	Description
0xC0	0x00	exf2	7	-	Timer 2 external reload flag
		tf2	6	-	Timer 2 overflow flag
		WU	5	Yes	Wakeup interrupt flag
		USBIRQ	4	Yes	USB interrupt flag
		USBWU	3	Yes	USB wakeup interrupt flag
		M- or S-DONE	2	Yes	Master or Slave SPI interrupt flag
		RFIRQ	1	Yes	RF interrupt flag
		RFRDY	0	-	RF SPI interrupt flag

a. Auto clear means that the flag is cleared by hardware automatically when the corresponding service routine is vectored.

Table 146. IRCON register

23 HW debugger support

The nRF24LU1+ has the following on-chip hardware debug support for a JTAG debugger:

- nRFProbe hardware debugger from Nordic Semiconductor.
- System Navigator from First Silicon Solutions (www.fs2.com).

These debug modules are available on device pins OCITO, OCTMS, OCITDO,OCITDI, OCITCK when enabled in the flash InfoPage. The HW debug features can be interfaced through USB to a PC and utilized in the Keil Integrated Development Environment (IDE) by running nRFProbe found in the nRFgo development kits or dedicated HW from First Silicon Solutions.

23.1 Features

- Read/write all processor registers, SFR, program and data memory.
- Go/halt processor run control.
- Single step by assembly and C source instruction.
- Four independent HW execution breakpoints.
- Driver software for Keil µVision debugger interface.

The features listed below are for the Keil µVision debugger only:

- Load binary, Intel Hex or OMF51 file formats.
- Symbolic debug.
- Load symbols, including code, variables and variable types.
- Support C and assembly source code.
- Source window can display C source and mixed mode.
- Source window provides execution control; go, halt; goto cursor; step over/into call.
- Source window can set or clear software and hardware breakpoints.

23.2 Functional description

The JTAG debug interface is enabled by writing (through the flash SPI slave interface) to address 0x24 in the InfoPage. Any byte value other than 0xFF enables debug. The Flash Status Register (FSR bit 7, 17.3.6 on page 139) shows the current status of the interface.

The GPIO allocated in debug mode is summarized in Table 147.

OCITO	P0.4
OCITDO	P0.3
OCITDI	P0.2
OCITMS	P0.1
OCITCK	P0.0

Table 147. HW debug physical interface for nRF24LU1+

Note: A pull-up on OCITCK is required for the MCU to run (in debug mode) without the system navigator cable plugged in.

A separate "Trigger Out" is available on the OCITO pin. This output can be activated when certain address and data combinations occur.

24 Peripheral information

This chapter describes peripheral circuitry and PCB layout requirements that are important for achieving optimum RF performance from the nRF24LU1+.

24.1 Antenna output

The ANT1 and ANT2 output pins provide a balanced RF output to the antenna. The pins must have a DC path to VDD_PA, either through a RF choke or through the center point in a balanced dipole antenna. A load of 15 Ω+j88 Ω is recommended for maximum output power (0dBm). Lower load impedance (for instance 50 Ω) can be obtained by fitting a simple matching network between the load and ANT1 and ANT2. A recommended matching network for 50Ω load impedance is illustrated in Chapter 25 on page 178.

24.2 Crystal oscillator

A crystal being used with the nRF24LU1+ must fulfil the specifications given in Table 10. on page 24.

You must use a crystal with a low load capacitance specification to achieve a crystal oscillator solution with low power consumption and fast start-up time. A lower C0 also gives lower current consumption and faster start-up time, but may increase the cost of the crystal. Typically C0=1.5pF at a crystal specified for C0max=7.0pF.

The crystal load capacitance, CL, is given by:

$C_L = \frac{C_1' \cdot C_2'}{C_1' + C_2'}$, where C1' = C1 + CPCB1 +CI1 and C2' = C2 + CPCB2 + CI2

C1 and C2 are SMD capacitors as shown in the application schematics, see Chapter 25 on page 178. CPCB1 and CPCB2 are the layout parasitic on the circuit board. CI1 and CI2 are the capacitance seen into the XC1 and XC2 pins respectively; the value is typically 1pF for each of these pins.

