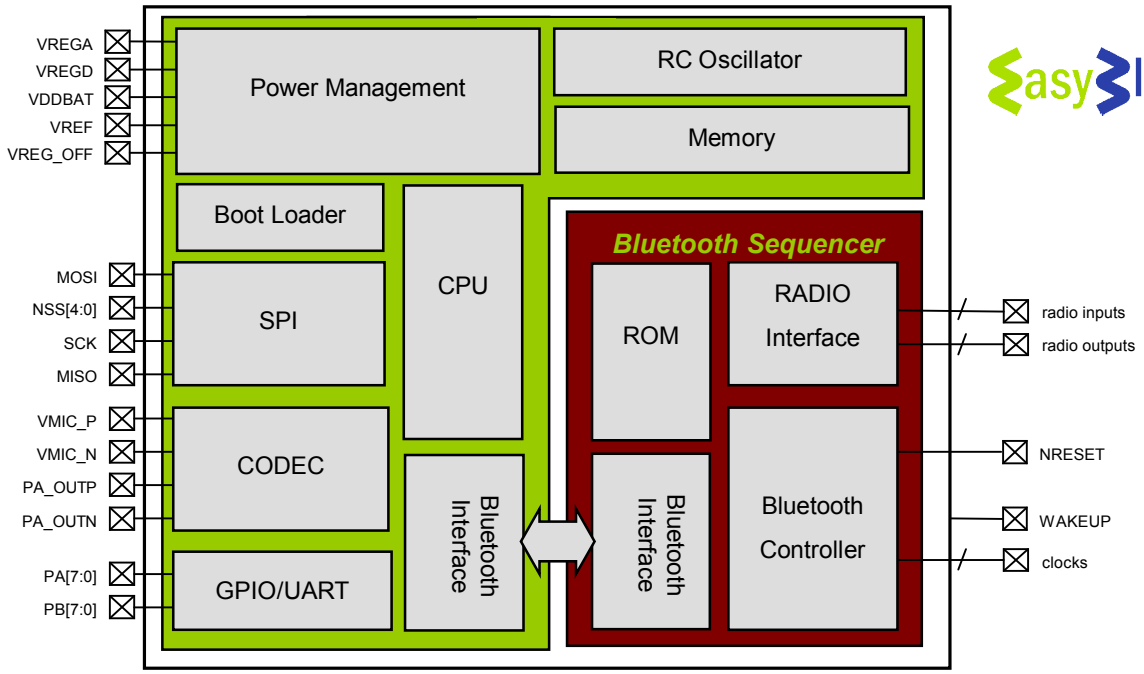
Personal Area Network **Bluetooth QD ID B012209**



SX1441

Ultra Low Power Bluetooth® V1.2 Soc for Wireless Headset and Data Applications with DSP Capabilities

GENERAL DESCRIPTION

The SX1441 is a Bluetooth® System-on-Chip based on the Semtech Bluetooth Sequencer, which includes a fully programmable 8-bit application microcontroller, a high speed UART, SPI interface, RC oscillator, power management unit, and an on-chip voice CODEC with DMA interface. The purpose of the SX1441 is to offer a very high level of integration requiring a minimum of external components to build complete voice and data applications whilst maintaining design flexibility.
This product has been designed for ultra low power consumption and low cost solutions. By combining the SX1441 with a low power 2.4 GHz radio device such as the XE1413, from Semtech, an ultra low power Bluetooth wireless headset consuming less than 23mW @1.8V (HV3) can be built.

APPLICATIONS

- Bluetooth wireless headset
- Handsfree kit
- VoIP, VoRF
- Cable replacement
- Computer accessories

KEY PRODUCT FEATURES

- Ultra low power single-chip Bluetooth SoC, fully Bluetooth rev 1.2 compliant. Supports AFH, Fast Connect and eSCO
- Fully integrated Bluetooth protocol stack up to the HCI, compliant to revision 1.2
- On-chip 16-bit audio linear Codec with DMA interface, preamplifier and audio power amplifier
- Minimum of external components required
- Small form factor
- On-chip battery level detector
- High speed general purpose UART
- Supports simultaneously one SCO and up to three ACL channels
- On-chip MCU and ROM/SRAM memory
- Ni-MH or Li-ion polymer rechargeable battery operation. Supply voltage range 1.8V to 3.6V
- Ultra low power consumption
- Supports CX72303 and XE1413 BT 1.2 radios

ORDERING INFORMATION

Part Number	Description
SX1441IO77TR LF	Bluetooth SoC for voice and data applications

Table of Contents

1 APPLICATION INFORMATION – SX1441-BASED SYSTEM LEVEL BLOCK DIAGRAM

The Semtech SX1441, a member of the EasyBlue™ family, is based on a unique Embedded-Host architecture which enables any data or voice application to be enhanced with ultra low power Bluetooth technology, with low risk and a short development time.
The core of the SX1441 is the Semtech ROM-based Bluetooth sequencer combined with an embedded 8-bit RISC microcontroller and several standard peripherals such as GPIO, high speed UART, audio CODEC, and a power management unit. The Bluetooth sequencer executes the lower layers of the Bluetooth stack, while the microcontroller runs the application and the higher levels of the protocol. Since the sequencer and the microcontroller are independent, the effort required for validation and qualification of the Bluetooth protocol is greatly decreased.
A typical wireless headset block diagram using the SX1441 is shown in Figure 1. The on-chip CODEC is connected with a microphone and a speaker. The Serial Peripheral Interface (SPI) directly interfaces to an external Flash memory. This memory stores the application and the upper layers of the Bluetooth protocol stack which are loaded at boot-up time, and then executed by the on-chip application processor.
Fully programmable General Purpose Input/Output ports (GPIO) are available to interface push-buttons, LED's or other peripherals. The high speed UART supports hardware flow control and data rates up to 921kbit/s.

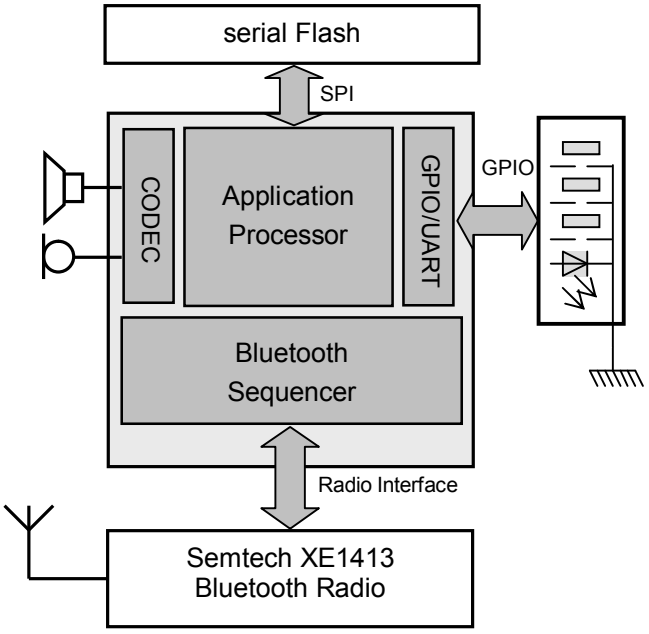


Figure 1 - Bluetooth Headset Application

The on-chip host processor runs the application software and the upper layer Bluetooth protocol stack software while the Bluetooth sequencer handles the low level of the protocol with no intervention by the application processor. This architecture guarantees that the real time operations of the lower levels cannot be influenced by the application. Qualified upper layer Bluetooth protocol software from various 3rd party suppliers can be supplied to run on the SX1441. This system architecture definitely eases the software development and Bluetooth qualification processes and guarantees the highest flexibility. The Bluetooth qualification process for the final application is simplified by the fact that the SX1441 uses a qualified Bluetooth ROM implementation

2 SX1441 PINOUT

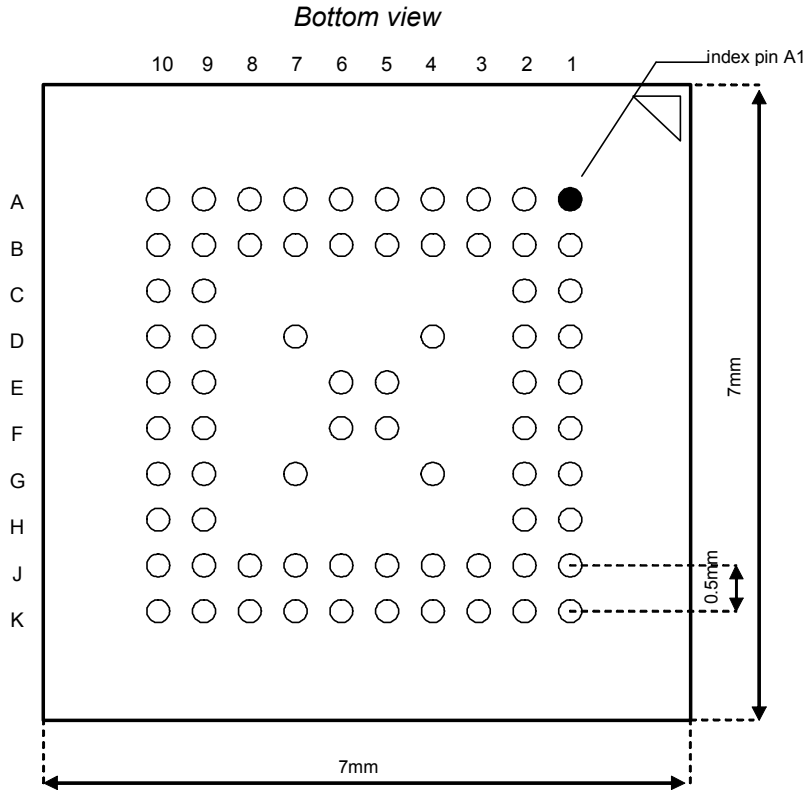


Figure 2 - LFBGA72, bottom view

2.1 PIN DESCRIPTION

Pin	Symbol	Type/ capabilities	Reset	Description	Voltage level
A1	TP0	Do not connect	Do not connect	Test pin	-
A2	TP1	Do not connect	Do not connect	Test pin	-
A3	TP2	Connect to ground	Connect to ground	Test pin	-
A4	PB[1]	D I O ud	D I u	General purpose port B I/O	VDDIO_DIG
A5	PB[3]	D I O ud	D I u	General purpose port B I/O	VDDIO_DIG
A6	PB[5] / UA_RTS	D I O ud	D I u	General purpose port B I/O UART RTS handshaking	VDDIO_DIG
A7	PB[7] / UA_RX	D I O ud	D I u	General purpose port B I/O UART receive signal	VDDIO_DIG
A8	MOSI	D I O u	D O	SPI master Out slave In	VDDIO_DIG
A9	MISO	D I O u	D I u	SPI master In slave Out	VDDIO_DIG
A10	SCK	D I O u	D O	SPI clock	VDDIO_DIG
B1	NRESET	D I u	D I u	Master Reset	VDD_M
B2	VSS_DIG	P	P	Digital core ground	-

Pin	Symbol	Type/ capabilities	Reset	Description	Voltage level
B3	PB[0]	D I O ud	D I u	General purpose port B I/O	VDDIO_DIG
B4	PB[2]	D I O ud	D I u	General purpose port B I/O	VDDIO_DIG
B5	PB[4] / UA_CTS	D I O ud	D I u	General purpose port B I/O UART CTS handshaking	VDDIO_DIG
B6	PB[6] / UA_TX	D I O ud	D I u	General purpose port B I/O UART transmit signal	VDDIO_DIG
B7	VSSIO_DIG	P	P	digital pads ground	-
B8	VDDIO_DIG	P	P	digital pads supply voltage	-
B9	NSS[0]	D I O u	D I u	First SPI slave select	VDDIO_DIG
B10	NSS[1]	D I O u	D I u	Second SPI slave select	VDDIO_DIG
C1	VREG_OFF	D O	D O	Internal regulators status	VDDM
C2	VDDBAT	A I	A I	Sensor input for battery end-of-life detection	-
C9	NSS[2]	D I O u	D I u	Third SPI slave select	VDDIO_DIG
C10	NSS[3]	D I O u	D I u	Fourth SPI slave select	VDDIO_DIG
D1	VMIC_P	A I	A I	Microphone positive input	-
D2	VDD_ANA	P	P	Analog core supply voltage	-
D4	DBG[4]	D I O k	D I k	Debug Interface HCI CTS	VDDIO_DIG
D7	DBG[7]	D I O k	D I k	Debug Interface HCI TX	VDDIO_DIG
D9	PA[0]	D I u d	D I u	General purpose port A input	VDDIO_DIG
D10	NSS[4]	D I O u	D I u	Fifth SPI slave select	VDDIO_DIG
E1	VMIC_N	A I	A I	Microphone negative input	-
E2	VREGA	A O	A O	Analog regulated voltage	-
E5	DBG[5]	D I O k	D I k	Debug Interface HCI RTS	VDDIO_DIG
E6	DBG[6]	D I O k	D I k	Debug Interface HCI RX	VDDIO_DIG
E9	PA[2]	D I u d	D I u	General purpose port A input	VDDIO_DIG
E10	PA[1]	D I u d	D I u	General purpose port A input	VDDIO_DIG
F1	VREF	A O	A O	Reference voltage output	-
F2	VDD_M	P	P	Main supply voltage	-
F5	DBG[1]	D I O k	D I k	Debug Interface PCM clock	VDDIO_DIG
F6	DBG[3]	D I O k	D I k	Debug Interface PCM data in	VDDIO_DIG
F9	PA[4]	D I u d	D I u	General purpose port A input	VDDIO_DIG
F10	PA[3]	D I u d	D I u	General purpose port A input	VDDIO_DIG
G1	VSS_M	P	P	Analog ground	-
G2	VREGD	A O	A O	Digital regulated voltage	-

Pin	Symbol	Type/ capabilities	Reset	Description	Voltage level
G4	DBG[0]	D I O k	D I k	Debug Interface PCM fsync	VDDIO_DIG
G7	DBG[2]	D I O k	D I k	Debug Interface PCM data out	VDDIO_DIG
G9	PA[6]	D I u d	D I u	General purpose port A input	VDDIO_DIG
G10	PA[5]	D I u d	D I u	General purpose port A input	VDDIO_DIG
H1	PA_OUTP	A O	A O	Power amplifier positive output	VDD_PA
H2	VDD_PA	P	P	Power amplifier supply voltage	-
H9	VDD_DIG	P	P	Digital core supply voltage	-
H10	PA[7]	D I u d	D I u	General purpose port A input	VDDIO_DIG
J1	PA_OUTN	A O	A O	Power amplifier negative output	VDD_PA
J2	TP3	Do not connect	Do not connect	Test pin	-
J3	WAKEUP	D I d	D I d	Chip wake up	VDD_M
J4	SPI_DATA_IN	D I k	D I k	Radio SPI input	VDDIO
J5	VDDIO	P	P	Radio pads supply voltage	-
J6	VSSIO	P	P	Radio pads ground	-
J7	SPI_CLK_OUT	D O	D O	Radio SPI serial clock	VDDIO
J8	SPI_DATA_OUT	D O	D O	Radio SPI data out	VDDIO
J9	SYS_CLOCK_IN	D I	DI	Master clock input	VDDIO
J10	DOC_SDIO	D I O u	D I u	Monitor data I/O	VDDIO_DIG
K1	VSS_PA	P	P	Power amplifier ground	-
K2	TP4	Do not connect	Do not connect	Test pin	-
K3	SLW_CLK_IN	D I k	D I k	32 kHz clock input	VDDIO
K4	RX_DATA	D I k	D I k	Radio RX data	VDDIO
K5	SPI_EN_BAR	D O	D O	Radio SPI select	VDDIO
K6	TX_EN	D O	D O	Radio TX enable	VDDIO
K7	SYNC_DETECT	D O	D O	Radio sync detect	VDDIO
K8	RX_EN	D O	D O	Radio RX enable	VDDIO
K9	TX_DATA	D O	D O	Radio TX data	VDDIO
K10	DOC_SCK	D I u	D I u	Monitor clock	VDDIO_DIG

A : Analog D : Digital I : Input O : Output
u : Internal pull-up d : Internal pull-down k : Internal keeper P : Power

Table 1 – Pin description

3 DETAILED FUNCTIONAL DESCRIPTION

3.1 BLOCK DIAGRAM

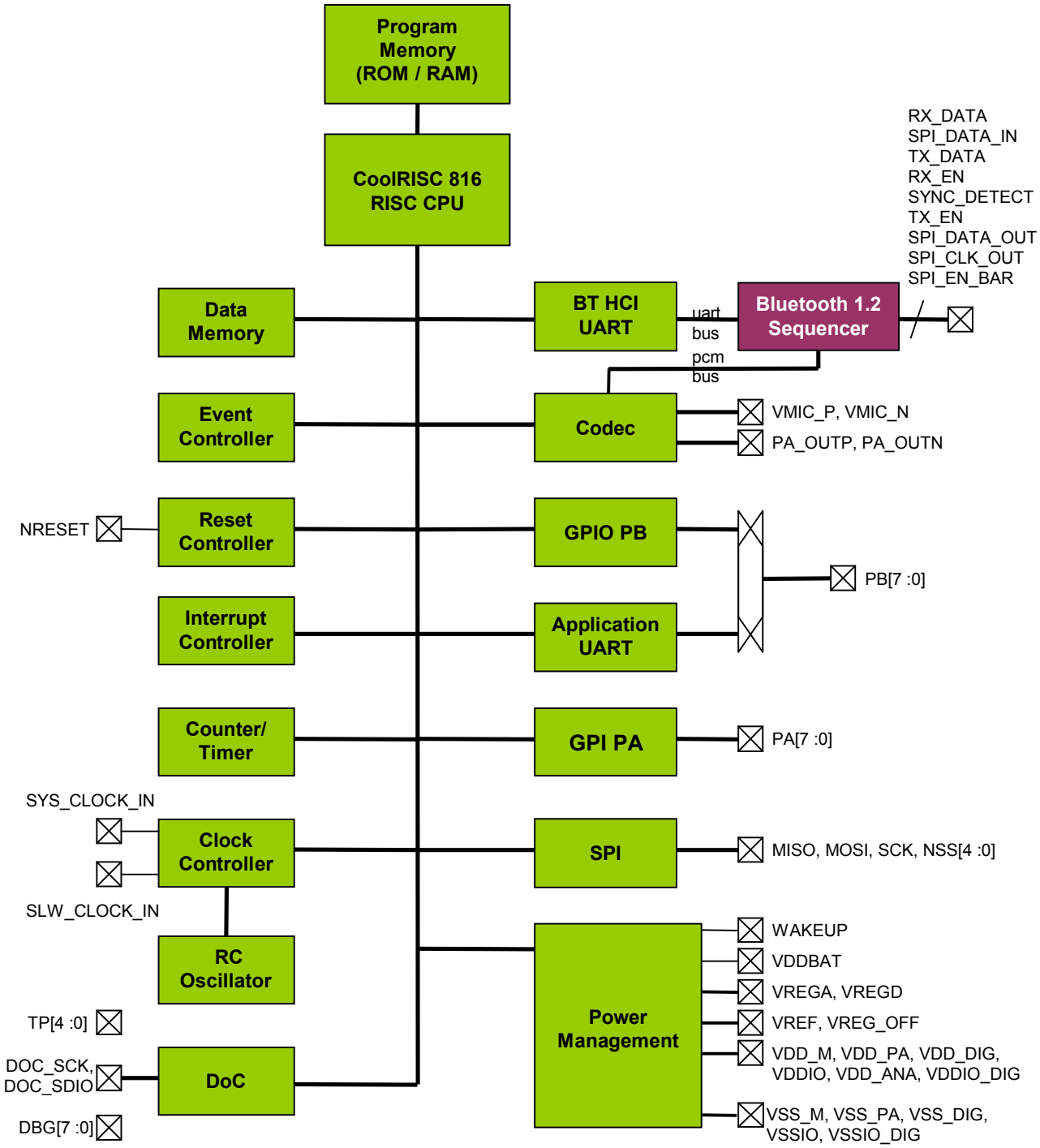


Figure 3 - SX1441 block diagram

A high-level block diagram of the SX1441 is shown in Figure 3. The CoolRISC® 816 8-bit RISC processor is optimized for both computation power and energy consumption efficiency. Every instruction executes in one clock cycle. The system frequency can be selected from different possible clock signals. SLW_CLOCK_IN, which is typically 32 kHz, SYS_CLOCK_IN, which is typically 13 MHz, or the internal programmable RC oscillator f_{RC} up to 15 MHz. The program memory consists of; firstly 4k instructions in ROM for the boot code and the debug drivers, and then 40k instructions in RAM dedicated to the application and the upper layers of the Bluetooth protocol stack. The data memory is 8 kbyte RAM. The interrupt and event controllers manage interrupts and events from peripherals and internal timers. The reset controller takes care of the power-on phase. The clock controller selects the processor and peripherals clocks between the internal RC oscillator or the two external clocks.
GPIO's are split into two peripherals: a) port A (PA) is an 8-bit wide input digital port with selectable pull-up and debouncer, which can also be programmed to generate interrupts and resets; and b) port B (PB), an 8-bit wide input/output digital port with selectable pull-up and open-drain capabilities.
The SPI and UART interfaces implement serial communication protocols. The SPI can communicate with up to 5 peripherals, one of them being the serial non-volatile memory storing the application code. The UART has an 8-byte FIFO and supports hardware flow control.
The integrated power management unit generates the regulated supply voltages for the SX1441, thus reducing the number of external components. It also monitors the battery voltage to detect the end-of-life of the battery.
The CODEC is compliant with the Bluetooth audio specifications and integrates a built-in CVSD coder/decoder. The audio samples are transferred directly from the Bluetooth sequencer to the CODEC to reduce the processor load and power consumption. A DMA interface allows transferring of samples directly between the memory and the CODEC.
The Bluetooth sequencer is a complete dedicated Bluetooth co-processor. It implements the lower layers of the Bluetooth protocol stack from the radio interface up to the HCI. It runs independently from the processor and communicates with it through a dedicated internal UART link. This massively simplifies the debugging of the final product and the Bluetooth qualification process since the application cannot disturb the time-critical part of the Bluetooth stack. The Bluetooth sequencer supports all modes of operation (Active, Hold, Sniff, Park, Standby), all packet types, simultaneous operation with up to 7 ACL (data) links and one SCO (audio) link in a point-to-point, piconet, or scatternet network configuration.
The debug-on-chip (DoC) peripheral interfaces the chip with the software debugger.

3.2 HOST PROCESSOR SYSTEM

3.2.1 CoolRISC 816 CPU

The CPU of the SX1441 is a CoolRISC816, an 8-bit low power RISC core. The instruction set is made up of 35 generic instructions coded on 22 bits and always executed in one clock cycle, including conditional jumps and 8x8 multiplications, thus providing 1 MIPS/MHz. Instructions and data memory are separated (Harvard architecture). The 16 8-bit registers enable the use of a C compiler.
The complete CPU hardware and software description is given in the document "CoolRISC 816, 8-bit Microprocesor Core, Hardware and Software Reference Manual", version 4.5 which can be found on the Semtech website (http://www.semtech.com).

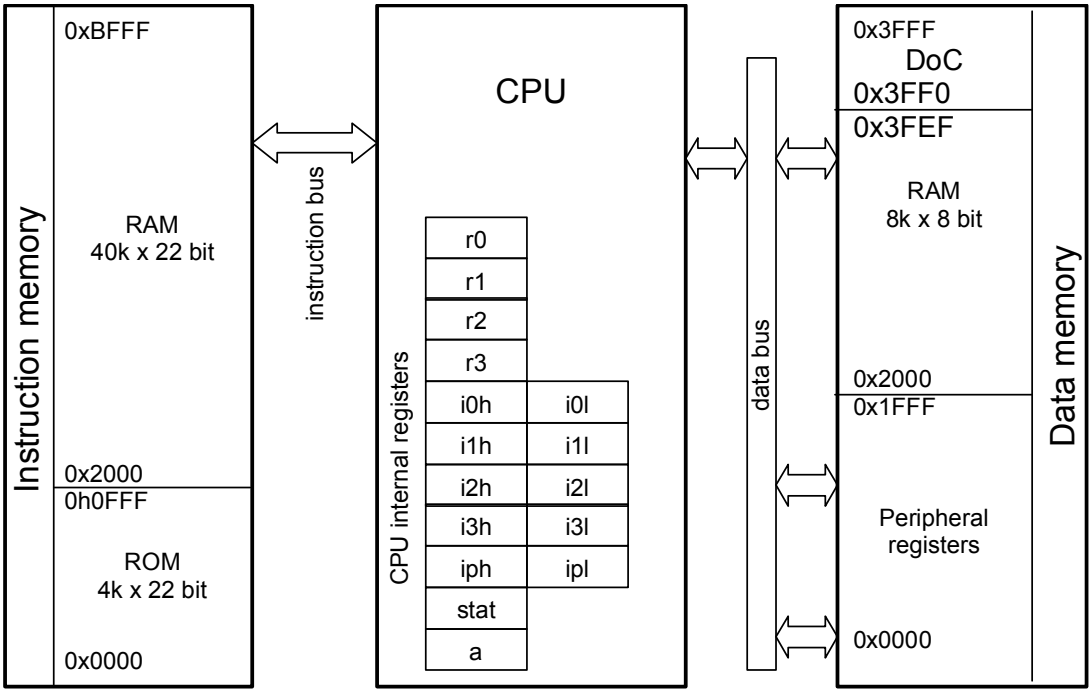


Figure 4 - Memory organization

3.2.2 Program Memory

The instruction memory is composed of both ROM and RAM. The ROM size is 4096 x 22-bit and stores the boot code and the Debug-On-Chip (DoC) driver. The RAM size is 40k instructions which is completely available for the application, except for the last 64 instructions between 0xBFB0 and 0xBFFF which are used by the Debug-on-Chip.
The ROM is located from the address 0x0000 to the address 0x0FFF. The RAM is located in the 0x2000 to 0xBFFF range. Addresses 0x2000 to 0x2004 are jump and interrupt vectors.

Address	Usage	Comment
0x2000	start vector	Usually set to 0x2005. The code actually begins at 0x2005
0x2001	Mid priority interrupt handler	
0x2002	Low priority interrupt handler	
0x2003	High priority interrupt handler	
0x2004	RESERVED	

Table 2 – Jump and interrupt vectors address table

3.2.3 Data Memory

The data memory space is made of 8 kbytes of RAM. The last 16 bytes between 0x3FF0 and 0x3FFF are reserved for the Debug-On-Chip (DoC) interface. The rest of the space from 0x2000 to 0x3FEF is available for the application. The peripheral registers are located in the page 0 of the data memory space.

Block	Address range
System registers (reset controller & clock controller)	0x0010 to 0x001F
Port A registers	0x0020 to 0x0027
Port B registers	0x0028 to 0x002D
Application UART registers	0x0030 to 0x0037
reserved	0x0038 to 0x003B
Event controller registers	0x003C to 0x003F
Interrupt controller registers	0x0040 to 0x0047
Power management unit registers	0x0048 to 0x004C
HCI UART (to/from Bluetooth Sequencer) registers	0x0050 to 0x0057
Counter registers	0x0058 to 0x005F
SPI registers	0x0068 to 0x006F
Bluetooth Sequencer registers	0x007C to 0x007D
Debug Interface (reserved)	0x0080 to 0x009F
Codec registers	0x00E0 to 0x00FF
Data Memory	0x2000 to 0x3FEF
Debug-on-Chip memory (reserved)	0x3FF0 to 0x3FFF

Table 3 - Data memory and registers map

3.3 POWER MANAGEMENT UNIT

3.3.1 Features

- Wide power supply range, VDD_M from 2.2 to 3.6V.
- High current (50 mA) integrated 1.8V regulator to supply the digital core of the SX1441 and external chips, VREGD output.
- Integrated 1.8V analog regulator to supply analog blocks, VREGA output.
- Integrated temperature-compensated voltage reference.
- Battery end-of-life detection, VDDBAT input.
- Mode of operation controller to suppress the need for external power supply switch.
- Ultra low power consumption in OFF mode.