24.3 PCB layout and decoupling guidelines

A well designed PCB is necessary to achieve good RF performance. A poor layout can lead to loss of performance or functionality. A fully qualified RF-layout for the nRF24LU1+ and its surrounding components, including matching networks, can be downloaded from www.nordicsemi.no.

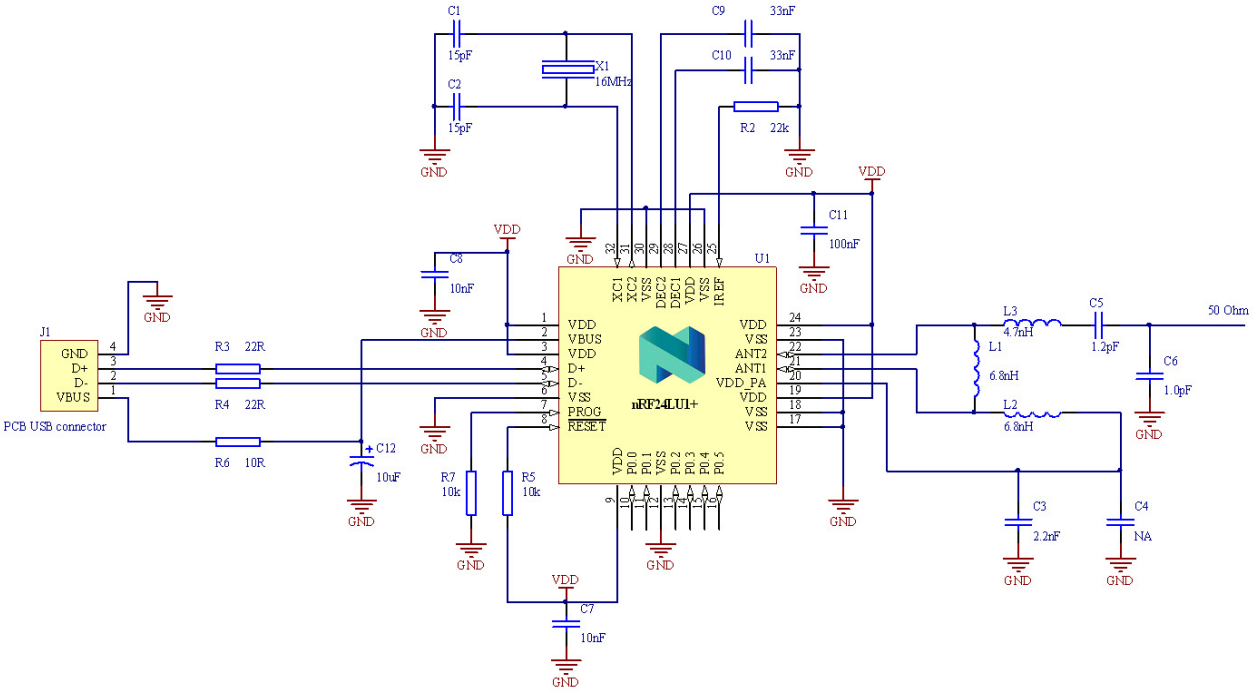
A PCB with a minimum of two layers including a ground plane is recommended for optimum performance. The nRF24LU1+ DC supply voltage should be decoupled as close as possible to the VDD pins with high performance RF capacitors.See the schematics layout in Chapter 25 on page 178 for recommended decoupling capacitor values. The nRF24LU1+ supply voltage should be filtered and routed separately from the supply voltages of any digital circuitry.

Long power supply lines on the PCB should be avoided. All device grounds, VDD connections and VDD bypass capacitors must be connected as close as possible to the nRF24LU1+ IC. For a PCB with a topside RF ground plane, the VSS pins should be connected directly to the ground plane. For a PCB with a bottom ground plane, the best technique is to have via holes as close as possible to the VSS pads. A minimum of one via hole should be used for each VSS pin.

Full swing digital data or control signals should not be routed close to the crystal or the power supply lines. The exposed die attach pad is a ground pad connected to the IC substrate die ground and is intentionally not used in our layouts. It is recommended to keep it unconnected.

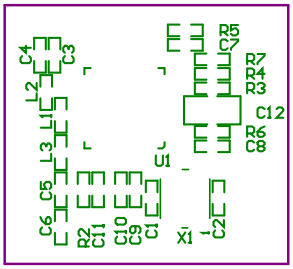
25 Application example

25.1 Schematics



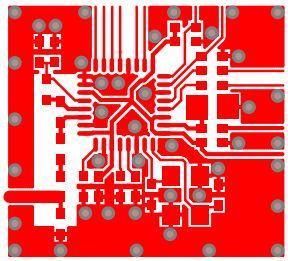
25.2 Layout

A double sided FR-4 board of 1.6 mm thickness is used. This PCB has a ground plane on the bottom layer. There are ground areas on the component side of the board to ensure sufficient grounding of critical components. A large number of via holes connect the top layer to ground areas to the bottom layer ground plane.

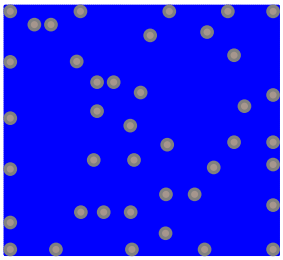


Top silk screen

No components in bottom layer



Top view



Bottom view

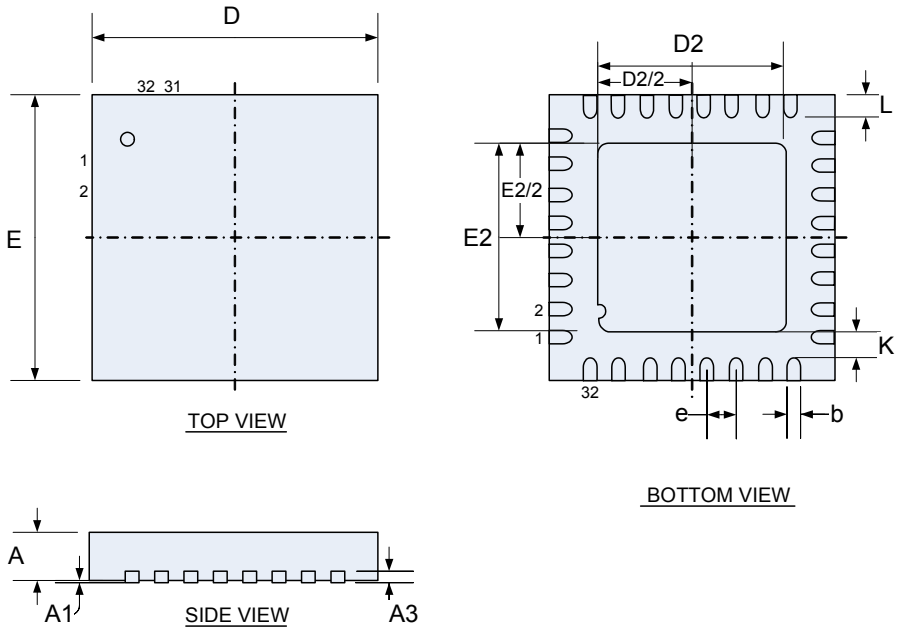
25.3 Bill Of Materials (BOM)

Designator	Value	Footprint	Comment
C1	15pF	0402	NP0 +/-2%
C2	15pF	0402	NP0 +/-2%
C3	2.2nF	0402	X7R +/-10%
C4	Not mounted	0402	
C5	1.2pF	0402	NP0 +/-0.1pF
C6	1.0pF	0402	NP0 +/-0.1pF
C7	10nF	0402	X7R +/-10%
C8	10nF	0402	X7R +/-10%
C9	33nF	0402	X7R +/-10%
C10	33nF	0402	X7R +/-10%
C11	100nF	0402	X7R +/-10%
C12	10uF	0805	X5R +/-10%
L1	6.8nH	0402	Chip inductor +/-5%
L2	6.8nH	0402	Chip inductor +/-5%
L3	4.7nH	0402	Chip inductor +/-5%
R2	22k	0402	1%
R3	22R	0402	1%
R4	22R	0402	1%
R5	10k	0402	1%
R6	10R	0402	1%
R7	10k	0402	1%
U1	nRF24LU1+	QFN32L/5x5	nRF24LU1+
X1	16 MHz	BT-XTAL	16 MHz, CL=9pF +/-60ppm

Table 148. Bill Of Materials

26 Mechanical specifications

nRF24LU1+ is packaged in a QFN32 5 x 5 x 0.85 mm, 0.5 mm pitch.