3.3.2 Register Map

Name	Address (Hex)
RegPmgtVrega	0x0048
RegPmgtVregd	0x0049
RegPmgtEol	0x004A

Table 4 - Power management unit register mapping

Pos	RegPmgtVrega	r/w	Reset	Function
7:6	-	r	00	reserved
5	DefaultVrega	rw	1	force analog tuning to default values
4	EnableVrega	rw	0	1 = VREGA voltage regulator switched on 0 = VREGA voltage regulator switched off
3:0	TuneVrega	rw	0011	adjust VREGA value

Table 5 - RegPmgtVrega register

Pos	RegPmgtVregd	r/w	Reset	Function
7	WakeUp	r	0	value of the wakeup pin
6:5	-	r	00	reserved
4	VregdStatus	r	1	1 = VREGD voltage regulator switched on 0 = VREGD voltage regulator switched off
3:2	TuneVregd	rw	00	adjust VREGD value
1	VregdLock	w	1	1 = lock VREGD voltage regulator 0 = shut down VREGD voltage regulator
0	-	r	0	reserved

Table 6 - RegPmgtVregd register

Pos	RegPmgtEol	r/w	Reset	Function
7	EolOk	r	x	0 = VDDBAT pin voltage < EolThreshold 1 = VDDBAT pin voltage ≥ EolThreshold
6	-	r	0	reserved
5	EnableEol	rw	0	1 = battery end-of-life switched on 0 = battery end-of-life switched off
4:0	EolThreshold	rw	00000	adjust battery end-of-life comparator threshold

Table 7 - RegPmgtEol register

3.3.3 Modes of Operation

The power management unit's role is to generate regulated voltages for both internal and the external components such as the non-volatile serial memory and the radio chip. It includes a controller to switch on/off all voltage regulators when needed in order to reduce power consumption. Three modes of operation are defined (see Figure 6):

- ***OFF mode***: all internal power supplies are shut down. Power consumption is very low, typically a few micro-amps.
- ***ON mode***: all blocks except the CODEC are powered. The application is running and a Bluetooth connection may be active. The chip enters this state when the WAKEUP input is set high (see Figure 5), and leaves this mode under software control, when requested by the application. In the ON mode, the pin VREGD outputs 1.8V and the pin VREGA is floating.
- ***AUDIO mode***: all blocks are powered. It is entered upon request from the application. In the AUDIO mode, the pins VREGA and VREGD output 1.8V.

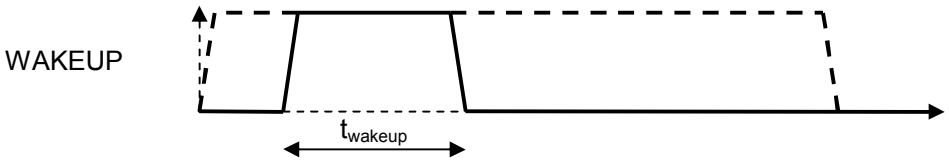


Figure 5 - WAKEUP timing diagram

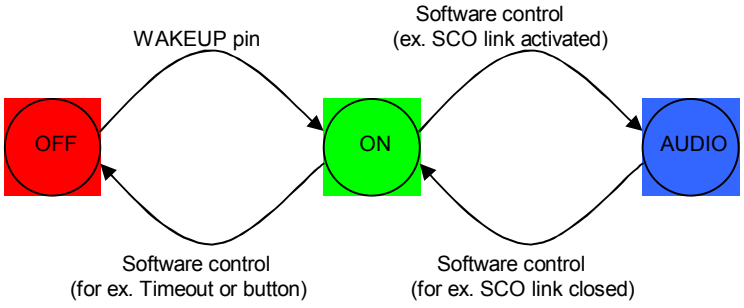


Figure 6 - SX1441 power management modes

3.3.4 Block Diagram

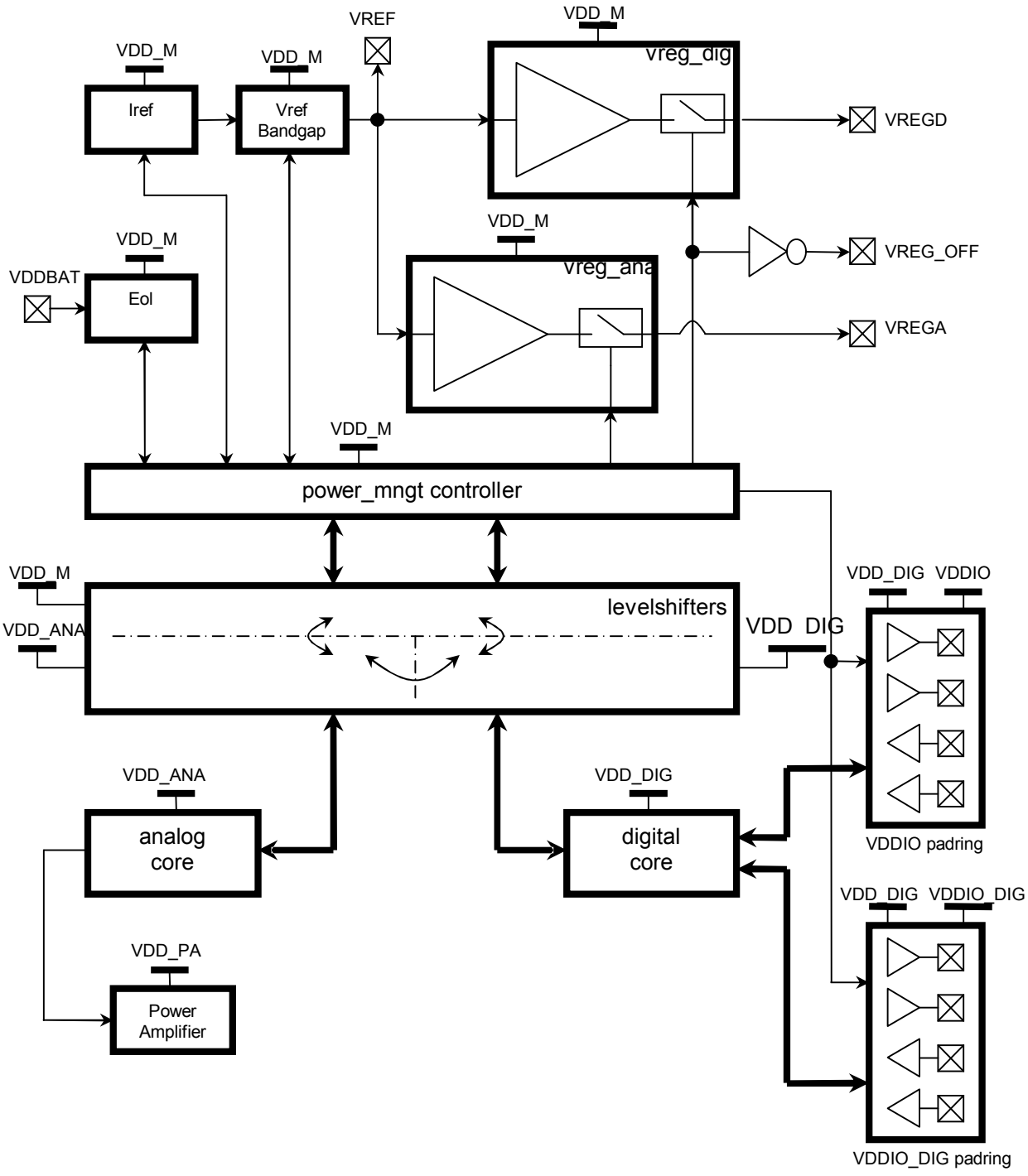


Figure 7 - Power management unit block diagram

The main power supply is VDD_M. It powers the power management unit and some I/O pads. The power management unit generates the VREGD and VREGA regulated voltages. VREGD is usually used to supply the digital core, through the VDD_DIG pin, and external components such as a serial non-volatile memory and the radio chip. VREGA is usually used to supply the internal analog blocks, through the pin VDD_ANA. It may also be used to power an external microphone.
VDDIO is the power supply for the radio interface. If the radio chip is powered by VREGD, then VDDIO should be connected to VREGD as well. VDDIO_DIG is the power supply for most of the digital pads. Depending on the application, it may be connected to VREGD, VDD_M, or any other power supply which fulfills the specifications.

VDD_M should be decoupled with a capacitor C_{VDD_M} for best performances. VREGD has to be connected to an external capacitor C_{VREGD} to insure the stability and the performance of the voltage regulator. VREGA has to be connected to an external capacitor C_{VREGA} to insure the stability and the performance of the voltage regulator. VREF is connected to an external capacitor C_{VREF}. VDD_PA can be connected to VREGD.

3.3.5 Regulators Specifications, External Components

Symbol	Description	Min	Typ	Max	Unit	Comments
V_{REGA}	Analog regulated output voltage	1.62	1.8	1.98	V	I_load=1mA
V_{REGD}	Digital regulated output voltage	1.62	1.8	1.98	V	I_load=50mA
I_{REGA}	Output current on VREGA			10	mA	Recommended max. load
I_{REGD}	Output current on VREGD			50	mA	Recommended max. load

Note : Values above are specified across temperature range and for VDD_M > 2.2V unless otherwise specified

Table 8 - On-chip voltage regulators specifications

Symbol	Value
C_{VREGD}	4.7 uF
C_{VREGA}	1 uF
C_{VREF}	1 uF
C_{VDD_M}	1 uF

Table 9 - Typical external components

Capacitors should be added to decouple VDD_PA, VDD_DIG, VDD_ANA, VDDIO, and VDDIO_DIG, as a common practice.

3.3.6 Battery End-Of-Life (EOL)

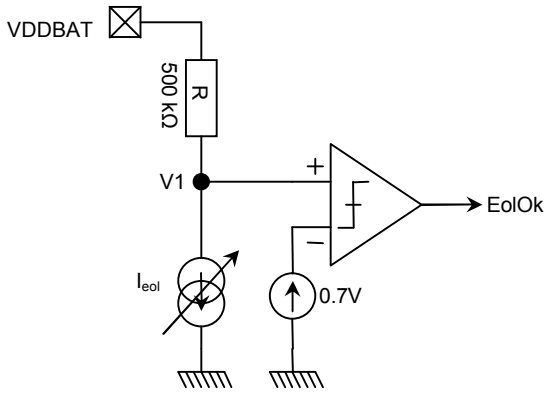


Figure 8 - Battery end-of-life structure

The battery end-of-life circuit structure is described in Figure 8. A voltage is created by drawing a constant current I_{eol} from the pin VDDBAT through the internal resistor R, and is compared with a voltage reference. The bit **EolOk** of the register **RegPmgtEol** is directly the output of the comparator. The EolOk is set to "1" whenever the voltage at VDDBAT is higher or equal to the threshold voltage $V_{EOLThreshold}$. This threshold voltage of the comparator is given by the Equation 1. For cThe start up time of the end-of-life circuit is $T_{EOLstart}$. The response time to a change on VDDBAT is T_{EOLres}.

$$V_{EOLThreshold} = V_{EOLref} + V_{EOLstep} \cdot \sum_{i=0}^{4} 2^i \cdot EolThreshold[i]$$

Equation 1 - Threshold voltage of the EOL comparator

Symbol	Parameter	Min	Typ	Max	Unit	Comment
V_{EOLref}	EOL reference voltage	0.710	0.725	0.740	V	@ VDD_M=3V, 25°C, (bandgap test)
$V_{EOLstep}$ (*)	Threshold tuning step	40	45	50	mV	@ VDD_M=3V, 25°C
EOLThreshold	EOLthreshold offset setting	0.710		2.34	V	Tested at VBAT=1.8V. and 0x31 and 0x3F register settings
$T_{EOLstart}$ (*)	start-up time			100	µs	
T_{EOLres} (*)	time response			20	µs	

Note1 : Values above are specified across temperature range and for VDD_M > 2.2V unless otherwise specified
Note2 : Values marked with asterisks are not production tested and guaranteed by design.

Table 10 - End-of-life analog specifications

3.4 RESET CONTROLLER

3.4.1 Features

- Handles different reset sources: power-on-reset, NRESET pin, BusError, Watchdog, and port PA
- Power-on-reset/Brownout detector without external components
- Programmable watchdog timer
- Reset can be triggered by NRESET pin

3.4.2 Register Map

Name	Address (Hex)
RegSysCtrl	0x0010
RegSysWd	0x0014

Table 11 - Reset controller registers

Pos	RegSysCtrl	r/w	Reset	Function
7	reserved	rw	0	Bit reserved for test purpose
6	-	r	0	reserved
5	EnableBusError	rw	0	1 = BusError reset is enabled 0 = BusError reset is disabled
4	EnableResetWD	rw	0	1 = Watchdog reset is enabled 0 = Watchdog reset is disabled
3:0	-	r	0000	reserved

Table 12 - RegSysCtrl register

Pos	RegSysWd	r/w	Reset	Function
7:4	-	r	0000	reserved
3:0	WDKey	rw	0000	Watchdog key

Table 13 - RegSysWd register

3.4.3 Power-On-Reset / Brownout Detector

The power-on-reset monitors both VDD_M and VDD_DIG. Upon start-up, when both voltages reach a level sufficient to ensure correct circuit behavior, the internal reset signal is released. Then, if during operations the supply voltage drops below the specified threshold (see
Note1 : Values above are specified across temperature range unless otherwise specified
Note2 : Values marked with asterisks are not production tested and guaranteed by design.

Table 14), the circuit goes into a reset mode.

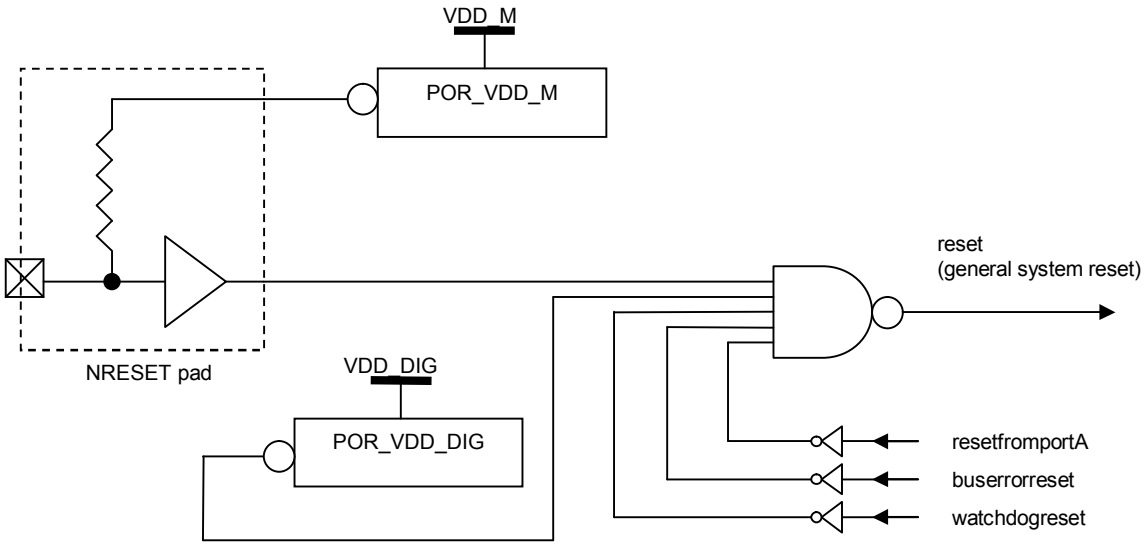


Figure 9 – POR, NRESET, and reset circuitry

The output of POR_VDD_M controls the pull resistor of the NRESET pad. If the NRESET pad is left unconnected (recommended) the POR_VDD_M is propagated into the system. Otherwise, the internal nreset_system signal may be activated by connecting the NRESET pad to the ground. The NRESET pad is active low.
The POR_VDD_M insures that VDD_M is stable so that the power management unit can operate safely. The POR_VDD_DIG insures that VDD_DIG is correct so that the digital core can start.

3.4.4 Bus Error

The address space is assigned as shown in the memory map in Table 3. If the bit EnableBusError is set in the register **RegSysCtrl** and an unused address is accessed by the processor, then a reset is generated.

3.4.5 Watchdog

Once enabled by setting the bit **EnableResetWD** of the **RegSysCtrl** register, a counter will be started and a reset condition (watchdogreset, Figure 9) will be generated when the counter reaches its maximum value, unless the counter is cleared by software. The counter is 3-bit wide and is clocked by the ck2Hz output of the low prescaler. Its period is typically around 4 seconds but will depend on the clock controller configuration. The watchdog is cleared by writing consecutively the values 0x0a and 0x03 in the **RegSysWd** register.
In assembler, the sequence will look like:
move RegSysWd, #0x0a
move RegSysWd, #0x03

Only writing 0x0a followed by 0x03 will clear the watchdog. If some other writing is done in and between, in **RegSysWd**, then the watchdog will not be cleared.
The status of the watchdog may be checked by reading the register **RegSysWd**. The watchdog is a four bit counter with a range of 0 to 7. The reset is generated when the counter reaches the value 8.

3.4.6 Analog Reset Specifications

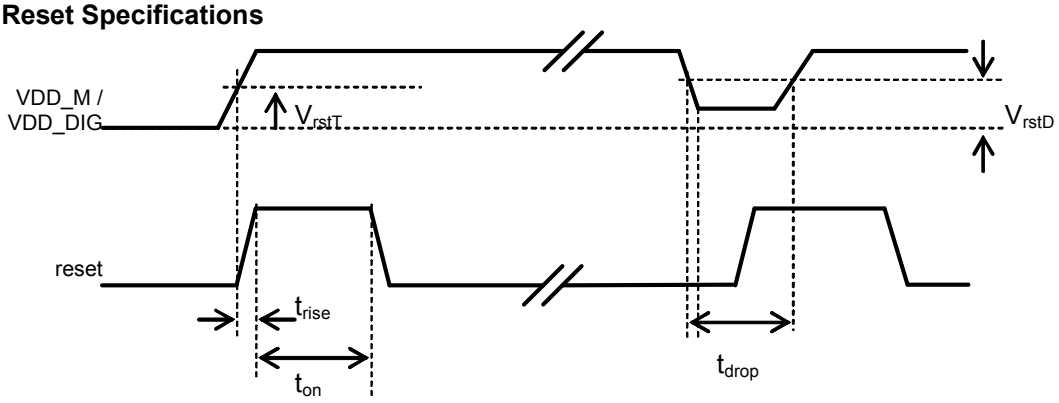


Figure 10 – Power-On / Brownout reset conditions

POR	Symbol	Description	Min	Max	Unit
VDD_M supervision	V_{rstT}	Start Voltage	0.8	1.5	V
	V_{rstD} (*)	Drop Voltage	0.8	1.5	V
	t_{on} (*)	Reset Time	-	300	µs
	t_{rise} (*)	Rise Time	-	15	µs
	t_{drop} (*)	Drop Time	6.0	-	µs
VDD_DIG supervision	V_{rstT}	Start Voltage	0.8	1.5	V
	V_{rstD} (*)	Drop Voltage	0.8	1.5	V
	t_{on} (*)	Reset Time	-	300	µs
	t_{rise} (*)	Rise Time	-	15	µs
	t_{drop} (*)	Drop Time	6.0	-	µs

Note1 : Values above are specified across temperature range unless otherwise specified
Note2 : Values marked with asterisks are not production tested and guaranteed by design.

Table 14 - POR specifications

3.5 CLOCK DISTRIBUTION UNIT

3.5.1 Features

- On-chip RC oscillator
- Three available clock sources: RC oscillator, SYS_CLOCK_IN pin, SLW_CLOCK_IN pin
- Two divider chains: high-prescaler (8 bits) and low-prescaler (15 bits).
- CPU clock disabled in halt mode.

3.5.2 Register Map

Name	Address (Hex)
RegSysClock	0x0012
RegSysMisc	0x0013
RegSysPre0	0x0015
RegSysRcTrim1	0x001B
RegSysRcTrim2	0x001C

Table 15 – Clock distribution registers addresses

Pos	RegSysClock	r/w	Reset	Function
7	CpuSel	rw	0	1 = Low Speed Clock Selected (generally: an external 32kHz clock crystal) 0 = High Speed Clock Selected (the nature of the clock selected will depend on the EnableSysClk bit value).
6	SelLowPresIn	rw	0	1 = Force the SLW_CLOCK_IN clock as the low prescaler input when a low speed clock is available (i.e. EnableSlwClock bit). Otherwise the high prescaler is selected. 0 = Select the SLW_CLOCK_IN clock as the low prescaler input when a low speed clock is available (i.e. EnableSlwClock bit) and the ckRC clock has been selected as the High Speed Clock (i.e. EnableSysClk bit).
5	EnableSysClk	rw	0	1 = SYS_CLOCK_IN clock is selected 0 = ckRC clock is selected
4	-	r	0	reserved
3	ColdSlwClock	r	1	Flag determining when the low speed clock starting phase is finished: 1 = SLW_CLOCK_IN still on the starting phase (32768 cycles) 0 = SLW_CLOCK_IN starting phase finished
2	-	r	0	reserved
1	EnableSlwClock	rw	0	Should be set to '1' when SLW_CLOCK_IN is available, '0' otherwise.
0	EnableRC	rw	1	1 = enable ckRC 0 = disable ckRC

Table 16 - RegSysClock register

Pos	RegSysMisc	r/w	Reset	Function
7:4	-	r	0000	reserved
3:2	-	rw	00	reserved
1	OutputCk32kHz	rw	0	output ck32kHz on pad PB[3]
0	OutputCkCpu	rw	0	output CkCpu on pad PB[2]

Table 17 - RegSysMisc register

Pos	RegSysPre0	r/w	Reset	Function
7:1	-	r	0000000	reserved
0	ResPre	w	0	1 = reset the low prescaler

Table 18 - RegSysPre0 register

Pos	RegSysRcTrim1	r/w	Reset	Function
7:5	-	r	000	reserved
4:2	RCDivFactor	rw	000	Divide RC frequency by $2^{RcDivFactor}$
1:0	RCCoarseMSB	rw	01	RC coarse adjustment (MSB)

Table 19 - RegSysTrim1 register

Pos	RegSysRcTrim2	r/w	Reset	Function
7:6	-	r	00	reserved
5:4	RCCoarseLSB	rw	00	RC coarse adjustment (LSB)
1:0	RCFine	rw	0000	RC fine adjustment

Table 20 - RegSysTrim2 register

3.5.3 RC Oscillator

The RC oscillator is always turned on and selected for CPU and system operation after a power-on reset or a negative pulse on pad NRESET. It can be deselected after the SYS_CLOCK_IN or the SLW_CLOCK_IN has been started and selected as system clock.
The **EnableRC** bit in the register **RegSysClock** controls the signal from the RC oscillator. The user can disable the RC oscillator clock signal by resetting the bit **EnableRC**.
The RC oscillator frequency is trimmed with the registers **RegSysRcTrim1** and **RegSysRcTrim2**. The absolute value of the frequency for a given register content may change from chip to chip due to process tolerances. However, the modification of the frequency as a function of a modification of the register content is fairly precise. The RC oscillator output frequency, f_{RC}, is obtained by the following trimming rule:

$$f_{RC} = fo \cdot \frac{1}{2^{RCDivFactor}} \cdot \left(1 + (RCCoarse - 8) \cdot CoarseStep + RCFine \cdot FineStep\right)$$

Equation 2 - RC oscillator clock frequency

Note : Values marked with asterisks are not production tested and guaranteed by design.

Table 21 summarizes the characteristics of the oscillator.

Symbol	Description	Min	Typ	Max	Unit
f_0 (*)	Internal oscillator frequency	5.5	8.25	11	MHz
FineStep (*)	Fine tuning step	-	0.5		%
CoarseStep (*)	Coarse tuning step	-	7		%

Note : Values marked with asterisks are not production tested and guaranteed by design.

Table 21 – RC oscillator specifications

Important note: the system is not guaranteed to operate properly with a frequency f_{RC} greater than 14 MHz. Setting the RC oscillator over this limit may produce unpredictable results.

3.5.4 SLOW_CLOCK_IN

SLW_CLOCK_IN must be present and conform to the Bluetooth specifications if the Bluetooth sequencer deep-sleep mode is used. It is typically generated by the XE1413 radio chip. Its frequency is 32'000 Hz or 32'768 Hz.

3.5.5 SYS_CLOCK_IN

It is used by the Bluetooth sequencer and the Codec. Its frequency must be 13 MHz with a tolerance of ± 20 ppm. SYS_CLOCK_IN is typically generated by the XE1413 radio chip.

3.5.6 Clock Source Selection

Different clock sources can be selected independently for the application processor and the reset of the system. The clock of the Bluetooth sequencer is hard-coded and can not be chosen by the user.
The RC clock is always selected after power-up or a negative pulse on the NRESET pin. The CPU clock selection is done with the register **RegSysClock** according to the Table 22. Switching from one clock source to another is glitch free. See also Figure 11, Figure 12, and Figure 13.

Clock Sources					Clock Targets			
					CpuCk			
Mode name	EnableSysClock	EnableRC	EnableSlwClock	SelLowPresIn	**CpuSel** = 0	**CpuSel** = 1	High prescaler clock input	Low prescaler clock input
SlwClock	0	0	1	1	SLW_CLOCK_IN	SLW_CLOCK_IN	Off	On
RC	0	1	0	0	ckRC	high prescaler output	ckRC	high prescaler output
RC + SlwClock	0	1	1	1	ckRC	SLW_CLOCK_IN	ckRC	SLW_CLOCK_IN
SysClock	1	0	0	0	SYS_CLOCK_IN	high prescaler output	SYS_CLOCK_IN	high prescaler output
SysClock + SlwClock	1	0	1	1	SYS_CLOCK_IN	SLW_CLOCK_IN	SYS_CLOCK_IN	SLW_CLOCK_IN

Table 22 – SX1441 Clock configuration

Switching from one clock to one other and stopping the unused clock must be performed in three MOVE instructions to **RegSysClock**. First enable the new clock, then select the CPU clock, and finally stop the unused clock. Combining the different operations in one instruction may cause system malfunction.

3.5.7 RegSysMisc Description

When OutputCk32kHz is 1, the ck32kHz clock is output of the port B PB[3]. The CPU clock is output on the port B PB[2] when the bit OutputCkCpu is 1.

3.5.8 Prescalers

The Figure 11 describes the overall structure of the prescaler.

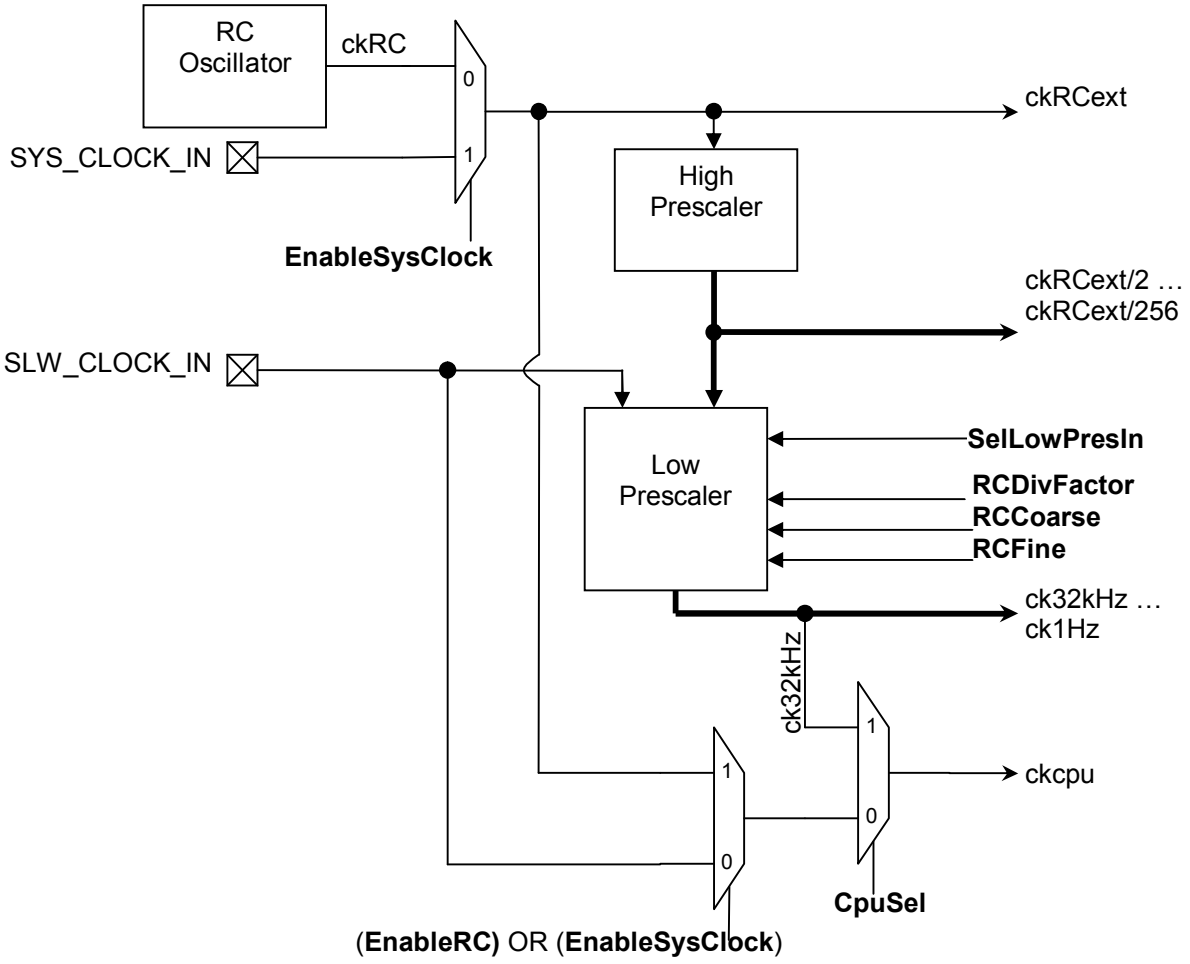


Figure 11 - Prescaler Unit block diagram

3.5.8.1 High Prescaler

The high prescaler is made up of an 8-stage dividing chain. It can be driven with the RC oscillator clock or the SYS_CLOCK pin, depending on the **EnableSysClock** parameter.

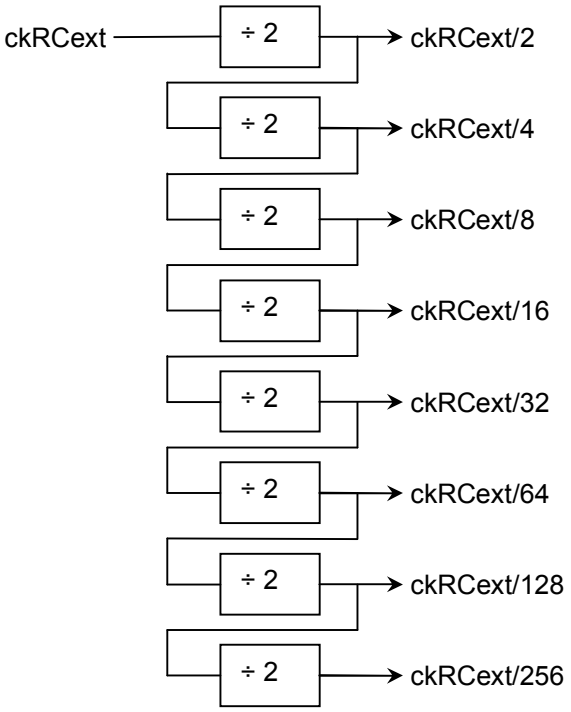


Figure 12 - High prescaler block diagram

The Table 23 summarizes which peripherals use the outputs of the high prescaler. Since each stage of the high prescaler divides the frequency by 2, the frequency of all the outputs of the high prescaler is proportional to the frequency of ckRCext.

High prescaler output	Peripherals
ckRCext	application UART, Bluetooth UART, SPI, counter/timer, port PA
ckRCext/2	application UART, Bluetooth UART
ckRCext/4	application UART, Bluetooth UART, SPI, counter/timer
ckRCext/8	application UART, Bluetooth UART, SPI
ckRCext/16	application UART, Bluetooth UART, SPI
ckRCext/32	application UART, Bluetooth UART
ckRCext/64	application UART, Bluetooth UART
ckRCext/128	application UART, Bluetooth UART

Table 23 - High prescaler outputs usage

3.5.8.2 Low Prescaler

The low prescaler can be driven from one of the high prescaler outputs ckRCext/2 to ckRCext/128 or directly with the SLW_CLOCK_OUT pin when the bit **EnableSlwClock** is set to 1 and bit **EnableSysClock** is reseted to 0. The bit **ResPre** in the register **RegSysPre0** synchronously resets the low prescaler. The low prescaler is also automatically cleared when the bit **EnableSlwClock** is set to 1. The bit **ColdSlwClock** value is 1 to indicate that the SLW_CLOCK_IN is in its starting phase. It automatically enters this phase when the bit **EnableSlwClock** is set to 1. During this phase, SLW_CLOCK_IN is not available. It becomes available after 32'768 cycles, when the bit **ColdSlwClock** returns to 0.

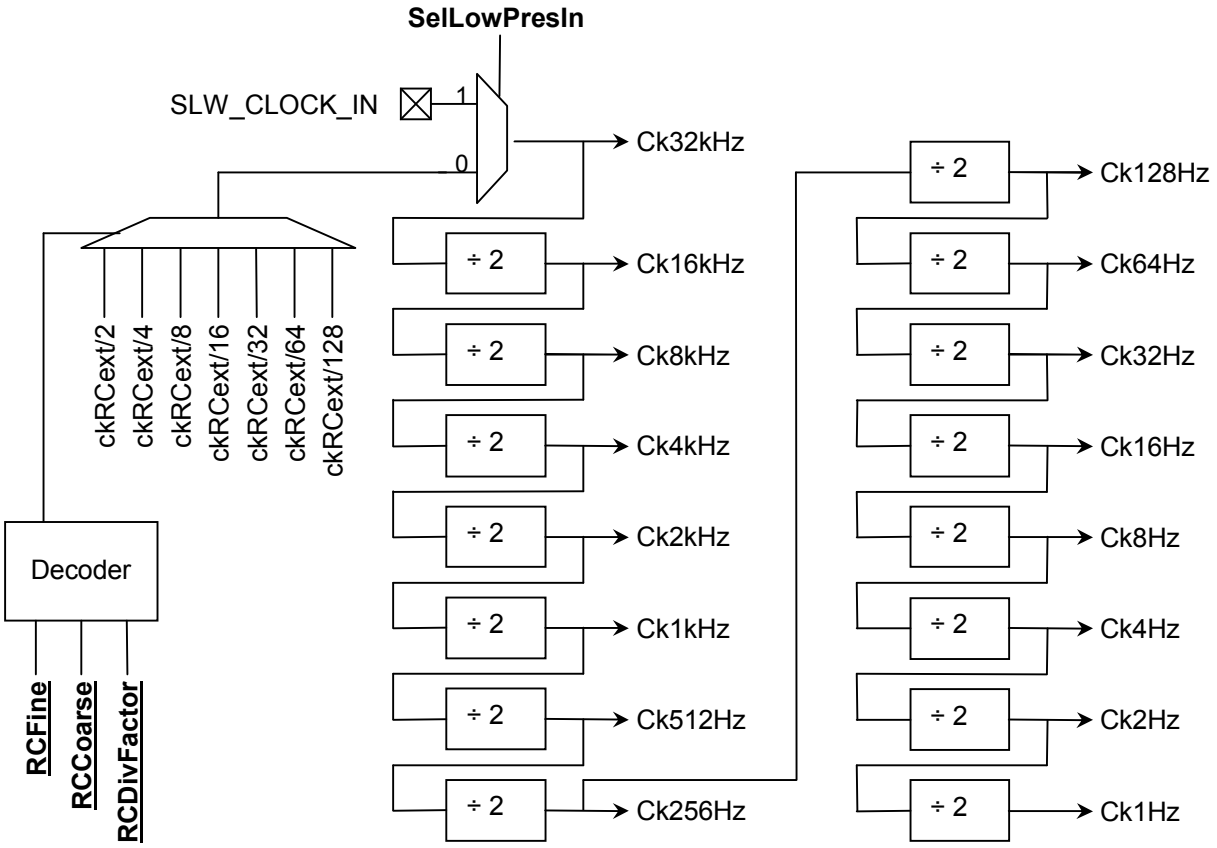


Figure 13 - Low prescaler block diagram

The high prescaler output may be used as the low prescaler input. A decoder is used to select from the high prescaler the frequency tap that is the closest to 32 kHz to operate the low prescaler when SLW_CLOCK_IN is not running. In this case, the RC oscillator frequency will also be valid for the low prescaler frequency outputs.
The Table 24 shows how the RC trimming values in the **RegSysRcTrim1** and **RegSysRcTrim2** registers are decoded to select the input frequency from the high prescaler. The least significant bits of the **RCFine** word are ignored.
In order to ensure the correct frequency selection for the low prescaler with an external clock, a proper value must be set in the RC trim registers. If the frequency is not set correctly, all timings derived from the low prescaler will be shifted accordingly (e.g. watchdog and interrupt frequencies).
In the Table 24, ckRCext stands for either ckRC or SYS_CLOCK_IN.

RCDivFactor & RCCoarseMSB & RCCoarseLSB & RCFine	Selected high prescaler tap	RCDivFactor & RCCoarseMSB & RCCoarseLSB & RCFine	Selected high prescaler tap
[0x0000]hex	ckRCext / 64	[0x04c8]hex	ckRCext / 32
[0x0002]hex	ckRCext / 128	[0x04d0]hex	ckRCext / 16
[0x0100]hex	ckRCext / 32	[0x04d6]hex	ckRCext / 32
[0x0102]hex	ckRCext / 64	[0x04e0]hex	ckRCext / 16
[0x010a]hex	ckRCext / 128	[0x04e5]hex	ckRCext / 32
[0x0110]hex	ckRCext / 64	[0x04f0]hex	ckRCext / 16
[0x0119]hex	ckRCext / 128	[0x04f3]hex	ckRCext / 32
[0x0120]hex	ckRCext / 64	[0x0500]hex	ckRCext / 2
[0x0127]hex	ckRCext / 128	[0x0502]hex	ckRCext / 4
[0x0130]hex	ckRCext / 64	[0x050a]hex	ckRCext / 8

RCDivFactor & RCCoarseMSB & RCCoarseLSB & RCFine	Selected high prescaler tap	RCDivFactor & RCCoarseMSB & RCCoarseLSB & RCFine	Selected high prescaler tap
[0x0135]hex	ckRCext / 128	[0x0510]hex	ckRCext / 4
[0x0140]hex	ckRCext / 64	[0x0519]hex	ckRCext / 8
[0x0144]hex	ckRCext / 128	[0x0520]hex	ckRCext / 4
[0x0150]hex	ckRCext / 64	[0x0527]hex	ckRCext / 8
[0x0152]hex	ckRCext / 128	[0x0530]hex	ckRCext / 4
[0x0160]hex	ckRCext / 64	[0x0535]hex	ckRCext / 8
[0x0161]hex	ckRCext / 128	[0x0540]hex	ckRCext / 4
[0x0200]hex	ckRCext / 16	[0x0544]hex	ckRCext / 8
[0x0202]hex	ckRCext / 32	[0x0550]hex	ckRCext / 4
[0x020a]hex	ckRCext / 64	[0x0552]hex	ckRCext / 8
[0x0210]hex	ckRCext / 32	[0x0560]hex	ckRCext / 4
[0x0219]hex	ckRCext / 64	[0x0561]hex	ckRCext / 8
[0x0220]hex	ckRCext / 32	[0x058e]hex	ckRCext / 16
[0x0227]hex	ckRCext / 64	[0x0590]hex	ckRCext / 8
[0x0230]hex	ckRCext / 32	[0x059d]hex	ckRCext / 16
[0x0235]hex	ckRCext / 64	[0x05a0]hex	ckRCext / 8
[0x0240]hex	ckRCext / 32	[0x05ab]hex	ckRCext / 16
[0x0244]hex	ckRCext / 64	[0x05b0]hex	ckRCext / 8
[0x0250]hex	ckRCext / 32	[0x05b9]hex	ckRCext / 16
[0x0252]hex	ckRCext / 64	[0x05c0]hex	ckRCext / 8
[0x0260]hex	ckRCext / 32	[0x05c8]hex	ckRCext / 16
[0x0261]hex	ckRCext / 64	[0x05d0]hex	ckRCext / 8
[0x028e]hex	ckRCext / 128	[0x05d6]hex	ckRCext / 16
[0x0290]hex	ckRCext / 64	[0x05e0]hex	ckRCext / 8
[0x029d]hex	ckRCext / 128	[0x05e5]hex	ckRCext / 16
[0x02a0]hex	ckRCext / 64	[0x05f0]hex	ckRCext / 8
[0x02ab]hex	ckRCext / 128	[0x05f3]hex	ckRCext / 16
[0x02b0]hex	ckRCext / 64	[0x0600]hex	ckRCext / 1
[0x02b9]hex	ckRCext / 128	[0x0602]hex	ckRCext / 2
[0x02c0]hex	ckRCext / 64	[0x060a]hex	ckRCext / 4
[0x02c8]hex	ckRCext / 128	[0x0610]hex	ckRCext / 2
[0x02d0]hex	ckRCext / 64	[0x0619]hex	ckRCext / 4
[0x02d6]hex	ckRCext / 128	[0x0620]hex	ckRCext / 2
[0x02e0]hex	ckRCext / 64	[0x0627]hex	ckRCext / 4
[0x02e5]hex	ckRCext / 128	[0x0630]hex	ckRCext / 2
[0x02f0]hex	ckRCext / 64	[0x0635]hex	ckRCext / 4
[0x02f3]hex	ckRCext / 128	[0x0640]hex	ckRCext / 2
[0x0300]hex	ckRCext / 8	[0x0644]hex	ckRCext / 4
[0x0302]hex	ckRCext / 16	[0x0650]hex	ckRCext / 2
[0x030a]hex	ckRCext / 32	[0x0652]hex	ckRCext / 4
[0x0310]hex	ckRCext / 16	[0x0660]hex	ckRCext / 2
[0x0319]hex	ckRCext / 32	[0x0661]hex	ckRCext / 4
[0x0320]hex	ckRCext / 16	[0x068e]hex	ckRCext / 8
[0x0327]hex	ckRCext / 32	[0x0690]hex	ckRCext / 4
[0x0330]hex	ckRCext / 16	[0x069d]hex	ckRCext / 8
[0x0335]hex	ckRCext / 32	[0x06a0]hex	ckRCext / 4
[0x0340]hex	ckRCext / 16	[0x06ab]hex	ckRCext / 8
[0x0344]hex	ckRCext / 32	[0x06b0]hex	ckRCext / 4
[0x0350]hex	ckRCext / 16	[0x06b9]hex	ckRCext / 8
[0x0352]hex	ckRCext / 32	[0x06c0]hex	ckRCext / 4

RCDivFactor & RCCoarseMSB & RCCoarseLSB & RCFine	Selected high prescaler tap	RCDivFactor & RCCoarseMSB & RCCoarseLSB & RCFine	Selected high prescaler tap
[0x0360]hex	ckRCext / 16	[0x06c8]hex	ckRCext / 8
[0x0361]hex	ckRCext / 32	[0x06d0]hex	ckRCext / 4
[0x038e]hex	ckRCext / 64	[0x06d6]hex	ckRCext / 8
[0x0390]hex	ckRCext / 32	[0x06e0]hex	ckRCext / 4
[0x039d]hex	ckRCext / 64	[0x06e5]hex	ckRCext / 8
[0x03a0]hex	ckRCext / 32	[0x06f0]hex	ckRCext / 4
[0x03ab]hex	ckRCext / 64	[0x06f3]hex	ckRCext / 8
[0x03b0]hex	ckRCext / 32	[0x0700]hex	ckRCext / 0
[0x03b9]hex	ckRCext / 64	[0x0702]hex	ckRCext / 1
[0x03c0]hex	ckRCext / 32	[0x070a]hex	ckRCext / 2
[0x03c8]hex	ckRCext / 64	[0x0710]hex	ckRCext / 1
[0x03d0]hex	ckRCext / 32	[0x0719]hex	ckRCext / 2
[0x03d6]hex	ckRCext / 64	[0x0720]hex	ckRCext / 1
[0x03e0]hex	ckRCext / 32	[0x0727]hex	ckRCext / 2
[0x03e5]hex	ckRCext / 64	[0x0730]hex	ckRCext / 1
[0x03f0]hex	ckRCext / 32	[0x0735]hex	ckRCext / 2
[0x03f3]hex	ckRCext / 64	[0x0740]hex	ckRCext / 1
[0x0400]hex	ckRCext / 4	[0x0744]hex	ckRCext / 2
[0x0402]hex	ckRCext / 8	[0x0750]hex	ckRCext / 1
[0x040a]hex	ckRCext / 16	[0x0752]hex	ckRCext / 2
[0x0410]hex	ckRCext / 8	[0x0760]hex	ckRCext / 1
[0x0419]hex	ckRCext / 16	[0x0761]hex	ckRCext / 2
[0x0420]hex	ckRCext / 8	[0x078e]hex	ckRCext / 4
[0x0427]hex	ckRCext / 16	[0x0790]hex	ckRCext / 2
[0x0430]hex	ckRCext / 8	[0x079d]hex	ckRCext / 4
[0x0435]hex	ckRCext / 16	[0x07a0]hex	ckRCext / 2
[0x0440]hex	ckRCext / 8	[0x07ab]hex	ckRCext / 4
[0x0444]hex	ckRCext / 16	[0x07b0]hex	ckRCext / 2
[0x0450]hex	ckRCext / 8	[0x07b9]hex	ckRCext / 4
[0x0452]hex	ckRCext / 16	[0x07c0]hex	ckRCext / 2
[0x0460]hex	ckRCext / 8	[0x07c8]hex	ckRCext / 4
[0x0461]hex	ckRCext / 16	[0x07d0]hex	ckRCext / 2
[0x048e]hex	ckRCext / 32	[0x07d6]hex	ckRCext / 4
[0x0490]hex	ckRCext / 16	[0x07e0]hex	ckRCext / 2
[0x049d]hex	ckRCext / 32	[0x07e5]hex	ckRCext / 4
[0x04a0]hex	ckRCext / 16	[0x07f0]hex	ckRCext / 2
[0x04ab]hex	ckRCext / 32	[0x07f3]hex	ckRCext / 4
[0x04b0]hex	ckRCext / 16		
[0x04b9]hex	ckRCext / 32		
[0x04c0]hex	ckRCext / 16		

Table 24 – ck32kHz frequency selector

Table 25 summarizes which peripherals use the outputs of the low prescaler. The frequencies of all outputs of the low prescaler are directly proportional to the frequency of the clock source of the low prescaler.

Low prescaler output	Peripherals
ck32kHz	application UART, Bluetooth sequencer UART, counter/timer, port PA debouncer
ck2kHz	counter/timer
ck1kHz	counter/timer, port PA debouncer

Low prescaler output	Peripherals
ck128Hz	interrupt controller, event controller
ck2Hz	Watchdog
ck1Hz	interrupt controller, event controller

Table 25 - Low prescaler outputs usage

3.5.9 CODEC and Bluetooth Sequencer Clocks

The CODEC can only use SYS_CLOCK_IN since it needs exactly a 13 MHz frequency to meet the Bluetooth audio specifications. The Bluetooth sequencer uses SYS_CLOCK_IN for normal operations and SLW_CLOCK_IN during the low power modes.

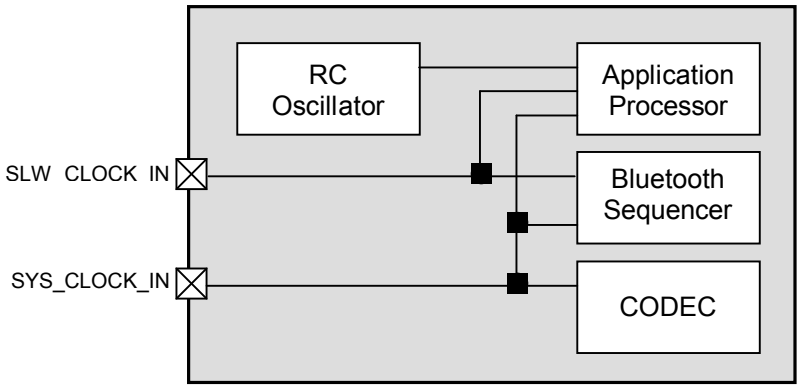


Figure 14 – Codec and Bluetooth Sequencer clocking sources

In a typical application where the SX1441 is used with its companion chip, the XE1413 radio, SYS_CLOCK_IN and SLW_CLOCK_IN are generated by the XE1413.

3.6 INTERRUPT CONTROLLER

3.6.1 Features

- Supports 24 sources of interrupt
- Three levels of priority

.

3.6.2 Register Map

Name	Address (Hex)
RegIrqHig	0x0040
RegIrqMid	0x0041
RegIrqLow	0x0042
RegIrqEnHig	0x0043
RegIrqEnMid	0x0044
RegIrqEnLow	0x0045
ReqIrqPriority	0x0046
ReqIrqIrq	0x0047

Table 26 - Interrupt controller register map

Pos	RegIrqHig	r/w	Reset	Function
7	-	r	0	Reserved
6	128Hz	rc1	0	interrupt from ck128Hz low prescaler output
5	Spi	rc1	0	interrupt from SPI
4	CntA	rc1	0	interrupt from counter A
3	CntC	rc1	0	interrupt from counter C

Pos	RegIrqHig	r/w	Reset	Function
2	Codec	rc1	0	interrupt from Codec
1	HUartTx	rc1	0	interrupt from Bluetooth Sequencer UART transmitter
0	HUartRx	rc1	0	interrupt from Bluetooth Sequencer UART receiver

Table 27 - RegIrqHigh register

Pos	RegIrqMid	r/w	Reset	Function
7	UartTx	rc1	0	interrupt from application UART transmitter
6	UartRx	rc1	0	interrupt from application UART receiver
5	Pa5	rc1	0	interrupt from port PA[5]
4	Pa4	rc1	0	interrupt from port PA[4]
3	1Hz	rc1	0	interrupt from ck1Hz low prescaler output
2	Wakeup	rc1	0	interrupt from WAKEUP pin
1	Pa1	rc1	0	interrupt from port PA[1]
0	Pa0	rc1	0	interrupt from port PA[0]

Table 28 - RegIrqMid register

Pos	RegIrqLow	r/w	Reset	Function
7	Pa7	rc1	0	interrupt from port PA[7]
6	Pa6	rc1	0	interrupt from port PA[6]
5	CntB	rc1	0	interrupt from counter B
4	CntD	rc1	0	interrupt from counter D
3	Pa3	rc1	0	interrupt from port PA[3]
2	Pa2	rc1	0	interrupt from port PA[2]
1	HUartFlowCtrl	rc1	0	interrupt from Bluetooth Sequencer UART flow control
0	UartFlowCtrl	rc1	0	interrupt from application UART flow control

Table 29 - RegIrqLow register

Pos	RegIrqEnHig	r/w	Reset	Function
7	-	r	0	reserved
6	En128Hz	rw	0	enable interrupt from ck128Hz low prescaler output
5	EnSpi	rw	0	enable interrupt from SPI
4	EnCntA	rw	0	enable interrupt from counter A
3	EnCntC	rw	0	enable interrupt from counter C
2	EnCodec	rw	0	enable interrupt from Codec
1	EnHUartTx	rw	0	enable interrupt from Bluetooth Sequencer UART transmitter
0	EnHUartRx	rw	0	enable interrupt from Bluetooth Sequencer UART receiver

Table 30 - RegIrqEnHig register

Pos	RegIrqEnMid	r/w	Reset	Function
7	EnUartTx	rc1	0	enable interrupt from application UART transmitter
6	EnUartRx	rc1	0	enable interrupt from application UART receiver
5	EnPa5	rc1	0	enable interrupt from port PA[5]
4	EnPa4	rc1	0	enable interrupt from port PA[4]
3	En1Hz	rc1	0	enable interrupt from ck1Hz low prescaler output

Pos	RegIrqEnMid	r/w	Reset	Function
2	EnWakeup	rc1	0	enable interrupt from WAKEUP pin
1	EnPa1	rc1	0	enable interrupt from port PA[1]
0	EnPa0	rc1	0	enable interrupt from port PA[0]

Table 31 - RegIrqEnMid register

Pos	RegIrqEnLow	r/w	Reset	Function
7	EnPa7	rc1	0	interrupt from port PA[7]
6	EnPa6	rc1	0	interrupt from port PA[6]
5	EnCntB	rc1	0	interrupt from counter B
4	EnCntD	rc1	0	enable interrupt from counter D
3	EnPa3	rc1	0	enable interrupt from port PA[3]
2	EnPa2	rc1	0	enable interrupt from port PA[2]
1	EnHUartFlowCtrl	rc1	0	enable interrupt from Bluetooth Sequencer UART flow control
0	EnUartFlowCtrl	rc1	0	enable interrupt from application UART flow control

Table 32 - RegIrqEnLow register

Pos	RegIrqPriority	r/w	Reset	Function
7:0	IrqPriority	r	11111111	Number of the highest priority interrupt set

Table 33 - RegIrqPriority register

Pos	RegIrqIrq	r/w	Reset	Function
7:3	-	r	00000	reserved
2	HighIrqTriggered	r	0	1 = one or more high priority interrupt have been triggered
1	MidIrqTriggered	r	0	1 = one or more mid priority interrupt have been triggered
0	LowIrqTriggered	r	0	1 = one or more low priority interrupt have been triggered

Table 34 – RegIrqIrq

3.6.3 Operation

The SX1441 supports 24 sources of interrupt, divided into 3 levels of priority: high (8 sources of interrupt), middle (8 sources of interrupt), and low (8 sources of interrupt). All sources of interrupt are sampled by the highest frequency available in the system. A CPU interrupt is generated and memorized when an interrupt source is triggered. The three levels of priority are directly mapped to those supported by the CoolRISC (IN0, IN1 and IN2; see the CoolRISC documentation for more information on the interrupt processing).