Package	A	A1	A3	b	D, E	D2, E2	e	K	L	
QFN32	0.80	0.00		0.18		3.20		0.20	0.35	Min.
	0.85	0.02	0.20	0.23	5	3.30	0.5		0.40	Typ.
	0.90	0.05		0.30		3.40			0.45	Max

Table 149. QFN32 dimensions in mm (Bold dimension denotes BSC)

27 Ordering information

27.1 Package marking

nRF24LU1P-F32 option:

n	R	F		B	X
2	4	L	U	1	P
Y	Y	W	W	L	L

nRF24LU1P-F16 option:

n	R	F		B	X
L	U	1	P	1	6
Y	Y	W	W	L	L

27.1.1 Abbreviations

Abbreviation	Definition
LU1P16	Product number, F16 option
24LU1P	Product number, F32 option
B	Build Code, that is, unique code for production sites, package type and, test platform.
X	"X" grade, that is, Engineering Samples (optional).
YY	Two digit Year number
WW	Two digit week number
LL	Two letter wafer lot number code

Table 150. Abbreviations

27.2 Product options

27.2.1 RF silicon

Ordering code	Flash memory size	Package	Container	MOQ[a]
nRF24LU1P-F32Q32-T	32 kB	5x5mm 32-pin QFN, lead free (green)	Tray	490
nRF24LU1P-F32Q32-R7	32 kB	5x5mm 32-pin QFN, lead free (green)	7” reel	1500
nRF24LU1P-F32Q32-R	32 kB	5x5mm 32-pin QFN, lead free (green)	13” reel	4000
nRF24LU1P-F32Q32-SAMPLE	32 kB	5x5mm 32-pin QFN, lead free (green)	Sample box	5
nRF24LU1P-F16Q32-T	16 kB	5x5mm 32-pin QFN, lead free (green)	Tray	490
nRF24LU1P-F16Q32-R7	16 kB	5x5mm 32-pin QFN, lead free (green)	7” reel	1500
nRF24LU1P-F16Q32-R	16 kB	5x5mm 32-pin QFN, lead free (green)	13” reel	4000
nRF24LU1P-F16Q32-SAMPLE	16 kB	5x5mm 32-pin QFN, lead free (green)	Sample box	5

a. Minimum Order Quantity

Table 151. nRF24LU1+ RF silicon options

27.2.2 Development tools

Type Number	Description
nRF24LU1P-FxxQ32-DK	nRF24LU1+ Development kit
nRF6700	nRFgo Starter Kit
nRF6704	nRFgo nRF24LU1P Flash/OTP Programming Adapter Kit (requires nRFgo Starter Kit)

Table 152. nRF24LU1+ solution options

28 Glossary of terms

Term	Description
ACC	Accumulator
ACK	Acknowledgement
ART	Auto Re-Transmit
BSC	Basic Spacing between Centers
Cclk	MCU Clock
CRC	Cyclic Redundancy Check
CSN	Chip Select NOT
DPS	Data Pointer Select register
ESB	Enhanced ShockBurst™
FCR	Flash Command Register
FPCR	Flash Protect Config Register
FSR	Flash Status Register
GFSK	Gaussian Frequency Shift Keying
HAL	Hardware Abstraction Layer
HID	Human Interface Device
IRQ	Interrupt Request
ISM	Industrial-Scientific-Medical
LNA	Low Noise Amplifier
LSB	Least Significant Bit
LSByte	Least Significant Byte
MCU	Microcontroller
Mbps	Megabit(s) per second
MISO	Master In Slave Out
MoQ	Minimum Order Quantity
MOSI	Master Out Slave In
MSB	Most Significant Bit
MSByte	Most Significant Byte
PCB	Printed Circuit Board
PER	Packet Error Rate
PID	Packet Identity Bits
PLD	Payload
PRX	Primary RX
PSW	Program Status Word Register
PTX	Primary TX
pwrdwn	Power Down
PWR_UP	Power Up
RAM	Random Access Memory
RDSR	Read Status Register
rfce	Radio transceiver chip enable
RX	Receive
RX_DR	Receive Data Ready
SDK	Software Development Kit
SP	Stack Pointer
SPI	Serial Peripheral Interface
TX	Transmit
TX_DS	Transmit Data Sent
USB	Universal Serial Bus
WO	Write Only