RegIrqHig, **RegIrqMid**, and **RegIrqLow** are 8-bit registers containing flags for the interrupt sources. Those flags are set when the interrupt is enabled (i.e. if the corresponding bit in the registers **RegIrqEnHig**, **RegIrqEnMid** or **RegIrqEnLow** is set) and a rising edge is detected on the corresponding interrupt source.

Once memorized, an interrupt flag can be cleared by writing a '1' in the corresponding bit of **RegIrqHig**, **RegIrqMid** or **RegIrqLow**. Writing a '0' does not modify the flag. To definitively clear the interrupt, one has to clear the CoolRISC interrupt in the CoolRISC status register in addition to cleaning the corresponding RegIrq register. All interrupts are automatically cleared after a reset.

Two registers are provided to facilitate the writing of interrupt service software. **RegIrqPriority** contains the number of the highest priority set (its value is 0xFF when no interrupt is memorized). **RegIrqIrq** indicates the priority level of the current interrupts.

3.7 EVENT CONTROLLER

3.7.1 Features

- Supports 8 sources of events
- Two levels of priority

3.7.2 Register Map

Name	Address (Hex)
RegEvn	0x003C
RegEvnEn	0x003D
RegEvnPriority	0x003E
RegEvnEvn	0x003F

Table 35 - Event controller registers

Pos	RegEvn	r/w	Reset	Function
7	CntA	rc1	0	event from counter A (high priority)
6	CntC	rc1	0	event from counter C (high priority)
5	-	r	0	reserved
4	Pa1	rc1	0	event from port PA[1] (high priority)
3	CntB	rc1	0	event from counter B (low priority)
2	CntD	rc1	0	event from counter D (low priority)
1	1Hz	rc1	0	event from ck1Hz low prescaler output
0	Pa01	rc1	0	event from port PA[0]

Table 36 - RegEvn register

Pos	RegEvnEn	r/w	Reset	Function
7	EnCntA	rw	0	enable event from counter A (high priority)
6	EnCntC	rw	0	enable event from counter C (high priority)
5	-	r	0	reserved
4	EnPa1	rw	0	enable event from port PA[1] (high priority)
3	EnCntB	rw	0	enable event from counter B (low priority)
2	EnCntD	rw	0	enable event from counter D (low priority)
1	En1Hz	rw	0	enable event from ck1Hz low prescaler output
0	EnPa0	rw	0	enable event from port PA[0]

Table 37 - RegEvnEn register

Pos	RegEvnPriority	r/w	Reset	Function
7:0	EvnPriority	r	00000000	number of the highest event triggered

Table 38 - RegEvnPriority register

Pos	RegEvnEvn	r/w	Reset	Function
7:2	-	r	000000	reserved
1	EvnHigh	r	0	1 = one or more high priority event have been triggered
0	EvnLow	r	0	1 = one or more low priority event have been triggered

Table 39 - RegEvnEvn register

3.7.3 Operation

The SX1441 supports 8 event sources, divided into 2 levels of priority. All sources of event are sampled by the highest frequency available in the system. A CPU event is generated and memorized when an event becomes source is triggered. The 8 sources of event are divided into 2 levels of priority: High (4 sources of event) and Low (4 sources of event). Those 2 levels of priority are directly mapped to those supported by the CoolRISC (EV0 and EV1; see CoolRISC documentation for more information on event processing).

RegEvn is an 8-bit register containing flags for the sources of event. Those flags are set when the event is enabled (i.e. if the corresponding bit in the registers **RegEvnEn** is set) and a rising edge is detected on the corresponding event source. Once memorized, writing a '1' in the corresponding bit of **RegEvn** clears the event flag. Writing a '0' does not modify the flag. All events are automatically cleared after a reset.

Two registers are provided to facilitate the writing of interrupt service software. **RegEvnPriority** contains the number of the highest event set (its value is 0xFF when no event is memorized). **RegEvnEvn** indicates the priority level of the current pending events.

3.8 DIGITAL INPUT PORT PA[7:0]

3.8.1 Features

- Input port, 8-bit wide
- Each bit can be programmed individually for debounced or direct input, with pull-up or not
- Snap-to-rail option for each input
- Each bit can be configured as a source of interrupt on the rising or falling edge
- A system reset can be generated on an input pattern
- PA[0] and PA[1] can be configured to generate two events
- PA[0] to PA[3] can be used as clock inputs for the counters/timers/PWM

3.8.2 Register Map

Name	Address (Hex)
RegPAIn	0x0020
RegPADebounce	0x0021
RegPAEdge	0x0022
RegPAPullup	0x0023
RegPARes0	0x0024
RegPARes1	0x0025
RegPACtrl	0x0026
RegPASnapToRail	0x0027

Table 40 - PA registers

Pos	RegPAIn	r/w	Reset	Function
7:0	PAIn	r	xxxxxxxx	value of pads PA[7:0]

Table 41 - RegPAIn register

Pos	RegPADebounce	r/w	Reset	Function
7:0	PADebounce	rw	00000000	1 = debouncer enabled (for each corresponding PA pad) 0 =debouncer disabled (for each corresponding PA pad)

Table 42 - RegPADebounce register

Pos	RegPAEdge	r/w	Reset	Function
7:0	PAEdge	rw	00000000	0 = positive edge (for each corresponding PA pad) 1 = negative edge (for each corresponding PA pad)

Table 43 - RegPAEdge register

Pos	RegPAPullup	r/w	Reset	Function
7:0	PAPullup	rw	11111111	1 = pull-up enabled (for each corresponding PA pad) 0 = pull-up disabled (for each corresponding PA pad)

Table 44 - RegPAPullup register

Pos	RegPARes0	r/w	Reset	Function
7:0	PARes0	rw	00000000	for each corresponding PA pad: bit 0 of reset configuration (see 3.8.9)

Table 45 - RegPARes0 register

Pos	RegPARes1	r/w	Reset	Function
7:0	PARes1	rw	00000000	for each corresponding PA pad: bit 1 of reset configuration (see 3.8.9)

Table 46 - RegPARes1 register

Pos	RegPACtrl	r/w	Reset	Function
7:1	-	rw	0000000	reserved
0	DebounceSelect	rw	0	1 = fast debounce clock selected 0 = slow debounce clock selected

Table 47 - RegPACtrl register

Pos	RegPASnapToRail	r/w	Reset	Function
7:0	SnapToRail	rw	00000000	1 = snap-to-rail mode enabled (for each corresponding PA pad) 0 =snap-to-rail mode disabled (for each corresponding PA pad)

Table 48 - RegPASnapToRail register

3.8.3 Block Diagram

Figure 15 shows the block diagram of the port PA.

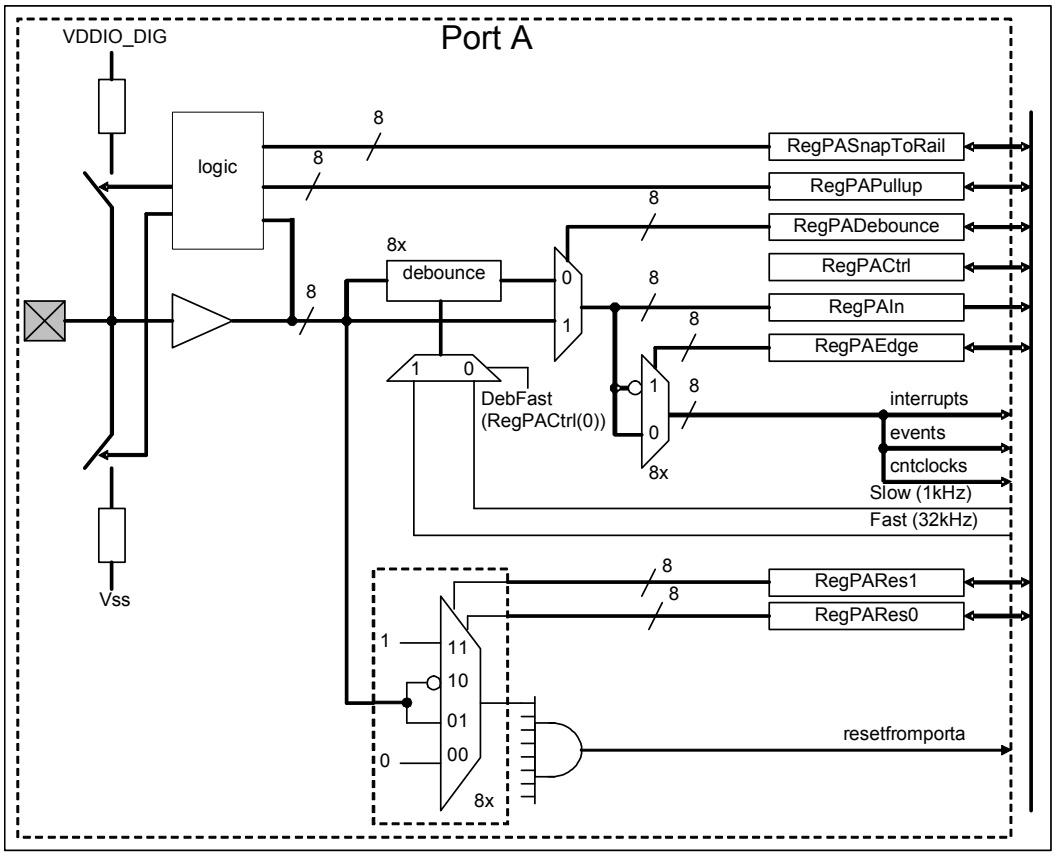


Figure 15 - Structure of PA[7:0]

3.8.4 Debounce Mode

Each bit of the port PA can be individually debounced by setting the corresponding bit in **RegPADebounce**. After reset, the debounce function is disabled. After enabling the debouncer, the change of the input value is accepted only if eight consecutive samples are identical. Selection of the clock is done by the bit **DebounceSelect** in register **RegPACtrl**.

DebounceSelect	Debounce filter clock
0	slow (ck1kHz low prescaler output)
1	Fast (ck32kHz low prescaler output)

Table 49 - Debouncer clock selection

3.8.5 Pull-ups/Snap-to-rail

Different functions are possible depending on the value of the registers **RegPAPullup** and **RegPASnapToRail**. When the corresponding bit in **RegPAPullup** is cleared, the inputs are floating (pull-up and pull-down resistors are disconnected). When the corresponding bits are set in **RegPAPullup** is 1 and cleared in **RegPASnapToRail** , a pull-up resistor is connected to the input pin. Alternatively, when the corresponding bits are cleared in **RegPAPullup** and set in **RegPASnapToRail**, the snap-to-rail function is active.

The snap-to-rail function connects a pull-up or pull-down resistor to the input pin depending on the value last forced on the input pin. This function can be used for instance when the input port is connected to a tri-state bus. When the bus is floating, the pull-up or pull-down maintains the bus in the last low impedance state before it became floating until another low impedance output drives the bus. It also reduces the power consumption with respect to a classic pull-up since it selects the pull-up or pull-down resistor that matches the detected input state.

The state of input pin is summarized in Table 50.

RegPAPullup[i]	RegPASnapToRail[i]	Last externally forced PA[i] value	PA[i] pull
0	X	X	none (floating)
1	0	X	pull-up
1	1	0	pull-down
1	1	1	pull-up

Table 50 - PA pin state vs. RegPAPullup and RegPASnapToRail registers

The port PA starts up with the pull-up resistor connected and the snap-to-rail function disabled.

3.8.6 Interrupt Sources

Every PA port input is an interrupt source which can be enabled on a rising or falling edge with the corresponding bit in **RegPAEdge**. After reset, the rising edge is selected for interrupt generation. The interrupt source can be debounced by setting register **RegPADebounce**. The interrupt signals are sampled on the fastest clock in the circuit. In order to guarantee that the interrupt is detected by the circuit, the minimal pulse length should be 1 cycle of this clock. Care must be taken when modifying **RegPAEdge** because this register performs an edge selection. The change of this register may result in a transition which may be interpreted as a valid interruption if the corresponding interrupt sources are not temporarily disabled in the interrupt controller.

3.8.7 Event Sources

Pins PA[0] and PA[1] are also available as events on the event controller.

3.8.8 Clock Sources

PA[0] to PA[3] input ports (debounced or not) are available as clock sources for the counter/timer/PWM peripherals.

3.8.9 Reset Sources

The PA port can be configured to generate a system reset when a predetermined word is detected on PA[7:0]. The reset is built using a logical AND of the 8 **PAReset[i]** signals:
resetfromportA = **PAReset[7]** AND **PAReset[6]** AND **PAReset[5]** AND ... AND **PAReset[0]**
PAReset[i] is itself a logical function of the corresponding pin PA[i]. One of four logical functions can be selected for each pin by writing into two registers **RegPARes0** and **RegPARes1** as shown in Table 51.

PARes1[i]	PARes0[x]	PAReset[i]
0	0	0
0	1	PA[i]
1	0	not PA[i]
1	1	1

Table 51 - PAReset generation

A reset from port PA can be inhibited by placing a 0 on both **PARes1[i]** and **PARes0[i]** for at least 1 pin. Setting both **RegPARes1[i]** and **RegPARes0[i]** to 1, makes the reset independent of the value on the corresponding pin. Setting both registers to 0xff, will reset the circuit independently of the PA input value. This makes it possible to generate a software reset. Depending on the value of PA[0] to PA[7], the change of **RegPARes0** and **RegPARes1** can cause a reset. Therefore it is safe to always have one (**RegPARes0[i], RegPARes1[i]**) equal to 0x00 during the setting operations.

3.9 DIGITAL INPUT/OUTPUT PORT PB[7:0]

3.9.1 Features

- 8-bit wide input/output port
- Each bit can be configured as input or output
- Each bit can be configured as open-drain or push-pull
- A pull-up can be enabled on each bit
- In open-drain mode, the pull-up is not active when corresponding pad is set to zero
- Two internal freq. (ck32kHz and CkCpu) can be output on PB[2] and PB[3]
- Two PWM signals can be driven on pads PB[0] and PB[1]
- The UART interface uses PB[7:4] for UA_RX, UA_TX, UA_RTS and UA_CTS respectively

3.9.2 Register map

Name	Address (Hex)
RegPBOut	0x0028
RegPBIn	0x0029
RegPBDir	0x002A
RegPBOpen	0x002B
RegPBPullup	0x002C

Table 52 - port PB registers

Pos	RegPBout	r/w	Reset	Function
7:0	PBOut	rw	00000000	port output value

Table 53 - RegPBOut register

Pos	RegPBIn	r/w	Reset	Function
7:0	PBIn	r	xxxxxxxx	input value, read from pads PB[7:0]

Table 54 - RegPBIn register

Pos	RegPBDir	r/w	Reset	Function
7:0	PBDir	rw	00000000	for each corresponding PB pad: 1 = pad is configured as digital output 0 = pad is configured as digital input

Table 55 - RegPBDir register

Pos	RegPBOpen	r/w	Reset	Function
7:0	PBOpen	rw	00000000	for each corresponding PB pad: 1 = pad is configured as open drain 0 = pad is configured as push-pull

Table 56 - RegPBOpen register

Pos	RegPBPullup	r/w	Reset	Function
7:0	PBPullup	rw	11111111	for each corresponding PB pad: 1 = pull-up enabled 0 = pull-up disabled

Table 57 - RegPBPullup

3.9.3 Multiplexing PB With Other Peripherals

Port PB acts as a GPIO port by default. This functionality can be overridden as other functions are enabled as shown in Table 58.

Port PB number	Priority	
	Medium	Low
7	UA_RX	GPIO
6	UA_TX	GPIO
5	UA_RTS	GPIO
4	UA_CTS	GPIO
3	ck32k	GPIO
2	CPU clock	GPIO
1	counter C (C+D) PWM1	GPIO
0	counter A (A+B) PWM0	GPIO

Table 58 - PB[7:0] usage

When the counters are used to implement a PWM function (see 3.10), the PB[0] and PB[1] terminals are used as outputs (PB[0] is used if bit 0 in **RegCntConfig1** is set to 1, PB[1] is used if bit 1 in **RegCntConfig1** is set to 1) and the PWM generated values override the values written in **RegPBout**. However, **RegPBDir[0]** and **RegPBDir[1]** are not automatically overwritten and have to be set to 1.
If bit 1 is set in **RegSysMisc**, the ck32kHz low prescaler output is output on PB[3]. This overrides the value contained in **RegPBOut[3]**. However, **RegPBDir[3]** must be set to 1. The frequency and duty cycle of the clock signal are given in Figure 16.

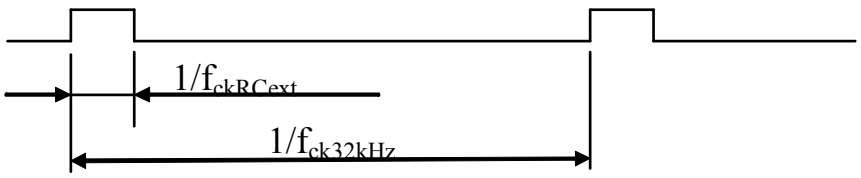


Figure 16 - ck32k output clock timing

Similarly, If bit 0 is set in **RegSysMisc**, the CPU clock is output on PB[2] as described on Figure 17. This overrides the value contained in **RegPBOut[2]**. However, **RegPBDir[2]** must be set to 1.

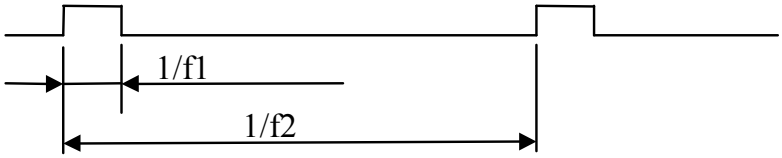


Figure 17 - CPU output clock timing

The timing of the CPU clock depends on the selection of the bit **CpuSel** in the **RegSysClock** register and is given in Table 59.

CpuSel	f_1	f_2
0	$f_{ckRCext}/4$	$f_{ckRCext}$
1	$f_{ckRCext}$	$f_{ck32kHz}$

Table 59 - CPU clock on PB[2] timing

3.9.4 Port B Digital Capabilities

The direction of each bit within PB[7:0] (input only or input/output) can be individually set using the **RegPBDir** register. If **RegPBDir[i]** = 1, both the input and output buffers are active on the corresponding pin. If **RegPBDir[i]** is 0, the corresponding PB pin is an input only and the output buffer is in high impedance. After reset PB is in input only mode; **RegPBDir[i]** is reset to 0.

The input values of PB are available in **RegPBIn** (read only). Reading is always direct - there is no debounce function. In case of possible noise on input signals, a software debouncer with polling or an external hardware filter has to be realized. The input buffer is also active when the port is defined as output and allows reading back of the effective value on the pin. Data stored in **RegPBOut** are output at Port B if **RegPBDir[x]** is 1. The default value after reset is low (0).

When a pin is in output mode (**RegPBDir[i]** is set), the output can be a conventional CMOS (Push-Pull) or an N-channel Open-drain, driving the output low. By default, after reset the **RegPBOpen** is cleared (push-pull). If **RegPBOpen[i]** is set the internal P-channel transistor in the output buffer is electrically removed and the output can only be driven low with **RegPBOut[i]** cleared, or be high-impedance when **RegPBOut[i]** is set. The internal pull-up or an external pull-up resistor can be used to drive the pin high. Because the P-channel transistor actually exists (this is not a real Open-drain output) the pull-up range is limited to VDDIO_DIG + 0.2V (avoid forward bias of the P transistor / diode).

An optional pull-up can be connected to every bit by configuring **RegPBPullup**. Input is pulled up when its corresponding bit in this register is set. Default status after reset is 1, which means with pull-up. To limit power consumption, pull-up resistors are only enabled when the associated pin is either a digital input or an N-channel open-drain output with the pad set (n-channel transistor disabled). In the other cases (push-pull output or open-drain output driven low), the pull up resistors are disabled independently from **RegPBPullup**. After power-on reset, the Port B is configured as an input port with pull-up activated.

The input buffer is always active. This means that the PB input should be a valid digital value at all time. An unused pin should be configured as input pull-up or output. Violating this rule may lead to high power consumption.

3.10 COUNTERS/TIMERS

3.10.1 Features

- 4 x 8-bits timer/counter modules or 2 x 16-bits timers/counter modules, each with 4 possible clock sources
- Up/down counter modes
- Interrupt and event generation
- Capture function (internal or external source)
- Rising, falling or both edge of capture signal (except for ck32k, only rising edge)
- PA[3:0] can be used as clock inputs (debounced or direct, frequency divided by 2 or not)
- 2 x 8 bits PWM or 2 x 16 bits PWM
- PWM resolution of 8, 10, 12, 14 or 16 bits

- Complex mode combinations are possible

3.10.2Register Map

Name	Address (Hex)
RegCntA	0x0058
RegCntB	0x0059
RegCntC	0x005A
RegCntD	0x005B
RegCntCtrlCk	0x005C
RegCntConfig1	0x005D
RegCntConfig2	0x005E
RegCntOn	0x005F

Table 60 - Counter Registers

Pos	RegCntA	r/w	Reset	Function
7:0	CntA	rw	00000000	counter A (1)

Table 61 - RegCntA register

Pos	RegCntB	r/w	Reset	Function
7:0	CntB	rw	00000000	counter B (1)

Table 62 - RegCntB registers

Pos	RegCntC	r/w	Reset	Function
7:0	CntC	rw	00000000	counter C (2)

Table 63 - RegCntC register

Pos	RegCntD	r/w	Reset	Function
7:0	CntD	rw	00000000	counter D (2)

Table 64 - RegCntD register

Pos	RegCntCtrlCk	r/w	Reset	Function
7:6	CntDCkSel	rw	00	counter D clock selection
5:4	CntCCkSel	rw	00	counter C clock selection
3:2	CntBCkSel	rw	00	counter B clock selection
1:0	CntACkSel	rw	00	counter A clock selection

Table 65 - RegCntCtrlCk register

Pos	RegCntConfig1	r/w	Reset	Function
7	CntDDownUp	rw	0	1 = counter D counting up 0 = counter D counting down
6	CntCDownUp	rw	0	1 = counter C counting up 0 = counter C counting down
5	CntBDownUp	rw	0	1 = counter B counting up 0 = counter B counting down

Pos	RegCntConfig1	r/w	Reset	Function
4	CntADownUp	rw	0	1 = counter A counting up 0 = counter A counting down
3	CascadeCD	rw	0	1 = cascade counters C and D 0 = do not cascade counters C and D
2	CascadeAB	rw	0	1 = cascade counter A and B 0 = do not cascade counters A and B
1	CntPWM1	rw	0	1 = counter C (or C + D) PWM enabled 0 = counter C (or C + D) PWM disabled
0	CntPWM0	rw	0	1 = counter A (or A + B) PWM enabled 0 = counter A (or A + B) PWM disabled

Table 66 - RegCntConfig1 register

Pos	RegCntConfig2	r/w	Reset	Function
7:6	CapSel	rw	00	capture source selection
5:4	CaptFunc	rw	00	capture function selection
3:2	Pwm1Size	rw	00	PWM1 size selection
1:0	Pwm0Size	rw	00	PWM0 size selection

Table 67 - RegCntConfig2 register

Pos	RegCntOn	r/w	Reset	Function
7	CntDExtDiv	rw	0	1 = divide external clock PA[3] by 2 0 = do not divide
6	CntCExtDiv	rw	0	1 = divide external clock PA[2] by 2 0 = do not divide
5	CntBExtDiv	rw	0	1 = divide external clock PA[1] by 2 0 = do not divide
4	CntAExtDiv	rw	0	1 = divide external clock PA[0] by 2 0 = do not divide
3	CntDEnable	rw	0	1 = counter D enabled 0 = counter D disabled
2	CntCEnable	rw	0	1 = counter C enabled 0 = counter C disabled
1	CntBEnable	rw	0	1 = counter B enabled 0 = counter B disabled
0	CntAEnable	rw	0	1 = counter A enabled 0 = counter A disabled

(1) When writing to **RegCntA** or **RegCntB**, the processor writes the counter comparison values. When reading these locations, the processor reads back either the actual counter value or the last captured value if the capture mode is active.

(2) When writing **RegCntC** or **RegCntD**, the processor writes the counter comparison values. When reading these locations, the processor reads back the actual counter value.

Table 68 - RegCntOn register

3.10.3General Operation Overview

Counter A and Counter B are 8-bit counters which can be cascaded to form 16-bit counters. Counter C and Counter D have the same features. The counters can also be used to generate two PWM outputs on PB[0] and PB[1]. PWM signals can be generated with 8-, 10-, 12-, 14- or 16-bit precision.

Counters A and B can be captured by events on an internal or an external signal. The capture can be performed on both 8-bit counters running individually on two different clock sources or on both cascaded counters to form a 16-bit counter. In any case, the same capture signal is used for both counters. When the counters A and B are not cascaded, they can be used in several configurations: A and B as counters, A and B as captured counters, A as

PWM and B as counter, A as PWM and B as captured counter. When counters C and D are not cascaded, both can be used either as counters or counter C as PWM and counter D as counter.
Counters are enabled by **RegCntOn**. When counters are cascaded, the bit **CntBEnable** controls the counter A + B, and **CntDEnable** controls the counter C + D.
All counters have a corresponding 8-bit read/write register: **RegCntA**, **RegCntB**, **RegCntC**, and **RegCntD**. When read, these registers contain the counter value (or the captured counter value). When written, they modify the counter comparison values. It is possible to read any counter at any time, even when the counter is running. The value is guaranteed to be correct when the counter is running on an internal clock source. For correct acquisition of the counter value when running on an external clock source, use one of the three following methods:

1) For slow operating counters (typically at least 8 times slower than the CPU clock), over-sample the counter content and perform a majority operation on the consecutive read results to select the correct actual content of the counter.
2) Stop the concerned counter, perform the read operation and restart the counter. While stopped, the counter content is frozen and the counter does not take into account the clock edges delivered on the external pin.
3) Use the capture mechanism.

When a value is written into the counter register while the counter is in counter mode, both the comparison value is updated and the counter value is modified. In upcount mode, the register value is reset to zero. In downcount mode, the comparison value is loaded into the counter. Due to the synchronization mechanism between the processor clock domain and the external clock source domain, this modification of the counter value can be postponed until the counter is enabled and receives its first valid clock edge.
In PWM mode or in capture mode, the counter value is not modified by the write operation in the counter register. Changing to counter mode does not update the counter value (no reset in upcount, no load in downcount mode).

3.10.4Clock Selection

The clock source for each counter can be individually selected by writing the appropriate value in the register **RegCntCtrlCk**. Table 69 gives the correspondence between the binary codes used for the configuration bits **RegCntCtrlCk[1:0]**, **RegCntCtrlCk[3:2]**, **RegCntCtrlCk[5:4]** or **RegCntCtrlCk[7:6]** and the clock source selected respectively for the counters A, B, C or D.

RegCntCtrlCk[i:j]	**Clock source for**			
	Counter A	Counter B	Counter C	Counter D
11	128 Hz timer from clock controller			
10	ckRCext / 4		ck1kHz low prescaler output	
01	ckRCext		ck32kHz low prescaler output	
00	PA[0]	PA[1]	PA[2]	PA[3]

Table 69 - Counter clock selection

See chapter 3.8.8 for details about the different clock sources.
Four external clocks may be provided to the counters through pins PA[3:0]. Optionally, the external clock sources can be debounced by configuring the port PA. Additionally, the external clocks may be divided by 2 by configuring **RegCntOn[7:4].**
Switching between an internal and an external clock source can only be performed while the counter is stopped. Enabling or disabling the external clock frequency division can only happen when the counter using this clock is stopped, or when this counter is running on an internal clock source.

3.10.5Mode Selection

Each counter can be configured in the following modes:
Counter
Capture
PWM
Captured PWM

The counter mode is set by writing the registers **RegCntConfig1** and **RegCntConfig2**.

RegCnt-Config1 [2]	RegCnt-Config1 [0]	RegCnt-Config2 [5:4]	Counter A mode	Counter B mode	Counter A IRQ source	Counter B IRQ source	PB[0] function
0	0	00	Counter 8b Downup: A	Counter 8b Downup: B	Counter A	Counter B	PB[0]
1	0	00	Counter 16b AB Downup: A		Counter AB	-	PB[0]
0	1	00	PWM 8b Downup: A	Counter 8b Downup: B	-	Counter B	PWM A
1	1	00	PWM 10 – 16b AB Downup A		-	-	PWM AB
0	0	1X or X1	Captured counter 8b Downup: A	Captured counter 8b Downup: B	Capture A	Capture B	PB[0]
1	0	1X or X1	Captured counter 16b AB Downup: A		Capture AB	Capture AB	PB[0]
0	1	1X or X1	Captured PWM 8b Downup: A	Captured counter 8b Downup: B	Capture A	Capture B	PWM A
1	1	1X or X1	Captured 10 – 16b PWM (captured value on 16b) Downup: A		Capture AB	Capture AB	PWM AB

Table 70 – Counters A&B operation modes

Switching between different modes must be done while the concerned counters are stopped. While switching capture mode on and off, unwanted interrupts can appear on the interrupt channels concerned by this mode change.

The Table 71 shows the operation modes for counters C and D as a function of the mode control bits.

RegCntConfig1 [3]	RegCntConfig1 [1]	Counter C mode	Counter D mode	Counter C IRQ source	Counter D IRQ source	PB[1] function
0	0	Counter 8b Downup: C	Counter 8b Downup: D	Counter C	Counter D	PB(1)
1	0	Counter 16b CD Downup: C		Counter CD	-	PB(1)
0	1	PWM 8b Downup: C	Counter 8b Downup: D	-	Counter D	PWM C
1	1	PWM 10 – 16b CD Downup: C		-	-	PWM CD

Table 71 - Counters C&D: operation modes

3.10.6Counter / Timer Mode

The counters in counter / timer mode are used to generate interrupts after a predefined number of clock periods applied on the counter clock input have elapsed.
Each counter can be set individually either in upcount mode by setting bit 4 to 7 in register **RegCntConfig1** or in downcount mode by resetting these bit. Counters A and B can be cascaded to behave as a 16-bit counter by setting **RegCntConfig1[2]**. Counters C and D can be also cascaded by setting **RegCntConfig1[3]**. When cascaded, the up/down count modes of counters B and D are defined respectively by the up/down count modes set for the counters A and C.
When in upcount mode, the counter will start incrementing from zero up to the target value which has been written in the corresponding **RegCntX** register(s). When the counter content is equal to the target value, an interrupt is

generated at the next counter clock pulse and the counter is loaded again with the zero value as described in Figure 18.
When in downcount mode, the counter will start counting down from the initial load value which has been written in the corresponding **RegCntX** register(s) down to the zero value. Once the counter content is equal to zero, an interrupt is generated at the next counter clock pulse and the counter is loaded again with the load value as described in Figure 18.
The counter must be configured (capture, PWM, cascade, up/down counting mode) before writing any target value to **RegCntX** register(s). This ensures that the counter will start from the correct initial value. When counters are cascaded, both counter registers must be written to ensure that both cascaded counters will start from the correct initial values.
Stopping and restarting a counter in counter mode without reloading a target or load value write can generate an unwanted interrupt if this counter has been stopped at the zero value (downcount) or at it is target value (upcount). This interrupt has already been generated when the counter has reached the zero or the target value.

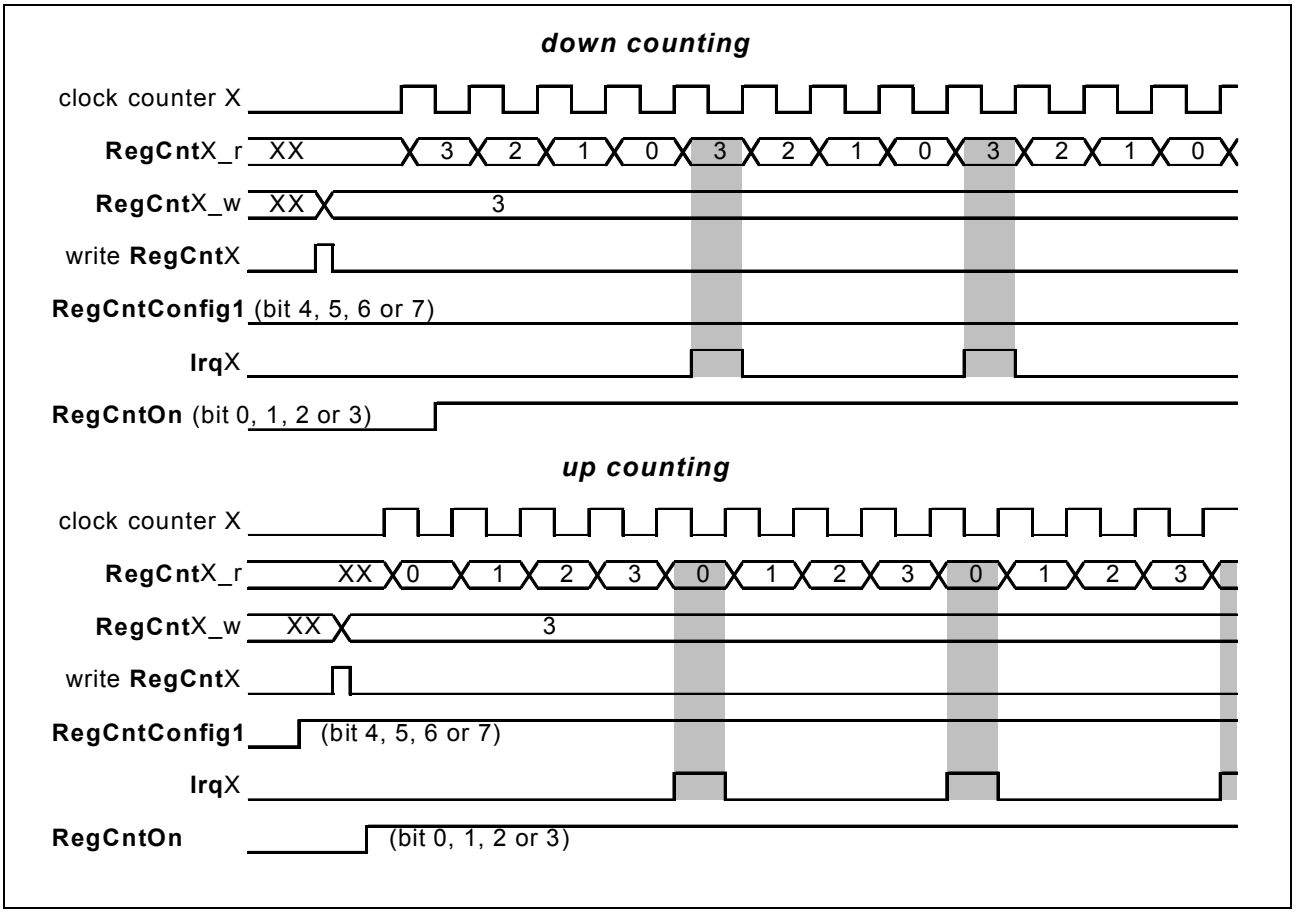


Figure 18 - Up and down count interrupt generation

3.10.7PWM Mode

The counters can generate PWM signals (Pulse Width Modulation) on port PB outputs PB[0] and PB[1]. The PWM mode is selected by setting **RegCntConfig1[0]** or **RegCntConfig1[1]** bit.
When **RegCntConfig1[0]** is set, the PWMA or PWMAB output value overrides the value set in **RegPBOut[0]**. When **RegCntConfig1[1]** is enabled, the PWMC or PWMCD output value overrides the value set in **RegPBOut[1]**. The corresponding ports (0 and/or 1) of PB must be configured as digital output.
Counters in PWM mode count down or up, according to the **RegCntConfig1[7:4]** bit setting. No interrupts and events are generated by the counters which are in this mode. Counters count circularly: they restart at zero or at the maximal value (0xFF when not cascaded or 0xFFFF when cascaded) when respectively an overflow or an underflow condition occurs.

The internal PWM signals are low as long as the counter contents are higher than the PWM code values written in the **RegCntX** registers. They are high when the counter contents are smaller or equal to these PWM code values. In order to have glitch free outputs, the PWM outputs on PB[0] and PB[1] are sampled versions of these internal PWM signals, therefore delayed by one counter clock cycle.
The PWM resolution is always 8 bits when a single counter is used for the PWM signal generation. **RegCntConfig2** register is used to set the PWM resolution for counters A and B or C and D respectively when they are in cascaded mode. The different possible resolutions in cascaded mode are shown in Table 72 - PWM resolution. Choosing a 16-bit PWM code which is higher than the maximum value results in a PWM output always tied to 1. The maximum value is $2^{resolution - 1}$.

RegCntConfig2[1:0] or RegCntConfig2[3:2]	PWM Resolution
11	16 bits
10	14 bits
01	12 bits
00	10 bits

Table 72 - PWM resolution

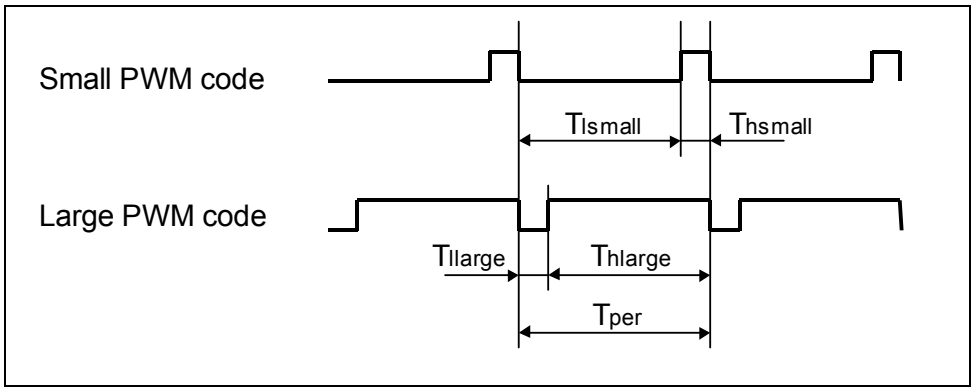


Figure 19 - PWM modulation examples

The period T_{per} of the PWM signal is given by the formula:

$$T_{per} = \frac{2^{resolution}}{f_{ckcnt}}$$

The duty cycle ratio DCR of the PWM signal is defined as:

$DCR = \frac{T_h}{T_{per}}$, where T_h is the time during which the output is "high" within T_{per}

DCR can be selected between $\frac{100}{2^{resolution}}$ % and 100 %.

DCR (in %) as a function of the **RegCntX** content(s) is given by the relation:

$$DCR = MIN\left(\frac{100 \cdot (1 + \mathrm{RegCntX})}{2^{resolution}}, 100\right)$$

3.10.8 Counter Capture Function

The 16-bit capture register is provided to facilitate frequency measurements. It provides a safe reading mechanism for the counters A and B when they are running. When the capture function is active, the processor does not read the counters A and B directly anymore, but instead reads shadow registers located in the capture block. An interrupt is generated after a capture condition has been met when the shadow register content is updated. The capture condition is user defined by selecting either internal capture signal sources derived from the prescaler or from the external PA[2] or PA[3] ports. Both counters use the same capture condition.

When the capture function is active, the A and B counters can either upcount or downcount. They do count circularly: they restart at zero or at the maximal value (either 0xFF when not cascaded or 0xFFFF when cascaded) when respectively an overflow or an underflow condition occurs in the counting. The capture function is also active on the counters when used to generate PWM signals.
The bits **RegCntConfig2[5:4]** determine if the capture function is enabled or not and selects which edges of the capture signal source are valid for the capture operation. The source of the capture signal can be selected by setting the **RegCntConfig2[7:6]** bits. For all sources; rising, falling or both, edge sensitivity can be selected.
The Table 73 shows the capture condition as a function of the setting of these configuration bits.

RegCntConfig2[7:6]	Selected capture signal	RegCntConfig2[5:4]	Selected condition	Capture condition
11	1 K	00 01 10 11	**Capture disabled** Rising edge Falling edge Both edges	- 1 K rising edge 1 K falling edge 2 K
10	16 K	00 01 10 11	**Capture disabled** Rising edge Falling edge Both edges	- 16 K rising edge 16 K falling edge 32 K
01	PA3	00 01 10 11	**Capture disabled** Rising edge Falling edge Both edges	- PA[3] rising edge PA[3] falling edge PA[3] both edges
00	PA2	00 01 10 11	**Capture disabled** Rising edge Falling edge Both edges	- PA[2] rising edge PA[2] falling edge PA[2] both edges

Table 73 – Capture conditions

The bits **RegCntConfig2**[7:6] and **RegCntConfig2**[5:4] should be modified only when the counters are stopped otherwise data may be corrupted during one counter clock cycle.
Due to the synchronization mechanism of the shadow registers and depending on the frequency ratio between the capture and counter clocks, the interrupts may be generated one or two counter clock pulses after the effective capture condition occurred. When the counters A and B are not cascaded and do not operate on the same clock, the counter A and counter B interruptions which inform that the capture condition was met, may appear at different instants. In this case, the processor should read the shadow register associated to a counter only if the interruption related to this counter has been detected. An edge is detected on the capture signals only if the minimal pulse widths of these signals in the low and high states are higher than a period of the counter clock source.

3.11 SERIAL PERIPHERAL INTERFACE (SPI)

3.11.1 Features

- Full duplex operating mode
- Master/slave configuration
- Separate transmit data, shift data, and receive data registers in order to perform back-to-back transmissions.
- Four programmable baud rates
- Programmable serial clock polarity and phase
- SPI receive register full interrupt
- Overflow detection flag
- 8 I/O pads which can be configured as SPI or GPIO.
- Support up to 5 slaves devices
- Content of a serial memory connected to NSS[0] automatically loaded upon reset

3.11.2 Register Map

Name	Address (Hex)
RegSpiControl	0x0068
RegSpiStatus	0x0069
RegSpiDataOut	0x006A
RegSpiDataIn	0x006B
RegSpiPullup	0x006C
RegSpiDir	0x006D
RegSpiSlvSel	0x006E

Table 74 - SPI registers

Pos	RegSpiControl	r/w	Reset	Function
7	ClearCounter	rw	0	1 = clear control counters
6	NotSlaveSelect	rw	1	In master mode: drives the NSS[0] pin. It must be set to 0 during byte transfer. In slave mode: unused
5	SpiMaster	rw	1	1 = master mode selected 0 = slave mode selected
4	SpiEnable	rw	1	1 = SPI mode 0 = GPIO mode
3	ClockPhase	rw	0	clock phase
2	ClockPolarity	rw	0	clock polarity
1:0	BaudRate	rw	00	In master mode: baud rate selection 00 = ckRCext/2 01 = ckRCext/8 10 = ckRCext/16 11 = ck4kHz In slave mode: unused

Table 75 - RegSpiControl register

Pos	RegSpiStatus	r/w	Reset	Function
7:3	-	r	00000	reserved
2	SpiOverflow	rc1	0	Overflow flag. Cleared when written to 1
1	SpiRxFull	r	0	1 = a byte has been received and is available in the receive register. This flag is cleared when reading RegSpiDataIn.
0	SpiTxEmpty	rw1	1	1 = the transmit register is empty and a new transfer can be initiated. The flag is cleared when writing RegSpiDataOut.

Table 76 - RegSpiStatus register

Pos	RegSpiDataOut	r/w	Reset	Function
7:0	SPIDataOut	rw	00000000	In SPI mode: byte to transmit in GPIO mode: value forced on pads if configured as digital outputs

Table 77 - RegSpiDataOut register

Pos	RegSpiDataIn	r/w	Reset	Function
7:0	SPIDataIn	r	00000000	In SPI mode: received byte in GPIO mode: value read on pads

Table 78 - RegSpiDataIn register

Pos	RegSpiPullup	r/w	Reset	Function
7:0	SPIPullup	rw	11111111	1 = pull-up enabled (for each corresponding pad) 0 = pull-up disabled (for each corresponding pad)

Table 79 - RegSpiPullup register

Pos	RegSpiDir	r/w	Reset	Function
7:4	SPIDir[7:4]	rw	0000	for each corresponding pad: 1 = pad configured as digital output 0 = pad configured as digital input
3:0	SPIDir[3:0]	rw	0000	In GPIO mode, for each corresponding pad: 1 = pad configured as digital output 0 = pad configured as digital input

Table 80 - RegSpiDir register

Pos	RegSpiSlvSel	r/w	Reset	Function
7:3	-	r	0000	Reserved
2	MultSlvSel	rw	0	1 = multiple slave mode selected 0 = SX1441 compatible mode
1:0	SlvSelect[1:0]	rw	00	0 = nss(1) activated 01 = nss(2) activated 10 = nss(3) activated 11 = nss(4) activated

Table 81 - RegSlvSel register

3.11.3Operation

The peripheral operates in two modes: SPI and GPIO. The mode is selected by the **SpiEnable** bit of register **RegSpiCtrl**. Selecting the SPI mode only affects the pins SCK, MOSI, MISO, and NSS[0]. The pins NSS[4:1] are always configured as GPIO.

In SPI mode, SCK is the serial clock. It is generated by the chip in master mode and its frequency is chosen by the bit field **BaudRate** of **RegSpiCtrl** as described in Table 75. In master mode, **SpiMaster** set to 1, the SX1441 is the master. In slave mode, **SpiMaster** set to 0, the master should always limit the frequency of SCK to ckRCext/2). The Figure 20 shows how to connect devices to the SPI interface. The slave connected to NSS[0] is the boot device. The program is read from it at reset. Others slaves may be connected as described in Figure 20.

The bit **ClearCounter** of **RegSpiCtrl** may be used if the synchronization with a slave is lost, to re-initialize the communication. The **SpiRxFull** flag of **RegSpiStatus** is used as an interrupt source in the interrupt controller block.

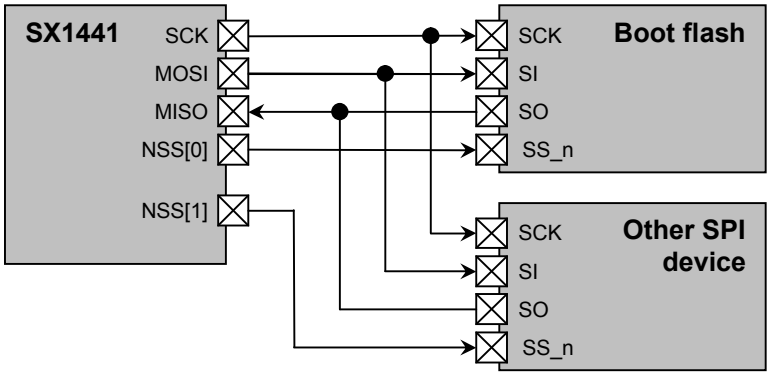


Figure 20 - Connecting SPI devices

The Figure 21 shows the timing diagrams for a SPI transmission with a clock phase equal to 0 (the active state of the serial clock SCK signal occurs on the 2nd half of the SCK cycle).

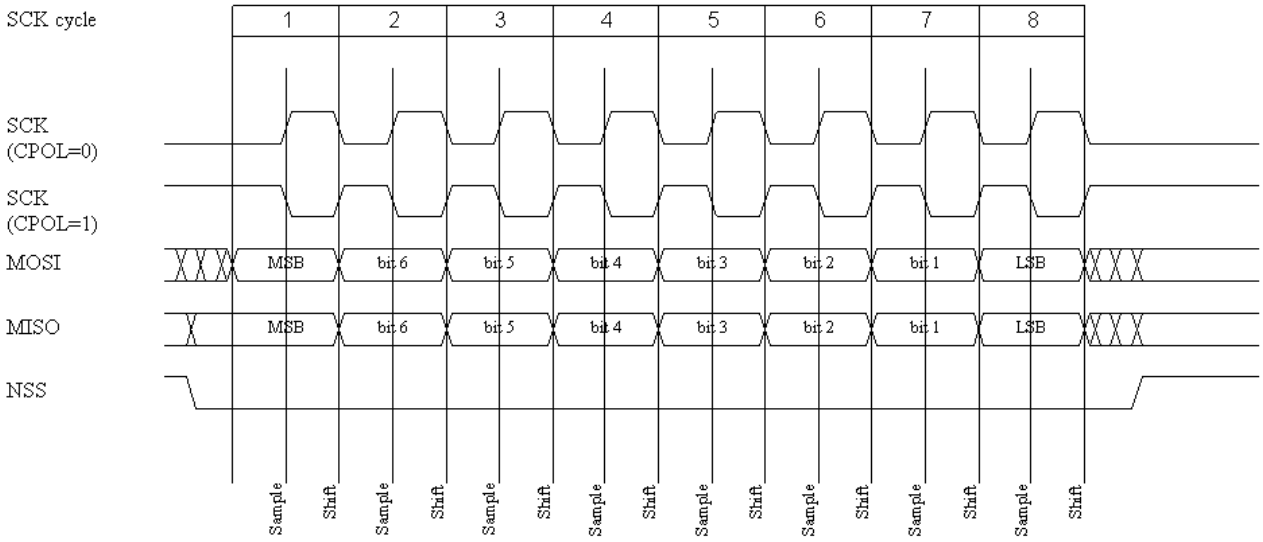


Figure 21 - SPI transmission format with a clock phase equal to 0

The Figure 22 shows the timing diagrams for a SPI transmission with a clock phase equal to 1 (the active state of the serial clock SCK signal occurs on the 1st half of the SCK cycle).

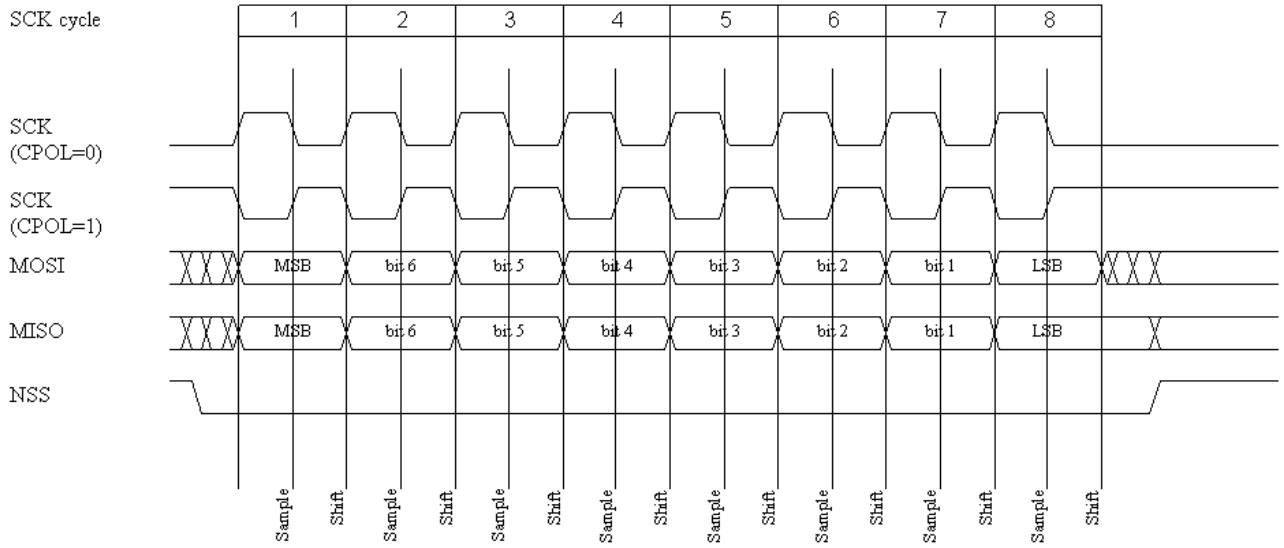


Figure 22 - SPI transmission format with a clock phase equal to 1

3.11.4Software Hints

3.11.4.1 Master Mode

Initialization

The SPI is configured in SPI master mode at reset. Clock phase, clock polarity, and baud rate can be changed by writing **RegSpiControl**. **SpiMaster** and **SpiEnable** are set in **RegSpiControl**.

Byte transfer

De-assert NSS[0] pin by clearing **NotSlaveSelect** in **RegSpiControl**
Check that the transmit buffer is empty (**SpiTxEmpty** set in **RegSpiStatus**)
Fetch the SPI with the data to be transmitted. The data is written in **RegSpiDataOut**
Trigger transmission by writing any value in **RegSpiStatus**
Wait until the receive buffer is full (**SpiRxFull** set in **RegSpiStatus**)
The received data is read from **RegSpiDataIn**
Assert NSS[0] pin by setting **NotSlaveSelect** in **RegSpiControl**

3.11.4.2 Slave Mode

Initialization

SpiEnable is set and **SpiMaster** is cleared in **RegSpiControl**. Clock phase, clock polarity, and baud rate can be changed by writing **RegSpiControl**.

Byte transfer

Wait until the receive buffer is full (**SpiRxFull** set in **RegSpiStatus**)
Read the received data from **RegSpiDataIn**
Fetch the SPI with the data to be transmitted. The data is written in **RegSpiDataOut**

3.11.5Pins

The Table 82 summarizes the SPI pins usage in either SPI or GPIO mode.