Table 153. Glossary

Appendix A - (USB memory configurations)

The USB buffer memory has a total size of 512 bytes. Bulk/control buffer size can be 2, 4, 8, 16, 32 or, 64 bytes, while ISO buffers (if used) must be multiples of 16 bytes.

Some example configurations are given below.

Configuration 1

Endpoint 0-5 Bulk/control IN/OUT, each of size 32 bytes.

Endpoint 8 ISO IN/OUT, each of size 32 bytes (with double buffering).

Total buffer area: 448 bytes.

Register	Value (hex)	Calculation	Comment
bout1addr	0x10	(ep0 out size)/2	Start addr. of bulk 1 OUT
bout2addr	0x20	bout1addr + (ep1out size)/2	Start addr. of bulk 2 OUT
bout3addr	0x30	bout2addr + (ep2 out size)/2	Start addr. of bulk 3 OUT
bout4addr	0x40	bout3addr + (ep3 out size)/2	Start addr. of bulk 4 OUT
bout5addr	0x50	bout4addr + (ep4 out size)/2	Start addr. of bulk 5 OUT
binstaddr	0x30	(bulk out size)/4	Start addr. of bulk 1 IN
bin1addr	0x10	(ep0 in size)/2	Start addr. of bulk 1 IN
bin2addr	0x20	bin1addr + (ep1 in size)/2	Start addr. of bulk 2 IN
bin3addr	0x30	bin2addr + (ep2 in size)/2	Start addr. of bulk 3 IN
bin4addr	0x40	bin3addr + (ep3 in size)/2	Start addr. of bulk 4 IN
bin5addr	0x50	bin4addr + (ep4 in size)/2	Start addr. of bulk 5 IN
isostaddr	0x18	(bulk size)/16	Start addr. of iso
out8addr	0x00	0	Start addr. of iso OUT
in8addr	0x08	(ep8 out size)/4	Start addr. of iso IN
isosize	0x04	(iso size)/16	

Table 154. Configuration 1

Configuration 2

Endpoint 0-2 bulk/control IN/OUT, each of size 32 bytes

Endpoint 3-4 bulk IN/OUT, each of size 16 bytes

Endpoint 8 ISO IN/OUT, each of size 32 bytes (with double buffering).

Total buffer area: 320 bytes

Register	value (hex)	Calculation	Comment
bout1addr	0x10	(ep0 out size)/2	Start addr. of bulk 1 OUT
bout2addr	0x20	bout1addr + (ep1 out size)/2	Start addr. of bulk 2 OUT
bout3addr	0x30	bout2addr + (ep2 out size)/2	Start addr. of bulk 3 OUT
bout4addr	0x38	bout3addr + (ep3 out size)/2	Start addr. of bulk 4 OUT
binstaddr	0x20	(bulk out size)/4	Start addr. of bulk 1 IN
bin1addr	0x10	(ep0 in size)/2	Start addr. of bulk 1 IN
bin2addr	0x20	bin1addr + (ep1 in size)/2	Start addr. of bulk 2 IN

Register	value (hex)	Calculation	Comment
bin3addr	0x30	bin2addr + (ep2 in size)/2	Start addr. of bulk 3 IN
bin4addr	0x40	bin3addr + (ep3 in size)/2	Start addr. of bulk 4 IN
isostaddr	0x10	(bulk size)/16	Start addr. of iso
out8addr	0x00	0	Start addr. of iso OUT
in8addr	0x08	(ep8 out size)/4	Start addr. of iso IN
isosize	0x04	(iso size)/16	

Table 155. Configuration 2

Unused bout5addr and bin5addr shall be 0x00.