Pin name	Function		Bit number in registers
	SPI	GPIO	
SCK	serial clock	digital I/O with selectable pull-up	0
MISO	master-in, slave-out	digital I/O with selectable pull-up	1
MOSI	master-out, slave-in	digital I/O with selectable pull-up	2
NSS[0]	slave 0 select (active low)	digital I/O with selectable pull-up	3
NSS[1]	digital I/O with selectable pull-up	digital I/O with selectable pull-up	4
NSS[2]	digital I/O with selectable pull-up	digital I/O with selectable pull-up	5
NSS[3]	digital I/O with selectable pull-up	digital I/O with selectable pull-up	6
NSS[4]	digital I/O with selectable pull-up	digital I/O with selectable pull-up	7

Table 82 - SPI pins

3.12 APPLICATION UART

3.12.1Features

- Full duplex operation with buffered 8-byte FIFO for receiver and transmitter.
- Internal baud rate generator with 8 x 32 programmable baud rates.
- 7 or 8 bits word length.
- Even, odd, or no-parity bit generation and detection
- 1 stop bit
- Error receive detection: Start, Parity, Frame and Overrun
- Receiver echo mode
- Three interrupts (receive: data ready, FIFO threshold. Transmit: FIFO empty)
- Enable receive and/or transmit
- Invert pad Rx and/or Tx
- Flow control (RTS and CTS)

3.12.2Registers Map

Name	Address (Hex)
RegUartFifoCtrl	0x0030
RegUartFifoBaud	0x0031
RegUartFifoTx	0x0032
RegUartFifoTxSta	0x0033
RegUartFifoRx	0x0034
RegUartFifoRxSta	0x0035
RegUartFifoMisc	0x0036

Table 83 - UART registers

Pos	RegUartFifoCtrl	r/w	Reset	Function
7	UartEcho	rw	0	1 = echo mode selected (UA_RX and UA_TX internally connected) 0 = echo mode not selected
6	UartEnRx	rw	0	1 = receiver enabled 0 = receiver disabled
5	UartEnTx	rw	0	1 = transmitter enabled 0 = transmitter disabled
4	UartXRx	rw	0	1 = UA_RX inverted 0 = UA_RX not inverted
3	UartXTx	rw	0	1 = UA_TX inverted 0 = UA_TX not inverted
2	UartPM	rw	0	1 = odd parity check selected 0 = even parity check selected
1	UartPE	rw	0	1 = parity check enabled 0 = parity check disabled
0	UartWL	rw	0	1 = 8-bit word selected 0 = 7-bit work selected

Table 84 - RegUartFifoCtrl register

Pos	RegUartFifoBaud	r/w	Reset	Function
7:5	UartRcSel	rw	000	prescaler tap selection
4:0	UArtRcDiv	rw	00000	baud rate selection

Table 85 - RegUartFifoBaud register

Pos	RegUartFifoTx	r/w	Reset	Function
7:0	UartTx	rw	00000000	data to be sent

Table 86 - RegUartFifoTx

Pos	RegUartFifoTxSta	r/w	Reset	Function
7:5	-	r	000	reserved
4	CTS	r	0	value of CTS pin
3	UartTxFifoOerr	r	0	transmitters overrun error flag. Cleared by reading **RegUartFifoTxSta**
2	UartTxFifoFull	r	0	transmit FIFO full flag
1	UartTxBusy	r	0	transmitter is busy transmitting data
0	UartTxFifoEmpty	r	1	transmit FIFO empty flag. Cleared by writing to **RegUartFifoTx**. Set when transferring the last data word from the FIFO to the internal shift register.
7:0	UartTxClear	w	-	clear the transmitter block when written

Table 87 - RegUartFifoTxSta register

Pos	RegUartFifoRx	r/w	Reset	Function
7:0	UartRx	r	00000000	received data

Table 88 - RegUartFifoRx register

Pos	RegUartFifoRxSta	r/w	Reset	Function
7	-	r	0	reserved
6	UartRxFifoFull	r	0	receive FIFO full flag
5	UartRxSErr	r	0	start error flag
4	UartRxPErr	r	0	parity error flag
3	UartRxFErr	r	0	frame error flag
2	UartRxFifoOErr	r/c	0	overrun error flag. Cleared by reading **RegUartFifoRxSta**
1	UartRxBusy	r	0	uart receiver busy flag
0	UartRxDataReady	r	0	1 = data available in the receive FIFO. Cleared by reading all the data in the FIFO.
7:0	UartRxClear	w	-	clear the receiver block when written

Table 89 - RegUartFifoRxSta

Pos	RegUartFifoMisc	r/w	Reset	Function
7:6	-	r	00	reserved
5	RtsMonitorMode	rw	0	1 = set RTS to 1 during monitor mode 0 = do not force RTS during monitor mode
4	RtsLevelMode	rw	0	1 = RTS rises when only 2 bytes left in the Rx FIFO and falls when the FIFO is empty 0 = RTS rises when only 2 bytes left in the FIFO and falls when more than 2 bytes left in the FIFO.
3	Sel32k	rw	0	1 = input clock is ck32kHz low prescaler output 0 = input clock is ckRCext,.
2	UartFlowCtrl	rw	0	1 = enable flow control 0 = disable flow control
1:0	-	r	00	reserved

Table 90 - RegUartFifoMisc register

3.12.3Block Diagram

The UART is a Universal Asynchronous Receiver Transmitter interface with separated receive and transmit 8-byte FIFO, and fully automatic flow control.

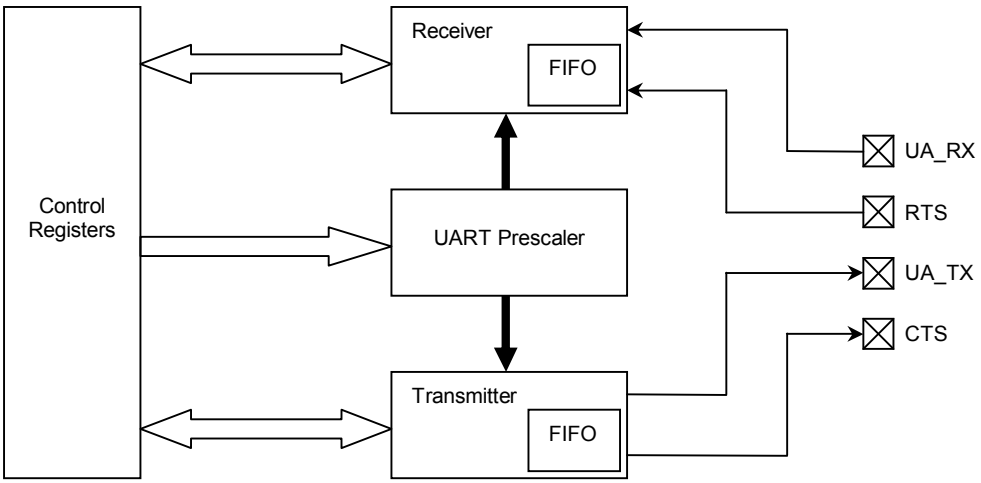


Figure 23 - UART block diagram

3.12.4Configuration

The configuration bits of the UART can be found in the registers **RegUartFifoBaud**, **RegUartFifoCtrl** and **RegUartFifoMisc**.
The bits **UartEnRx** and **UartEnTx** are used to enable or disable the reception and transmission. The word length (7 or 8 data bits) can be chosen with **UartWL**. A parity bit is added during transmission or checked during reception if **UartPE** is set. The parity mode (odd or even) can be chosen with **UartPM**. Setting the bits **UartXRx** and **UartXTx** respectively inverts the UA_RX and UA_TX signals.
The bit **UartEcho** automatically sends back the received data. The transmission function becomes then UA_TX = UA_RX XOR **UartXRx** XOR **UartXTx**. **UartEnRx** or **UartEnTx** must be set to 1 to use the echo mode.
The bits **UartFlowCtrl**, **RtsLevelMode** and **RtsMonitorMode** are used to control the flow through RTS and CTS pins as described in paragraph 3.12.8.

3.12.5Baud Rates

The UART interface can be clocked by ckRCext when the **Sel32k** bit is cleared. **UartRcSel** and **UartRcDiv** select the baud rate. The relation between the baudrate and the ckRCext clock frequency is given by:

$$Baudrate = \frac{f_{ckRCext}}{16 \cdot factorUartRcSel \cdot factorUartRcDiv}$$

Equation 3 – Baudrate vs. ckRCext

FactorUartRcSel is the prescaler tap selection set by the bits **UartRcSel** in **RegUartFifoBaud** and factorUartRcDiv is the division set by the bits **UartRcDiv** in **RegUartFifoBaud**. The values of these factors are given in the Table 91 and Table 92. With a 14 MHz frequency clock, the highest baudrate is 875 kbits/s.
Due to the RC clock dispersion, a digital frequency lock loop (DFLL) must be used to calibrate it before using it as a clock source.

UartRcSel	factorUartRcSel
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Table 91 - Division factor for UartRcSel

UartRcDiv	factorUartRcDiv
00000	1
00001	2
00010	3
00011	4
00100	5
00101	6
00110	7
00111	8
01000	9
01001	10
01010	11
01011	12
01100	13
01101	14
01110	15

UartRcDiv	factorUartRcDiv
01111	16
10000	17
10001	18
10010	19
10011	20
10100	21
10101	22
10110	23
10111	24
11000	25
11001	26
11010	27
11011	28
11100	29
11101	30
11110	31
11111	32

Table 92 - Division factor for UartRcDiv

The ck32kHz low prescaler output can be selected as UART clock source if the bit **Sel32k** in **RegUartFifoCtrl** is set. The baud rate selection follows Equation 4.

$$Baudrate = \frac{f_{ck32kHz}}{14 \cdot factorUartRcDiv}$$

Equation 4 – Baudrate vs. ck32kHz

Ck32kHz can be generated from SLW_CLOCK_IN or the high prescaler input as described in 3.5.8.2. Table 93 shows the baud rate and the precision when SLW_CLOCK_IN is selected.

UartRcDiv	Baud rate (Bd/s)	SLW_CLOCK_IN frequency	
		32'000 Hz	32'768 Hz
0000	2400	- 4.8 %	- 2.5 %
0001	1200	- 4.8 %	- 2.5 %
0011	600	- 4.8 %	- 2.5 %
0111	300	- 5 %	- 2.5 %

Table 93 - Baud rate selection (Sel32k = 1, EnableSwClock = 1)

3.12.6Transmission

The transmitter has to be enabled by setting **UartEnTx**. Data to be sent have to be written to the transmit FIFO through the register **RegUartFifoTx**. The transmitter loads and sends data automatically, as long as the transmission FIFO is not empty (bit **UartTxFifoEmpty** = 0) and the value on pad CTS is 0. When the transmit FIFO is empty, the **UartTxFifoEmpty** bit returns to 1 and an interrupt is generated.
The bit **UartTxFifoFull** in **RegUartFifoTxSta** indicates that the transmit FIFO is full. If new data are written in the FIFO while it is full, the bit **UartTxFifoOerr** is set to 1 and the last data are ignored. The bit **UartTxFifoOerr** is cleared when the register **RegUartFifoTxSta** is read.
The bit **UartTxBusy** in **RegUartFifoTxSta** shows that the transmitter is busy transmitting a word.
Writing in RegUartFifoTxSta resets the transmitter block. Data in the FIFO that were not transmitted are lost. The flags in RegUartFifoTxSta are reset.

Figure 24 and
Figure 25 show an UART transmission. The figures are drawn with a FIFO depth of 2, for simplification, although the FIFO depth is 8.

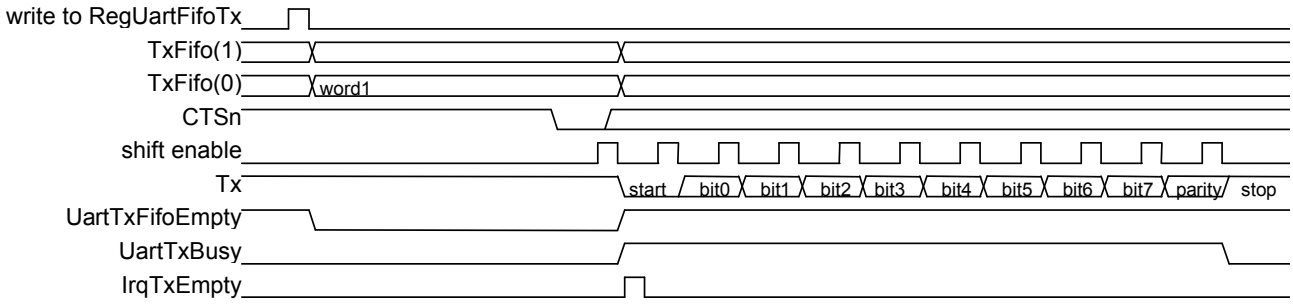


Figure 24 - Uart transmission timing diagram with FIFO depth = 2.

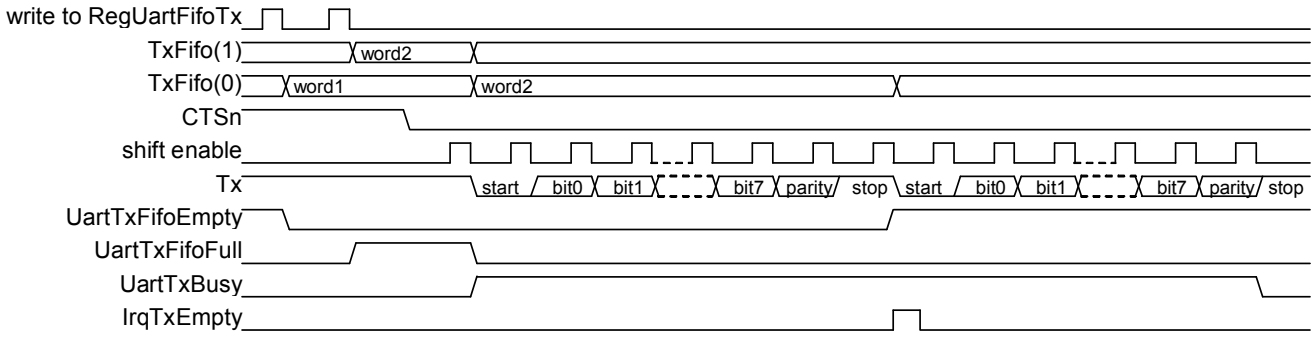


Figure 25 - Uart transmission timing diagram back to back with FIFO depth = 2

3.12.7 Reception

On detection of the start bit, the UartRxBusy bit is set. On detection of the stop bit, the received data and flags are transferred from the internal shift register to the receive FIFO. At the same time, the bit UartRxDataReady and the interrupts RxDataReady or RxComp are updated. The bit UartRxDataReady is set as long as the data present in the FIFO are not read by the software. The interrupt RxDataReady is generated each time new data are written to the FIFO. The interrupt RxComp is generated when only two free data words are left in the reception FIFO.
The flags in the register RegUartFifoRxSta give the status of the next word to be read in the reception FIFO. Therefore, in order to know the status of the received data, RegUartFifoRxSta has to be read before reading the actual data in RegUartFifoRx. Each data word in the reception FIFO has three flags associated to it: UartRxSErr, UartRxPErr and UartRxFErr.
The bit UartRxSErr is set if a start error has been detected. The bit UartRxPErr is set if a parity error has been detected, i.e. the received parity bit is not equal to the calculated parity of the received data. The bit UartRxFErr shows that a frame error has been detected: no stop bit has been detected.
The UartRxFifoFull bit is set when the receive FIFO is full. If the FIFO is full and new data are transferred from the shift register to FIFO, the bit UartRxFifoOErr (overflow error) is set and the new data are lost. Reading RegUartFifoRxSta clears UartRxFifoOErr.
Writing any data to RegUartFifoRxSta resets the reception block: all flags in RegUartFifoRxSta are reset and data in the reception FIFO that were not yet read by the software are lost.
RTS is used for the flow control. While the reception FIFO reached the threshold level, RTS is set. RTS is cleared as soon as the software reads data in the reception FIFO depending on RtsLevelMode.
Error! Reference source not found. shows the timing diagram for a possible reception. In this example, the depth of the FIFO is 4. RTS1 shows the functionality when RtsLevelMode = 0 and RTS2 when RtsLevelMode = 1. The actual depth of the FIFO is 8.

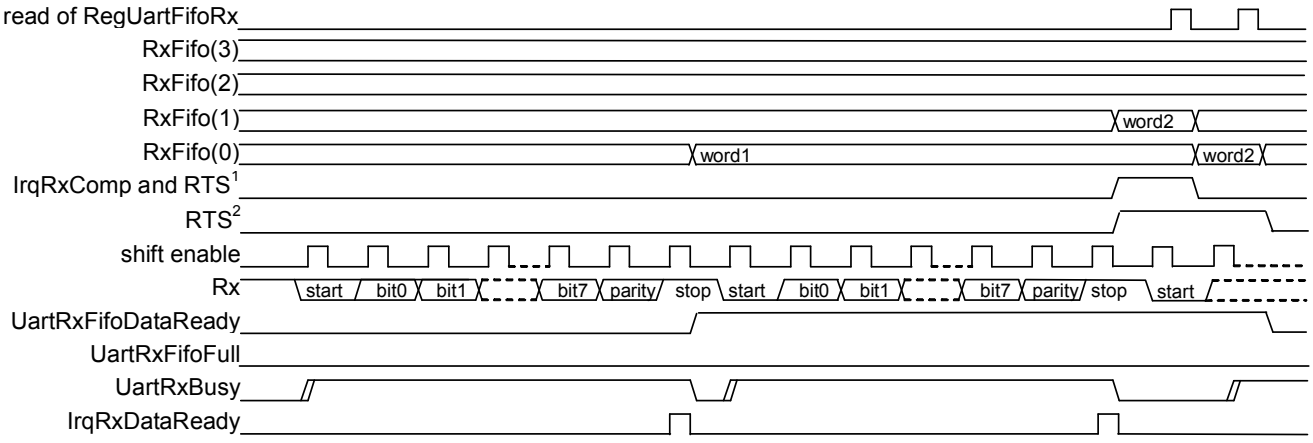


Figure 26 – Uart reception timing diagram with FIFO depth = 4

3.12.8 Flow Control

When automatic flow control is activated (**UartFlowCtrl** = 1), the transmission stops as soon as signal CTS is raised. It transmits otherwise as long as data are available for transmission.
On the receiver side, the RTS signal will automatically be driven high as the antepenultimate byte of the receive FIFO is filled. It is driven low again as the receive FIFO is emptied (**RtsLevelMode** = 1) or as only two bytes are left in the FIFO (**RtsLevelMode** = 0).
By connecting two devices as shown in Figure 26, transmission overruns are avoided as a device will automatically stop transmitting as the other one - receive FIFO - gets full.

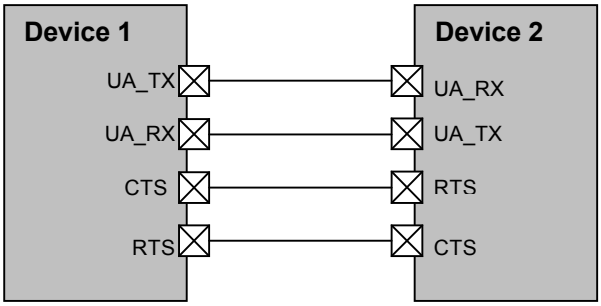


Figure 26 - Connecting devices with flow control

If flow control is disabled, pins CTS and RTS can be used as digital input/output ports.
Setting bit **RtsMonitorMode** to 1 will automatically raise the RTS signal as the chip enters the debug mode. This bit should be set by default to prevent loss of data.

3.12.9 Software Hints

The transmission and reception software can be driven by interruption or by polling the status bits.

3.12.9.1 Transmission with Polling

Initialize **RegUartFifoBaud** and **RegUartFifoCtrl** with the communication parameters (for example 8-bit word length, odd parity, 115200 bauds, enable UART transmission)
Write 8 bytes into the transmit FIFO (**RegUartFifoTx**)
Wait until **UartTxFifoEmpty** in **RegUartFifoTxSta** is set
Jump to 2) to write the next 8 bytes if the message is not finished
End of transmission

3.12.9.2 Transmission with Interrupt

Initialize **RegUartFifoBaud** and **RegUartFifoCtrl** with the communication parameters (for example 8-bit word length, odd parity, 115200 bauds, enable UART transmission)

Write 8 bytes into the transmit FIFO (**RegUartFifoTx**)
Jump to 2) after IrqTxEmpty has been triggered, to write the next 8 bytes if the message is not finished
End of transmission

3.12.9.3 Reception with Polling
Initialize **RegUartFifoBaud** and **RegUartFifoCtrl** with the communication parameters (for example 8-bit word length, odd parity, 115200 bauds, enable UART reception)
Wait until **UartRxDataReady** or **UartRxFifoFull** are set in **RegUartFifoRxSta**
Check errors in **RegUartFifoRxSta**
Read data in **RegUartFifoRx**
Repeat 3) and 4) until the receive FIFO is empty (**UartRxDataReady** = 0)
Jump to 2)

3.12.9.4 Reception with Interrupt
Initialize **RegUartFifoBaud** and **RegUartFifoCtrl** with the communication parameters (for example 8-bit word length, odd parity, 115200 bauds, enable UART reception)
Wait until IrqRxDataReady is triggered
Check errors in **RegUartFifoRxSta**
Read data in **RegUartFifoRx**
Repeat 3) and 4) until the receive FIFO is empty (**UartRxDataReady** = 0)
Jump to 2)

3.13 BLUETOOTH SEQUENCER INTERFACE

3.13.1 Features

- Fully embedded qualified ROM-based implementation of the lower layers of the Bluetooth protocol stack
- Compliant with Revision 1.2 of the Bluetooth specification
- Enhanced SCO (eSCO) mode
- Adaptive Frequency Hopping (AFH) support
- Fast Connect feature
- Embedded CVSD audio compression
- Direct interface with the audio codec
- Direct interface with the radio chip
- Up to 3 slaves and one audio-link
- Supports point-to-point, piconet, and scatternet networks

3.13.2 Registers Map

Name	Address (Hex)
RegBtmCtrl1	0x007C
RegBtmCtrl2	0x007D

Table 94 – Bluetooth Sequencer registers

Pos	RegBtmCtrl1	r/w	Reset	Function
7-6	-	r	00	unused
6	BtmWlanBusy	rw	0	1 = colocated WLAN is currently receiving 0 = no WLAN activity currently active
5	BtmReset	rw	0	1 = Reset Bluetooth Sequencer 0 = Enable Bluetooth Sequencer (see also bit 4)
4	BtmEnable	rw	0	1 = Bluetooth Sequencer is enabled 0 = Bluetooth Sequencer is in power down mode
3	BtmSpiEnBar	rw	0	1 = Force ‘1’ on SPI_EN_BAR of the Bluetooth UART (reserved for test) 0 = Default value
2	BtmBusy	r	0	BT radio transmitter or receiver is active when set.

Pos	RegBtmCtrl1	r/w	Reset	Function
1	Reserved	rw	0	unused
0	BtmSync	R	0	BT Sequencer is locked on slot timing when set.

Table 95 – RegBtmCtrl1 register

Pos	RegBtmCtrl1	r/w	Reset	Function
7-6	-	r	00	unused
5	BtmWakeUp	rw	0	1 = Request BT Sequencer to resume from deep sleep state 0 = default value
4	BtmSleepRst	rw	0	1 = Request BT Sequencer to resume from HALT state 0 = Enable Bluetooth Sequencer (see also bit 4)
3	BtmClkStat	r	1	1 = Internal BT sequencer clock is derived from CLK_IN 0 = Internal BT sequencer clock is internally generated
2	BtmSleepStat	r	0	1 = BT Sequencer in HALT mode 0 = BT Sequencer is active.
1	BtmBusy	r	0	BT radio transmitter or receiver is active when set.
0	BtmOscEn	r	1	1 = BT Sequencer oscillator is active 0 = BT Sequencer oscillator is shut down.

Table 96 – RegBtmCtrl2 register

3.13.3 Overview

The Bluetooth sequencer implements the Bluetooth specific hardware and lower layers of the protocol stack. The lower layers handle time-critical and hardware-dependant tasks that must not be disturbed by the application. As a qualified dedicated ROM-based coprocessor, the Bluetooth sequencer isolates the lower layers from the application. As a consequence, debug and qualification time is dramatically decreased.

The Host Controller Interface (HCI) has been specified into the Bluetooth protocol as a standardized interface between the lower and the upper layers. The upper layers are pieces of software implemented on the host processor and communicating with the Bluetooth sequencer though the HCI. The HCI commands are carried by an internal UART link between the host processor and the Bluetooth sequencer.

All Bluetooth specific commands are usually embedded into abstraction layers by the upper layers of the Bluetooth stack, so that the programmer only deals with common services (e.g. serial port emulation, etc …). Full Bluetooth stacks from various vendors have been successfully ported to the SX1441.

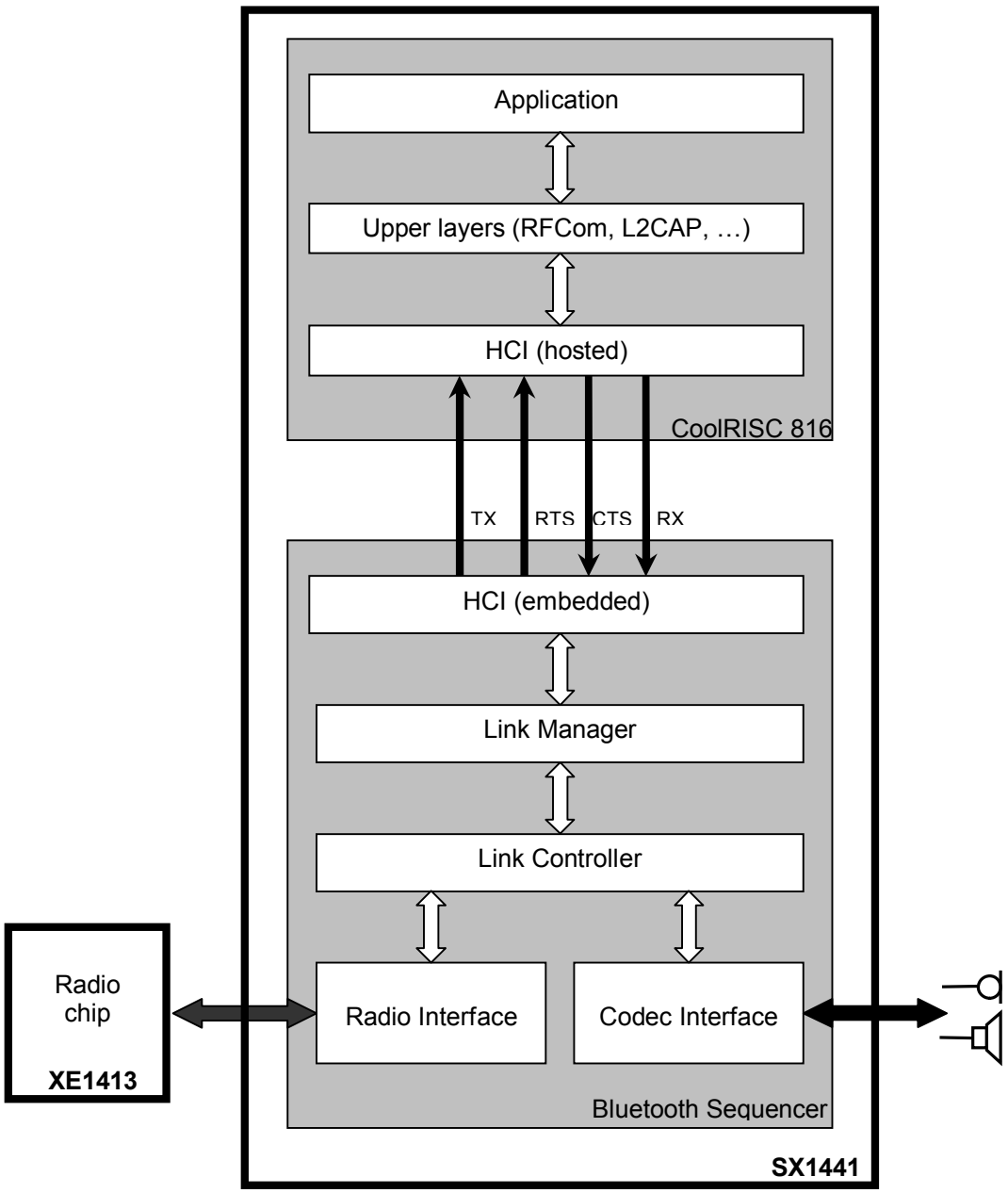


Figure 27 - SX1441 Bluetooth stack implementation

3.13.4Link Controller Features

The Link Controller supports the features as described in Table 97.

Link Controller Feature	Supported
SCO links	Yes
eSCO links	Yes
ACL links	Yes
Packet formatting	Yes
Control packets (ID, NULL, POLL, FHS)	Yes
Voice packets (HV1, HV2, HV3)	Yes
eSCO packets (EV3, EV4, EV5)	Yes
Mixed voice-data packets (DV)	Yes

Link Controller Feature	Supported
Single-slot data packets (DM1, DH1, AUX1)	Yes
Multi-slots data packets (DM3, DH3, DM5, DH5)	Yes
Page and page scan	Yes
Inquiry and inquiry scan	Yes
Broadcasting of messages	Yes
Sniff mode	Yes
Hold mode	Yes
Park mode	Yes
Single piconet point-to-point operation (master or slave)	Yes
Single piconet operation (master with multiple slaves)	Yes
Master-Slave switch	Yes
Scatternet operation (master of a piconet and slave of another)	Yes
Scatternet operation (slave of two piconets)	Yes
Adaptive Frequency Hopping	Yes
CVSD hardware compression	Yes
PCM support (linear) from codec (internal or external)	Yes
PCM support (A- or µ-law) from codec (internal or external)	Yes
Voice channel (1 channel)	Yes
Voice channel (2 or 3 channels)	No
Bluetooth test mode (standard)	Yes
Bluetooth test mode (reduced hopping sequence)	Yes

Table 97 – Link controller features

3.13.5Link Manager Features

The Table 98 shows the supported Link Manager (LM) features.

Link Manager Feature	Supported
Encryption	Yes
Encryption key size	Yes
Clock offset request	Yes
Slot offset information	Yes
Master-slave switch	Yes
Hold mode	Yes
Sniff mode	Yes
Park mode	Yes
Power control	Yes

Table 98 – Link manager features

3.13.6Standard Host Controller Interface (HCI) Commands

The table contains all messages understood by the Link Manager of the SX1441. A detailed description of the command and of its parameters can be found in the HCI specification of the Bluetooth Specification [1].

3.13.7Vendor Specific HCI Commands – "EasyBlue™ Commands"

In addition to standard HCI commands, EasyBlue commands can be used to access SX1441 specific features. Those commands allow the reading/writing of internal sequencer registers and setting the BdAddress.

3.13.7.1 Registers Writing

Synopsis

Command	OGF	OCF	Parameters	Return parameters
EasyBlue_WriteReg	0x3F	0x02	Address, Length, Data	Status

Table 99 - EasyBlue_WriteReg synopsis

Parameters

Parameter	Size	Comment
Address	4 bytes	address of the register (MSB first)
0x08	1 byte	fixed
Length	1 byte	Number of bytes to transfer
Data	[Length] bytes	data
Status	1 byte	0x00 : command succeed

Table 100 - EasyBlue_WriteReg parameters

Description

This command is used to write a value in the Bluetooth sequencer registers.

3.13.7.2 Registers Reading

Synopsis

Command	OGF	OCF	Parameters	Return parameters
EasyBlue_ReadReg	0x3F	0x01	Address, Type, Length1	Status, Length2, Data

Table 101 - EasyBlue_ReadReg synopsis

Parameters

Parameter	Size	Comment
Address	4 bytes	address of the register (MSB first)
0x08	1 byte	fixed
Length1	4 byte	Number of bytes to transfer.
Data	[Length] bytes	data
Status	1 byte	0x00 : command succeed
Length2	1 byte	number of bytes returned

Table 102 - EasyBlue_ReadReg parameters

Description

This command is used to read a value from the Bluetooth sequencer registers.

3.13.7.3 Setting the Bluetooth Address

Synopsis

Command	OGF	OCF	Parameters	Return parameters
EasyBlue_SetBdAddr	0x3F	0x03	Reserved1, BdAddr, Reserved2	Status

Table 103 - EasyBlue_SetBdAddr synopsis

Parameters

Parameter	Size	Comment
Reserved1	1 byte	0x00
BdAddr	6 bytes	BdAddr
Reserved2	11 bytes	0x00000000000000000000
Status	1 byte	0x00: command succeeded

Table 104 - EasyBlue_SetBdAddr parameters

Description
This command sets the 48-bit unique identifier for the Bluetooth device.

3.13.8 Radio Interface

Table 105 shows how to connect the XE1413 radio chip to the SX1441.

SX1441 Pin name	Location	XE1413 Pin name
SYS_CLOCK_IN	J9	SYS_CLK_OUT
RX_DATA	K4	RX_DATA
SPI_DATA_IN	J4	SPI_DATA_OUT
SLW_CLOCK_IN	K3	SLW_CLK_OUT
TX_DATA	K9	TX_DATA
RX_EN	K8	RX_EN
SYNC_DETECT	K7	SYNC_DETECT
TX_EN	K6	TX_EN
SPI_DATA_OUT	J8	SPI_DATA_IN
SPI_CLK_OUT	J7	SPI_CLK_IN
SPI_EN_BAR	K5	SPI_EN_BAR

Table 105 - Connecting the SX1441 and the XE1413

3.13.9 HCI UART

The host processor communicates with the Bluetooth Sequencer through an UART identical to the one described in paragraph 3.12. The register map for this peripheral is given in Table 106.

Name	Address (Hex)
RegHUartFifoCtrl	0x0050
RegHUartFifoBaud	0x0051
RegHUartFifoTx	0x0052
RegHUartFifoTxSta	0x0053
RegHUartFifoRx	0x0054
RegHUartFifoRxSta	0x0055
RegHUartFifoMisc	0x0056

Table 106 - HCI UART registers

Flow control is enabled at all time for the HCI UART. Upon start-up / reset, the HCI UART is configured for 115'200 kbits/s, 8 bits, no parity.

3.13.10 Bluetooth Sequencer Clock Source

As shown in paragraph 3.5.9, the Bluetooth sequencer is clocked by SYS_CLOCK_OUT and SLW_CLOCK_OUT. Both must fulfill the Bluetooth specifications. They are usually generated by the radio chip.

3.14 AUDIO CODEC

3.14.1 Features

- On-chip 16-bit audio linear codec, 8 kHz sampling rate, fully compliant with Bluetooth revision 1.2 specifications
- Internal voltage references to reduce external components count
- Single-ended or differential microphone input
- Class D DAC output stage with simple passive filter
- Integrated ADC preamplifier with configurable gain to adapt for various microphones
- Connected directly to the Bluetooth Sequencer
- Audio samples available to the host processor through direct memory access (DMA)

3.14.2 Register Map

Please note that the register map is split in 2 address ranges. The range 0x00E0-0x00FF maintains compatibility with SX1441, while the register in the range 0x0064-0x0067 is associated new functionality introduced into the SX1441 device (speaker and microphone volume control).

Name	Address (Hex)
RegVolCtrl	0x0064
RegVolCmdADC	0x0065
RegVolCmdDAC	0x0066
* reserved	0x0067
RegCodecCtrl	0x00E0
* reserved	0x00E1
RegDACSampleH	0x00E2
RegDACSampleL	0x00E3
RegADCSampleH	0x00E4
RegADCSampleL	0x00E5
RegDmaRdStartAddrH	0x00E6
RegDmaRdStartAddrL	0x00E7
RegDmaRdStopAddrH	0x00E8
RegDmaRdStopAddrL	0x00E9
RegDmaWrStartAddrH	0x00EA
RegDmaWrStartAddrL	0x00EB
RegDmaWrStopAddrH	0x00EC
RegDmaWrStopAddrL	0x00ED
RegDmaCtrl	0x00EE
RegCodecDataFlow	0x00EF
* reserved	0x00F0
RegADCGain	0x00F1
* reserved	0x00F2-0x00F8
RegCodecPaMute	0x00F9
* reserved	0x00FF

Table 107 - Codec registers

Pos	RegVolCtrl	r/w	Reset	Function
7-3	-	r	00000	unused
2	VolCtrlDACMute	rw	0	1 = Mute DAC output 0 = default DAC gain
1	VolCtrlADCMute	rw	0	1 = Mute ADC output 0 = default ADC gain
0	VolCtrlEn	rw	0	1 = enable volume control 0 = bypass volume control circuitry

Table 108 - RegVolCtrl register

Pos	RegVolCmdADC	r/w	Reset	Function
7-0	VolADCscaling	rw	00000000	Signed ADC scaling factor. See Description below

Table 109 - RegVolCmdADC register

Pos	RegVolCmdDAC	r/w	Reset	Function
7-0	VolDACscaling	rw	00000000	Signed DAC scaling factor. See Description below

Table 110 - RegVolCmdDAC register

Pos	RegCodecCtrl	r/w	Reset	Function
7	DmaIrqEnable	rw	0	1 = generate an interrupt when DMA stop address reached 0 = no interrupt generated
6	SampleIrqEnable	rw	0	1 = generate an interrupt when a ADC audio sample is available 0 = no interrupt generated
5	reserved	rw	0	Reserved
4	PcmEnable	rw	0	1 = enable PCM interface 0 = disable PCM interface
3	SerialLoopback	rw	0	1 = select ADC to DAC loopback through the PCM interface
2	ParallelLoopback	rw	0	1 = select direct ADC to DAC loopback
1	DacEnable	rw	0	1 = enable DAC 0 = disable DAC
0	AdcEnable	rw	0	1 = enable ADC 0 = disable ADC

Table 111 - RegCodecCtrl register

Pos	RegDACSampleH	r/w	Reset	Function
7:0	DACSample[15:8]	rw	00000000	MSB of audio sample sent to DAC

Table 112 - RegDACSampleHl register

Pos	RegDACSampleL	r/w	Reset	Function
7:0	DACSample[7:0]	rw	00000000	LSB of audio sample sent to DAC

Table 113 - RegDACSampleL register

Pos	RegADCSampleH	r/w	Reset	Function
7:0	ADCSample[15:8]	rw	00000000	MSB of audio sample read from ADC

Table 114 - RegADCSampleH register

Pos	RegADCSampleL	r/w	Reset	Function
7:0	ADCSample[7:0]	rw	00000000	LSB of audio sample read from ADC

Table 115 - RegADCSampleL register

Pos	RegDmaRdStartAddrH	r/w	Reset	Function
7:0	DmaReadStartAddr[15:8]	rw	00000000	MSB of the start address of the DMA read channel buffer

Table 116 - RegDmaRdStartAddrH register

Pos	RegDmaRdStartAddrL	r/w	Reset	Function
7:0	DmaReadStartAddr[7:0]	rw	00000000	LSB of start address of the DMA read channel buffer

Table 117 - RegDmaRdStartAddrL register

Pos	RegDmaRdStopAddrH	r/w	Reset	Function
7:0	DmaReadStopAddr[15:8]	rw	00000000	MSB of the end address of the DMA read channel buffer

Table 118 - RegDmaRdStopAddrH register

Pos	RegDmaRdStopAddrL	r/w	Reset	Function
7:0	DmaReadStopAddr[7:0]	rw	00000000	LSB of the end address of the DMA read channel buffer

Table 119 - RegDmaRdStopAddrL register

Pos	RegDmaWrStartAddrH	r/w	Reset	Function
7:0	DmaWriteStartAddr[15:8]	rw	00000000	MSB of the start address of the DMA write channel buffer

Table 120 - RegDmaWrStartAddrH register

Pos	RegDmaWrStartAddrL	r/w	Reset	Function
7:0	DmaWriteStartAddr[7:0]	rw	00000000	LSB of the start address of the DMA write channel buffer

Table 121 - RegDmaWrStartAddrL register

Pos	RegDmaWrStopAddrH	r/w	Reset	Function
7:0	DmaWriteStopAddr[15:8]	rw	00000000	MSB of the end address of the DMA write channel buffer

Table 122 - RegDmaWrStopAddrH register

Pos	RegDmaWrStopAddrL	r/w	Reset	Function
7:0	DmaWriteStopAddr[7:0]	rw	00000000	LSB of the end address of the DMA write channel buffer

Table 123 - RegDmaWrStopAddrL register

Pos	RegDmaCtrl	r/w	Reset	Function
7:6	-	r	00	Reserved
5	DmaReadCntLoad	rw	0	1 = load DmaReadStartAddr[15:0] in the DMA read channel address counter
4	DmaWriteCntLoad	rw	0	1 = load DmaWriteStartAddr[15:0] in the DMA write

Pos	RegDmaCtrl	r/w	Reset	Function
				channel address counter
3	DmaReadFull	r	0	set to 1 when the DMA read channel address counter reaches DmaReadStopAddr[15:0]
2	DmaWriteFull	r	0	set to 1 when the DMA write channel address counter reaches DmaWriteStopAddr[15:0]
1	DmaReadEnable	rw	0	1 = enable DMA read channel 0 = disable DMA read channel
0	DmaWriteEnable	rw	0	1 = enable DMA write channel 0 = disable DMA write channel

Table 124 - RegDmaCtrl register

Pos	RegCodecDataFlow	r/w	Reset	Function
7:4	-	r	0000	Reserved
3	DmaWriteSource	rw	0	1 = select PCM interface as DMA write channel source 0 = select ADC as DMA write channel source
2	PcmSource	rw	0	1 = select DMA read channel as PCM interface source 0 = select ADC as PCM interface source
1	DacSource	rw	0	1 = select DMA read channel as DAC source 0 = select PCM interface as DAC source
0	-	r	0	Reserved

Table 125 - RegCodecDataFlow register

Pos	RegAdcGain	r/w	Reset	Function
7:3	-	r	00000	Reserved
2	PreampDisable	rw	0	1 = disable ADC preamplifier 0 = enable ADC preamplifier
1:0	PreampGain	rw	00	Select ADC preamplifier gain 11 = gain x20 10 = gain x10 01 = gain x5 00 = preamplifier bypassed

Table 126 - RegADCGain register

Pos	RegCodecPaMute	r/w	Reset	Function
7:2	-	r	000000	Reserved
1	PaMuteP	rw	0	1 = force pin PA_OUTP to ground
0	PaMuteN	rw	0	1 = force pin PA_OUTN to ground

Table 127 - RegCodecPaMute register

3.14.3 Block Diagram

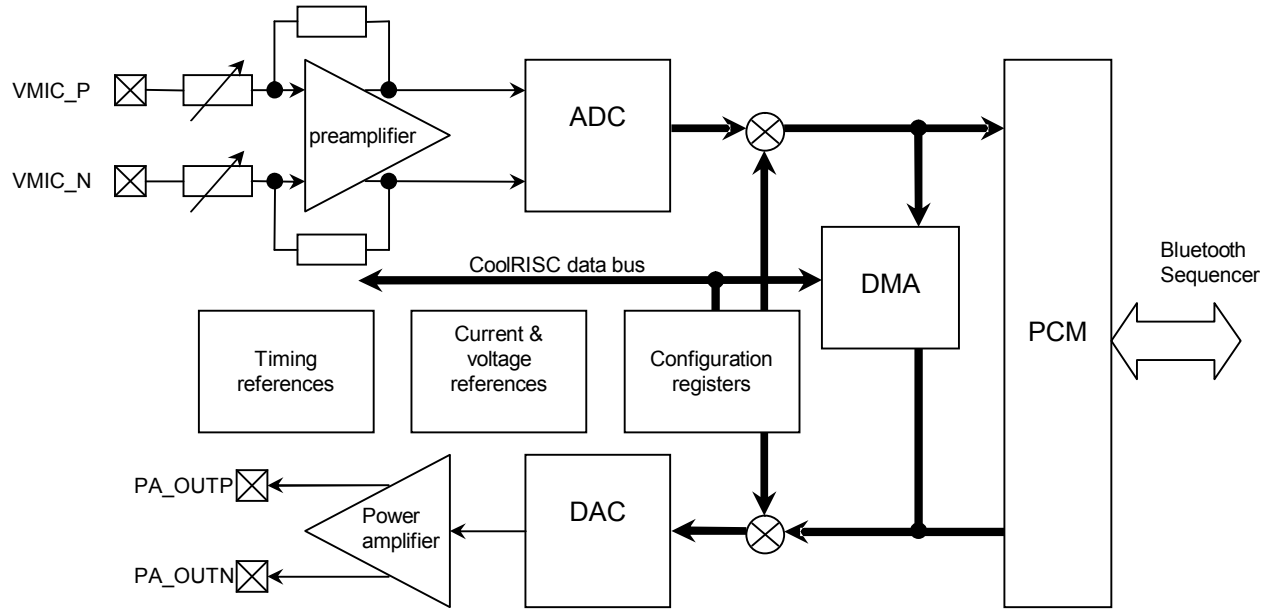


Figure 28 - Codec block diagram

The microphone input is fully differential. The signal from the microphone between inputs VMIC_P and VMIC_N is first amplified by a variable gain amplifier. The amplified signal is then sampled and digitized by the Σ-Δ ADC. The single bit output stream from the ADC is then converted into 16-bit, 8 kHz, linear PCM data representing the audio samples. These samples are sent to the PCM interface (connected to the Bluetooth Sequencer) or to the host processor memory through the DMA interface depending on the settings of the **RegCodecDataFlow** configuration register. The samples out of the ADC can also be directly read from the **RegADCSampleH/-L** registers.
The 16-bit digital data from either the Bluetooth Sequencer or the host processor memory through the DMA are converted into a PWM bit stream by the DAC. The host processor can also write samples directly into the **RegDACSampleH/-L** registers. This PWM bit stream is then amplified by the output power amplifier. The signal is available between the PA_OUTP and PA_OUTN outputs. The output power amplifier is a class D amplifier. It requires only a simple output filter. It is capable of driving a speaker directly if its impedance is equal or greater than 32Ω.
The Codec also includes a Direct Memory Access to the host processor data memory (0x2000 to 0x3FEF) to read and write 16-bit audio samples, defined as the read channel and the write channel. The area of the host processor data memory used the write channel is defined by the two 16-bit pointers stored in **RegDmaWrStartAddrH/-L** and **RegDmaWrStopAddrH/-L**. Similarly, the area of the host processor data memory used the read channel is defined by the two 16-bit pointers stored in **RegDmaRdStartAddrH/-L** and **RegDmaRdStopAddrH/-L**. Software engineering should make sure these two areas do not overlap with other application data otherwise this may lead to unpredictable behavior.
The use of the DMA strictly requires the host processor clock is the same as the Codec input clock which is SYS_CLOCK_IN (see 3.5.9).
To enable the read and/or write channels, the corresponding start and stop addresses must loaded into the internal address pointers from the corresponding RegDma(Read/Write)(Start/Stop)(H/L) registers and the read and/or write channel must be enabled. This is performed by setting appropriately the **RegDmaCtrl** register. These pointers are then automatically incremented at the sampling frequency defined for the audio samples, to read and/or write one samples after the other from/to the data memory.
When the DMA read or write pointer reach the value defined in **RegDmaRdStopAddrH/-L** or **RegDmaWrStopAddrH/-L** an interrupt to the host processor is generated. The interrupt service routine must stop the DMA access, write new start and stop addresses in **RegDmaRdStartAddrH/-L** and **RegDmaRdStopAddrH/-L**, or **RegDmaWrStartAddrH/-L** and **RegDmaWrStopAddrH/-L**, and enable again the read or write channel. If not,

the read or write pointer will increment beyond the stop addresses which means audio samples will be read from the application data memory, or audio samples will overwrite application data in the memory.
Note that odd settings may lead to unpredictable behavior: a) if the **RegDmaWrStartAddr** is higher then the **RegDmaWrStopAddr** then the internal pointer counter will increment until it reaches the 0xFFFF address, then restarts at 0x0000 and increments until it reaches the **RegDmaWrStopAddr**; b) the behavior is similar for the **RegDmaRdStartAddr** and **RegDmaRdStopAddr**; c) if **RegDmaWrStartAddr** = **RegDmaWrStopAddr**, and/or **RegDmaRdStartAddr** = **RegDmaRdStopAddr**, then the interrupt is immediately generated.

3.14.4CODEC Clock Source

As shown in paragraph 3.5.9, the codec is clocked by SYS_CLOCK_IN. Its frequency must be 13 MHz in order to fulfill the Bluetooth Audio specifications.

3.14.5Specifications

Symbol	Description	Min	Typ	Max	Unit	Comments
V_{MIC}	input range	0	0.8	1.5	V_p	Min/max levels on VMIC_P and VMIC_N
		-	-	1.0	V_{pp}	Differential input |VMIC_P – VMIC_N|
Preamplifier gain		5		20		x5, x10, x20, under software control
Gain error (*)		-0.5		0.5	dB	@ gain = x20
Preamplifier Noise (x5) (*)	input noise level of the preamplifier @ gain = x5	-	4.3	-	μV_{rms}	50 Hz to 4 kHz bandwidth
Preamplifier Noise (x10) (*)	input noise level of the preamplifier @ gain = x10	-	1.8	-	μV_{rms}	50 Hz to 4 kHz bandwidth
Preamplifier Noise (x20) (*)	input noise level of the preamplifier @ gain = x20	-	1.5	-	μV_{rms}	50 Hz to 4 kHz bandwidth
F_H (*)	Preamplifier high frequency roll-off	10	18	25	kHz	
Z_{Preamp} @ gain x5 (*)	Equivalent input impedance of the preamplifier	-	20	-	kΩ	gain = x5
Z_{Preamp} @ gain x10 (*)	Equivalent input impedance of the preamplifier	-	10	-	kΩ	gain = x10
Z_{Preamp} @ gain x20 (*)	Equivalent input impedance of the preamplifier	-	5	-	kΩ	gain = x20
ADC Noise (*)	equivalent input noise level of the ADC	-	13	40	μV_{rms}	
DR_{ADC} (*)	Dynamic Range ADC	-	86	-	dB	
PWM (*)	PWM output rate	-	256	-	kHz	-
DR_{DAC} (*)	Dynamic range DAC	-	84	-	dB	-
THD	Total harmonic distortion (relative to full scale)	-	-78	-65	dB	@ 1kHz, 32Ω load, with ADC and DAC in direct loopback mode, gainx5
SNR	Signal-to-Noise Ratio	-	72	-	dB	with ADC and DAC in direct loopback mode gain x5 in 4kHz bandwidth

Note 1 : Values below are specified at 25degC and for VDD_M > 2.2V unless otherwise specified
Note2 : Values marked with asterisks are not production tested and guaranteed by design.

Table 128 – Codec Specifications

3.14.6Microphone Input

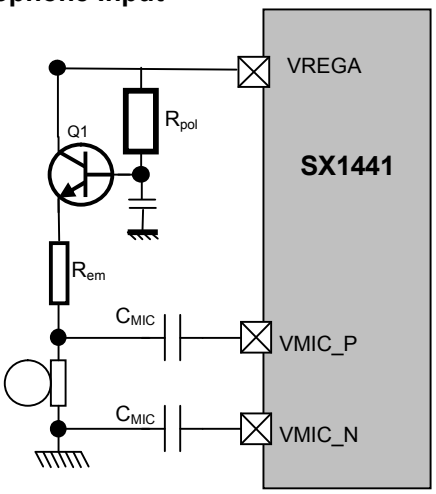


Figure 30 - typical microphone configuration

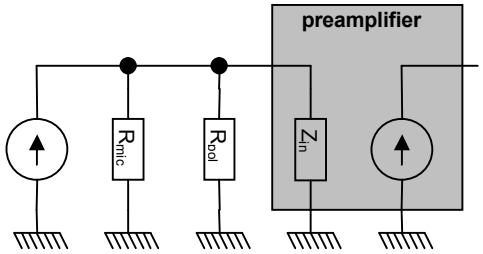


Figure 29 - Equivalent schematic for gain calculation

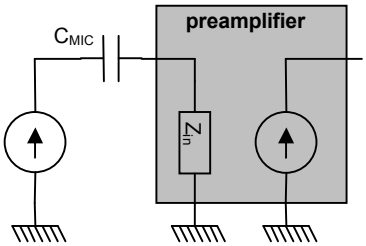


Figure 31 - Equivalent schematic for C_{MIC} calculation

The Figure 30 shows how to connect the microphone. The microphone is powered by the pin VREGA, or any clean constant voltage through the resistor R_{pol}. The two capacitors C_{MIC} remove the DC voltage level.
The Figure 29 shows the equivalent schematic for gain and dynamic range calculation. The input impedance of the preamplifier, the polarization resistor, and the internal resistor of the microphone are to be considered in parallel. The input impedance of the preamplifier varies with the preamplifier gain.

The

Figure 31 is the equivalent schematic for the C_{MIC} and low frequency roll-off calculations. The frequency is given by the relation:

$$f_L = \frac{1}{2\pi \cdot Z_{in} \cdot C_{MIC}} \times 2$$

Z_{in} varies from 5 kΩ to 20 kΩ, depending on the gain of the preamplifier. CMIC is typically about 470 nF. R_{pol} value is usually about a few kΩ and depends on the microphone.

3.14.7Speaker Output

The power amplifier operates in class D. The pins PA_OUTP and PA_OUTN output two complementary digital signals (see Figure 32) at high frequency and whose cyclic ratio is proportional to the amplitude of the audio signal. The PWM switching frequency has to be filtered to limit power consumption and risk of degradation of the speaker.

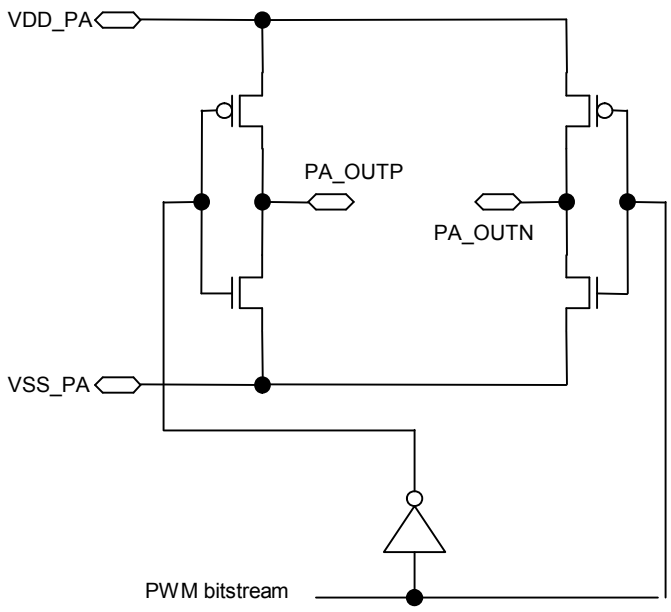


Figure 32 - Power Amplifier (PA) structure

Figure 33 shows the typical configuration of the speaker and power amplifier. VDD_PA may be connected to VREGD or any voltage lesser or equal to 1.8V.

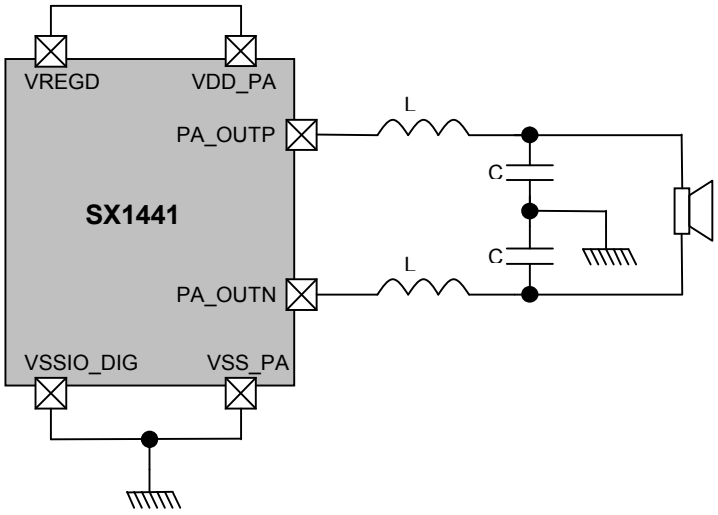


Figure 33 - Typical output filter

The output filter is a balanced 2-pole filter. Equation 5 gives a raw estimation of the values of the external inductors and capacitors as a function of the speaker impedance Z_L and the cut-off frequency f_0 of the filter. f_0 is usually 4 kHz, as defined by the Bluetooth audio specifications. Z_L may be very dependant on the speaker.

$$C = \frac{2}{\sqrt{2} \cdot Z_L \cdot \omega 0}, \quad L = \frac{\sqrt{2} \cdot Z_L}{2 \cdot \omega 0} \quad where \quad \omega 0 = 2 \cdot \pi \cdot f 0$$

Equation 5 – Typical L and C values for the speaker output filter

Equation 5 gives best estimation for the component values, since it does not take into account the resistance of the inductor and assumes that the speaker impedance is resistive and constant over the whole frequency range. However, the estimation is good starting point for the optimization of the components.
Choosing the filter cutoff frequency f_0 = 4 kHz and Z_L = 32 Ω as typical values, gives L = 860 µH and C = 1 µF.