Configuration 3

Endpoint 0-3 bulk IN/OUT, each of size 16 bytes

Endpoint 4-5 bulk IN/OUT, each of size 32 bytes

Endpoint 8 iso IN/OUT, each of size 32 bytes (with double buffering)

Total buffer area: 320 bytes.

Register	Value (h)	Calculation	Comment
bout1addr	0x08	(ep0 out size)/2	Start addr. of bulk 1 OUT
bout2addr	0x10	bout1addr + (ep1 out size)/2	Start addr. of bulk 2 OUT
bout3addr	0x18	bout2addr + (ep2 out size)/2	Start addr. of bulk 3 OUT
bout4addr	0x20	bout3addr + (ep3 out size)/2	Start addr. of bulk 4 OUT
bout5addr	0x30	bout4addr + (ep4 out size)/2	Start addr. of bulk 5 OUT
binstaddr	0x20	(bulk out size)/4	Start addr. of bulk 1 IN
bin1addr	0x08	(ep0 in size)/2	Start addr. of bulk 1 IN
bin2addr	0x10	bin1addr + (ep1 in size)/2	Start addr. of bulk 2 IN
bin3addr	0x18	bin2addr + (ep2 in size)/2	Start addr. of bulk 3 IN
bin4addr	0x20	bin3addr + (ep3 in size)/2	Start addr. of bulk 4 IN
bin5addr	0x30	bin4addr + (ep4 in size)/2	Start addr. of bulk 5 IN
isostaddr	0x10	(bulk size)/16	Start addr. of iso
out8addr	0x00	0	Start addr. of iso OUT
in8addr	0x08	(ep8 out size)/4	Start addr. of iso IN
isosize	0x04	(iso size)/16	

Table 156. Configuration 3

Configuration 4

Endpoint 0-1 bulk/control IN/OUT, each of size 32 bytes

Endpoint 8 ISO IN/OUT, each of size 32 bytes (with double buffering).

Total buffer area: 192 bytes.

Register	Value (h)	Calculation	Comment
bout1addr	0x10	(ep0 out size)/2	Start addr. of bulk 1 OUT
binstaddr	0x10	(bulk out size)/4	Start addr. of bulk 1 IN
bin1addr	0x10	(ep0 in size)/2	Start addr. of bulk 1 IN
isostaddr	0x08	(bulk size)/16	Start addr. of iso
out8addr	0x00	0	Start addr. of iso OUT
in8addr	0x08	(ep8 out size)/4	Start addr. of iso IN
isosize	0x04	(iso size)/16	

Table 157. Configuration 4

Unused bout2addr to bout5addr and bin2addr to bin5addr shall be 0x00.

Appendix B - Configuration for compatibility with nRF24XX

How to setup the radio module in nRF24LU1+ to receive from an nRF2401/nRF2402/nRF24E1/nRF24E2/ nRF24LE1:

1. Use the same CRC configuration as the nRF2401/nRF2402/nRF24E1/nRF24E2/nRF24LE1.
2. Set the `PWR_UP` and `PRIM_RX` bit to 1.
3. Disable auto acknowledgement on the addressed data pipe.
4. Use the same address width as the PTX device.
5. Use the same frequency channel as the PTX device.
6. Select data rate 1 Mbps on both nRF24LU1+ and nRF2401/nRF2402/nRF24E1/nRF24E2/ nRF24LE1.
7. Set correct payload width on the addressed data pipe.
8. Set `rfce` high.

How to setup the radio module in nRF24LU1+ to transmit to an nRF2401/nRF24E1/nRF24LE1:

1. Use the same CRC configuration as the nRF2401/nRF24E1/nRF24LE1.
2. Set the `PRIM_RX` bit to 0.
3. Set the Auto Retransmit Count to 0 to disable the auto retransmit functionality.
4. Use the same address width as the nRF2401/nRF24E1/nRF24LE1.
5. Use the same frequency channel as the nRF2401/nRF24E1/nRF24LE1.
6. Select data rate 1 Mbps on both nRF24LU1+ and nRF2401/nRF24E1/nRF24LE1.
7. Set `PWR_UP` high.
8. Clock in a payload that has the same length as the nRF2401/nRF24E1/nRF24LE1 is configured to receive.
9. Pulse `rfce` to transmit the packet.