3.14.8 Volume Control

3.14.8.1 Description

The data supplied to the DAC may be scaled digitally before analog conversion to provide volume amplification or attenuation through the use of the **RegVolCtrl** and **RegVolCmdDAC** registers. The provided gain ranges from -22.5dB (attenuation) to +22.5dB (amplification) in 1.5dB steps. Because such amplification is digital only, this feature should be used only after optimum setting is chosen on the analog gain settings, so as not to increase quantification noise level while maintaining proper SNR+THD levels.
Similarly, the data read from the ADC may be scaled before beeing transferred to the CPU in a similar manner. Similarly to the DAC path, analog settings should be set at their optimum levels before using the digital volume control.

3.14.8.2 ADC and DAC Scaling

Once volume control is enabled through setting the bit **VolCtrEn** in the **RegVolCtrl** register, the data from the ADC (respectively DAC) path is digitally multiplied from a scaling factor derived from the **ADCscaling** bits from the **RegVolCmdADC** register (respectively **DACScaling** from the **RegVolCmdDAC** register). The **ADCscaling** allows the data to be amplified by up to 22.5dB or conversely attenuated by (up to) -22.5dB. The **ADCscaling** is applied onto the data per formula :

$$ADCData\_scaled = sign(ADCScaling) * \max(abs(ADCScaling),15) * 1.5dB * ADCData$$

Please note that although only 5 bits are relevant to the scaling, the ADCScaling value is an 8 bit signed integer.

3.15 DEBUG INTERFACE

3.15.1 Description

The debug interface can be used to observe and/or force the HCI UART and Codec signals. It can also be used as a GPIO port.

3.15.2 Register Map

Name	Address (Hex)
RegDbgDir	0x0078
RegDbgOut	0x0079
RegDbgIn	0x007A
RegDbgMode	0x007B

Table 129 – Debug interface registers

Pos	RegDbgDir	r/w	Reset	Function
7-0	DbgDir[7:0]	rw	00000000	DBG[7:0] pad direction GPIO mode: 1 = output, 0 = input Debug mode: 1 = observation, 0 = force

Table 130 - RegDbgDir register

Pos	RegDbgOut	r/w	Reset	Function
7-0	DbgOut[7:0]	rw	00000000	DBG[7:0] pad output value. Valid only in GPIO mode

Table 131 - RegDbgOut register

Pos	RegDbgIn	r/w	Reset	Function
7-0	DbgIn[7:0]	r	00000000	DBG[7:0] pad input value. Valid only in GPIO mode

Table 132 - RegDbgOut register

Pos	RegDbgMode	r/w	Reset	Function
7	DbgMode	rw	0	0 = GPIO 1 = Debug mode
6-0	-	r	00000000	reserved

Table 133 - RegDbgMode register

3.15.3Pins Mapping

Pin	Signal (Bluetooth macro side)
DBG[7]	HCI_ DBG_ TX
DBG[6]	HCI_ DBG_RX
DBG[5]	HCI_ DBG_RTS
DBG[4]	HCI_ DBG_CTS
DBG[3]	PCM_ DBG_D_IN
DBG[2]	PCM_ DBG_D_OUT
DBG[1]	PCM_ DBG_CLK
DBG[0]	PCM_ DBG_FSYNC

Table 134 – Pins mapping

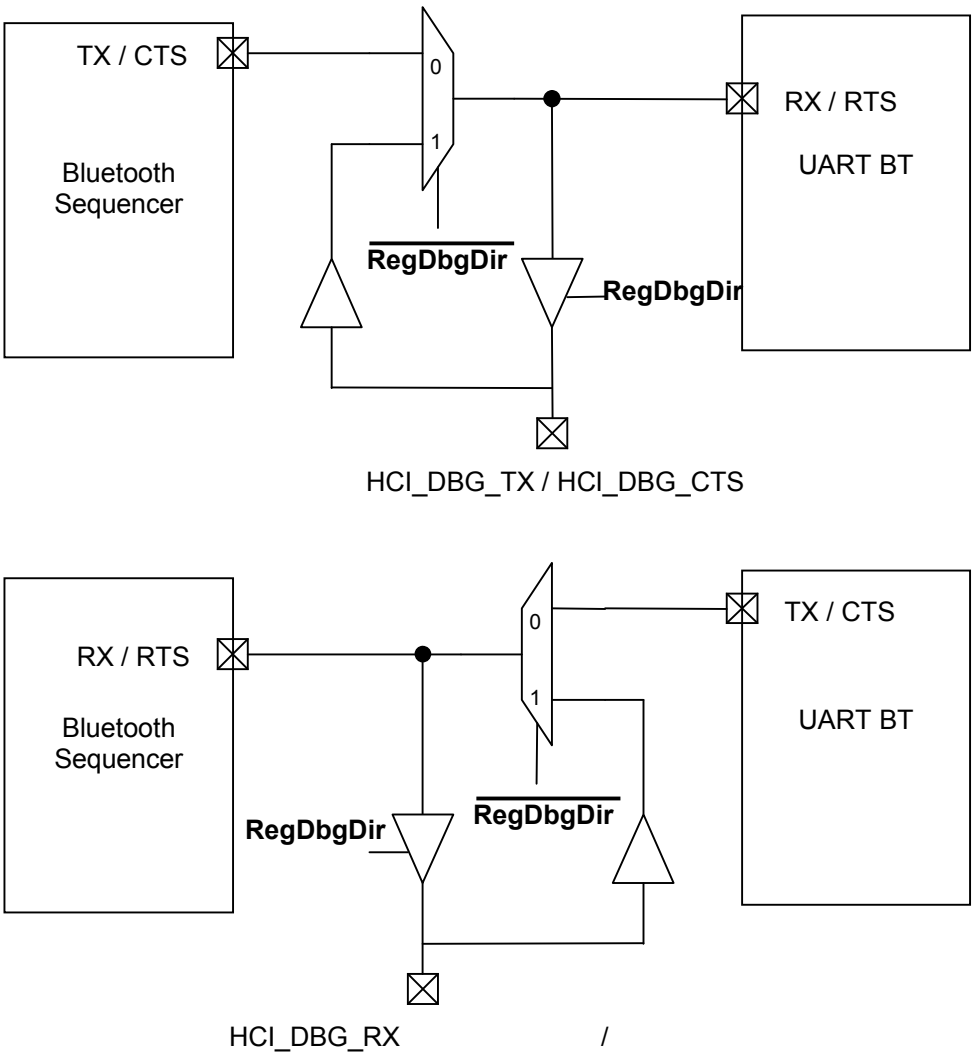


Figure 34 – HCI Debug Interface block schematics

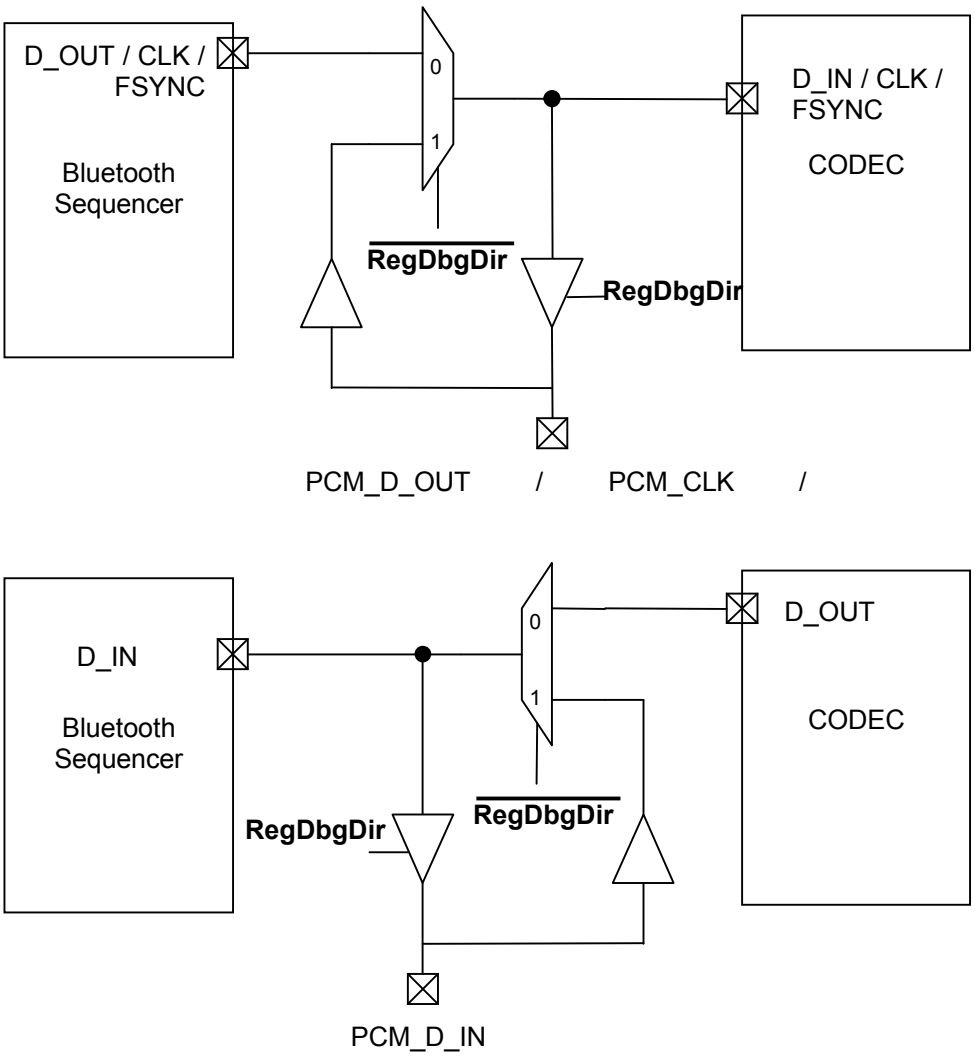


Figure 35 – Codec Debug Interface block schematics

3.15.4Configuration

The debug interface has 2 modes, GPIO mode (default) and Debug mode.

3.15.4.1 GPIO Mode

To enter GPIO mode the MSB bit of **RegDbgMode** register has to be set to 0. The default value after reset is low (0).

The direction of each bit within DBG[7:0] (input only or input/output) can be individually set using the **RegDbgDir** register. If **RegDbgDir[i]** = 1, both the input and output buffers are active on the corresponding pin. If **RegDbgDir[i]** is 0, the corresponding DBG pin is an input only and the output buffer is in high impedance. After reset DBG is in input only mode; **RegDbgDir[i]** is reset to 0.

The input values of DBG are available in **RegDbgIn** (read only). Reading is always direct - there is no debounce function. In case of possible noise on input signals, a software debouncer with polling or an external hardware filter has to be implemented. The input buffer is also active when the port is defined as output and allows reading back of the effective value on the pin.

Data stored in **RegDbgOut** are output at DBG if **RegDbgDir[i]** is 1. The default value after reset is low (0).

3.15.4.2 Debug Mode

To enter Debug mode the MSB bit of **RegDbgMode** register has to be set to 1.

In Debug mode DBG pins can be observed / forced. If **RegDbgDir[i]** = 1 the corresponding pin is set in observation mode. If **RegDbgDir[i]** = 0 the corresponding pin can be forced from outside.

3.15.5 Configuration Examples

3.15.5.1 Use SX1441 as XE1401
Set the Debug interface registers as follows:

RegDbgDir = 0xAF
RegDbgMode = 0x80

3.15.5.2 Observe HCI Traffic
Set the Debug interface registers as follows:

RegDbgDir = 0xFF
RegDbgMode = 0x80

3.16 DEVELOPMENT / DEBUG ON CHIP

This is the interface with the SX1441 development tool. It includes the DOC_SCK and DOC_SDIO pins. They are powered by VDDIO_DIG and VSSIO_DIG.
When in normal operation in applications, DOC_SCK and DOC_SDIO should remain unconnected (N.C)
When operated in development / debug mode DOC_SCK and DOC_SDIO, in addition to VDDIO_DIG and VSSIO_DIG, are connected to the SX1441 development tools through the appropriate interface. For more details, please contact Semtech technical support.

4 ELECTRICAL SPECIFICATIONS

4.1 ABSOLUTE MAXIMUM RATINGS

Stress above the limits listed in the following table may cause permanent failure. Exposure to absolute ratings for extended time periods may affect device reliability. The limiting values are in accordance with the Absolute Maximum Rating System (IEC 134). All voltages are referenced to ground (VSS_M).

Symbol	Parameter	Conditions	Min	Max	Unit
	Supply voltage: VDD_M	-	-0.3	3.65	V
	Codec power amplifier supply voltage: VDD_PA	-	-0.3	2.0	V
T_{stor}	Storage temperature	-	-65	150	°C
V_{es}	Electrostatic handling	See (1)	-	2000	V
I_{lup_dig}	Latchup free trigger current on digital I/O pins	See (2)	-20.0	100.0	mA
I_{lup_ana}	Latchup free trigger current on analog and supply pins	See (2)	-100.0	100.0	mA

(*1) Tested according to MIL883C Method 3015.6 (Standardized Human Body Model: 100 pF, 1500Ω, 3 pulses, protection related to substrate).
(*2) Tested according to JEDEC Standard 17

Table 135 – Absolute maximum ratings

4.2 RECOMMENDED OPERATING CONDITIONS

All voltages are referenced to ground (VSS_M). Typical operating conditions are at 25 °C in typical configuration. Operating ranges define the limits for functional operation and parametric characteristics of the device as described in this section. Functionality outside these limits is not implied.

Symbol	Description	Min	Typ	Max	Unit	Comments
T_{amb}	Operating ambient temperature	-40		85	°C	
VDD_M	Main power supply	2.2	-	3.6	V	VDD_M
VDDBAT	Battery end-of-life sensor	0.7	-	1.98	V	see para. 3.3.6
VDD_DIG	Digital core voltage	1.62	-	1.98	V	usually connected to VREGD
VDD_PA	Codec power amplifier supply voltage	1.2	-	1.98	V	-
VDD_ANA	Analog core voltage	1.62	-	1.98	V	usually connected to VREGA
VDDIO	Radio I/O voltage level	1.62	-	3.6	V	-
VDDIO_DIG	Digital I/O voltage level	1.62	-	3.6	V	-

Table 136 – Operating supply ranges

Symbol	Description	Min	Typ	Max	Unit	Comments
V_{IH}	input logic level high	0.7*VDDIO_DIG	-	VDDIO_DIG	V	
V_{IL}	input logic level low	VSSIO_DIG	-	0.3*VDDIO_DIG	V	
V_{OH}	output high voltage	VDDIO_DIG–0.2	-	VDDIO_DIG	V	I_{OH}=-6mA, VDDIO_DIG=3.6V
V_{OL}	output low voltage	VSSIO_DIG	-	VSSIO_DIG+0.2	V	I_{OL}=6mA, VDDIO_DIG=3.6V
R_{pu}	internal pull-up resistor		80		kΩ	
C_{in}	input capacitance		3.5		pF	

Note : Values marked with asterisks are not production tested and guaranteed by design.

Table 137 - Digital I/O's specifications

4.3 SUPPLY CONFIGURATION, POWER CONSUMPTION

Several configurations are possible to supply power to the SX1441 and the associated XE1413 or CX72302 radio chip. The paragraphs 4.3.1 and 4.3.2 below show two recommended examples for 3 and 1.8V supply.
All typical averaged values are measured at room temperature (20°C) using the XE1413 or the CX72303 Bluetooth radio in a Class 2 (typ. +2 dBm) mode. A high frequency system clock of 13 MHz provided by the radio chip is used. The default UART speed is 115 kbits/s.

4.3.1 3V Supply Configuration, Single 13 MHz Crystal Oscillator

The on chip voltage regulators supply VDD_DIG, VDD_PA, VDD_ANA, VDDIO, and VDDIO_DIG, as well as the radio.

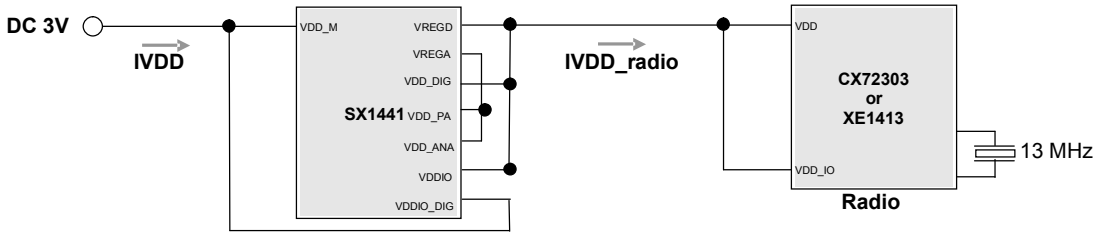


Figure 36 – Current measurement diagram, 3V supply, single 13 MHz crystal oscillator

Modes	IVDD avg. [mA]	IVDD peak [mA]	Comments
SCO link, HV3 (Master/Slave), sniff mode	12.5		1.28s interval sniff mode
SCO link, HV3 (Slave)	15.6		-
SCO link, HV1 (Master/Slave)	23.3		-
ACL link maintained (Master)	7.0		Poll interval 25 ms
ACL link maintained (Slave)	11.6		Poll interval 25 ms
ACL link connection (Master/Slave), sniff mode enabled, no data transfer	1.41		20 ms wake-up time, 200 ms interval 115 kbits/s UART
ACL link connection (Master/Slave), sniff mode enabled, no data transfer	0.65		20 ms wake-up time, 1.28s interval 115 kbits/s UART
ACL link, DM1 (Master/Slave) (*1)	14.9		115 kbits/s UART, continuous transmit/receive
ACL link, DH5 (Master/Slave) (*2)	21		115 kbits/s UART, continuous transmit/receive
Page scan		32.1	Peak duration: 12 ms, 1.28s interval 115 kbits/s UART, 32 kHz clock derived from 13 MHz radio Xtal oscillator
Inquiry & page scan		33.1	Peak duration: 2 x 12 ms, 1.28s interval 115 kbits/s UART, 32 kHz clock derived from 13 MHz radio Xtal oscillator
Parked (Slave)		32.1	Peak duration: 2 x 4 ms, 1.28s beacon interval
Reset	0.01		Excluding leakage current

(*1) Each slot is used for TX/RX protocol (1 TX for 1 RX)
(*2) Each slot is used for TX/RX protocol (5 TX for 5 RX)

Table 138 – Typical system current consumption, 3V supply, single 13 MHz crystal

4.3.2 1.8V supply configuration, single 13 MHz crystal oscillator

The on chip regulated voltage units are not used; a stabilized 1.8V supply is used for the complete system.

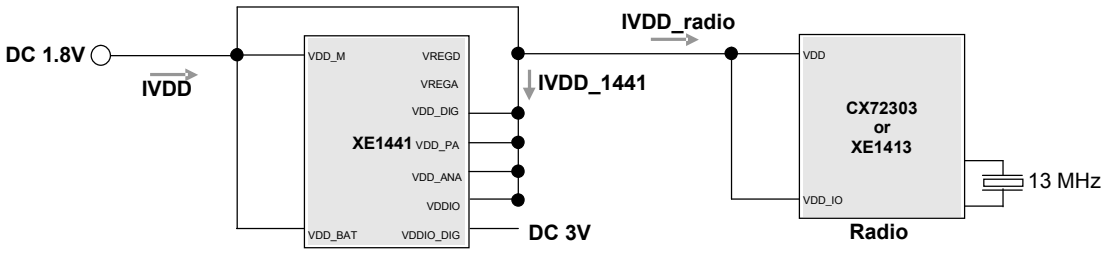


Figure 37 – Current measurement diagram, 1.8V supply, single 13 MHz crystal oscillator

5 APPLICATION SCHEMATICS – BLUETOOTH HEADSET

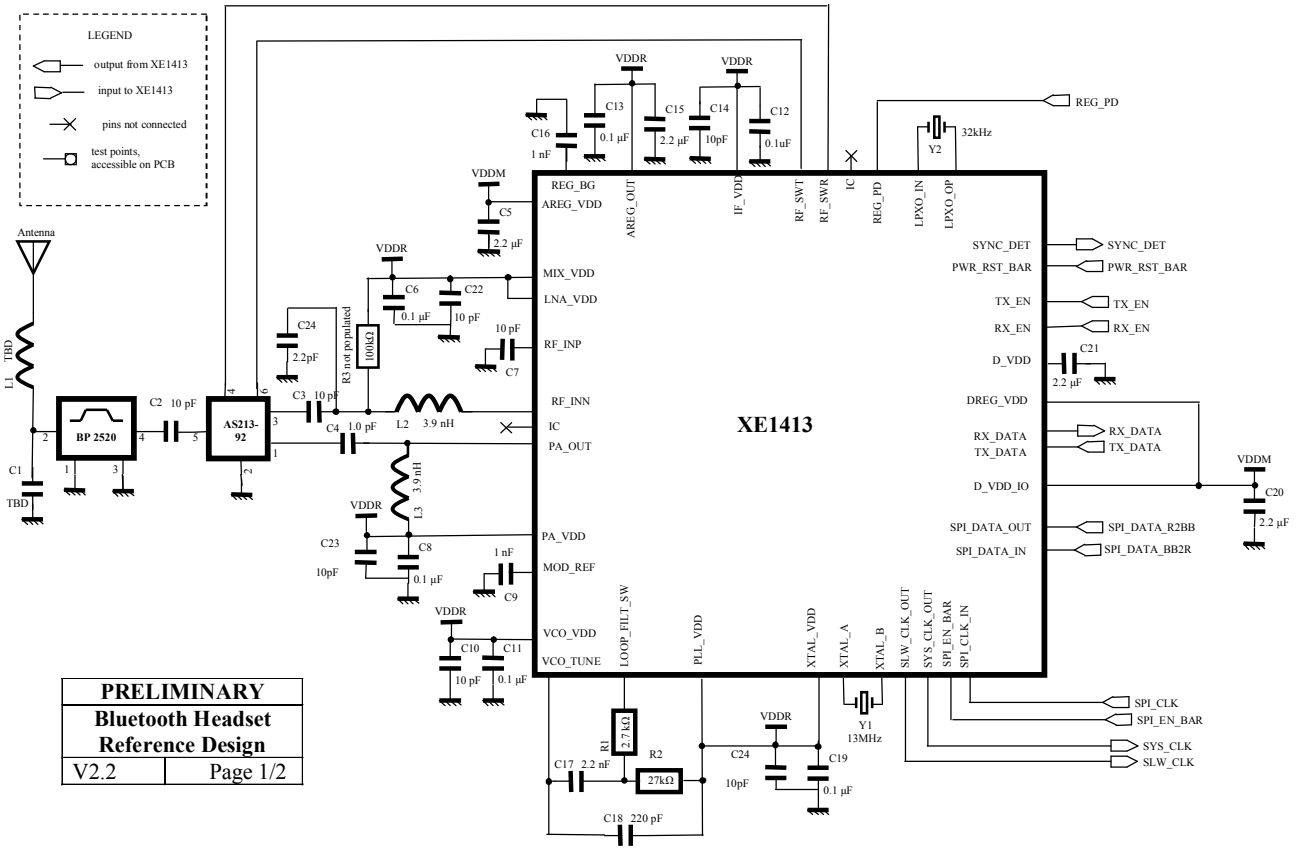


Figure 38 - Headset application schematic, part 1 of 2

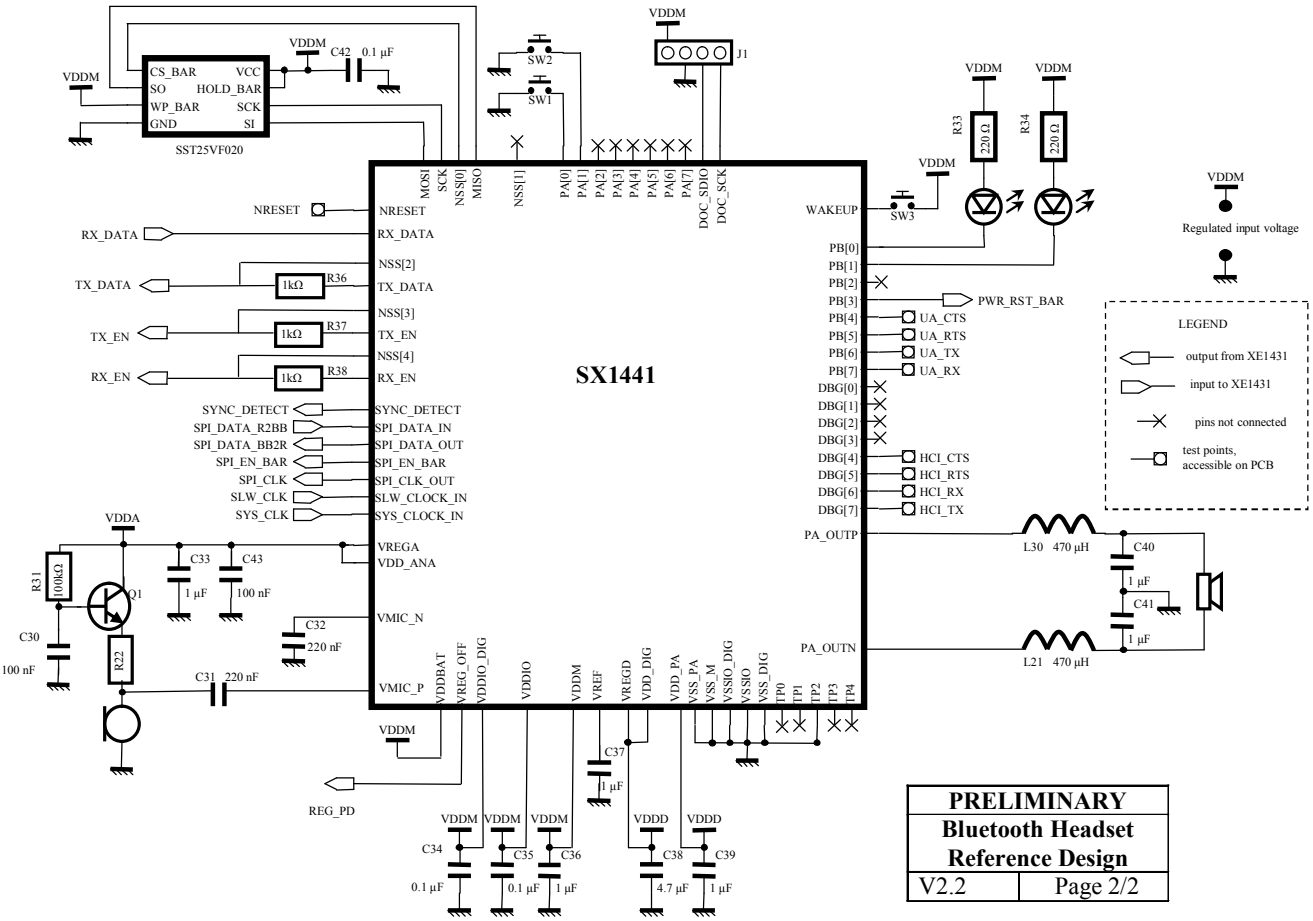


Figure 39 - Headset application schematic, part 2 of 2

6 PACKAGING INFORMATION – 72-PIN LFBGA

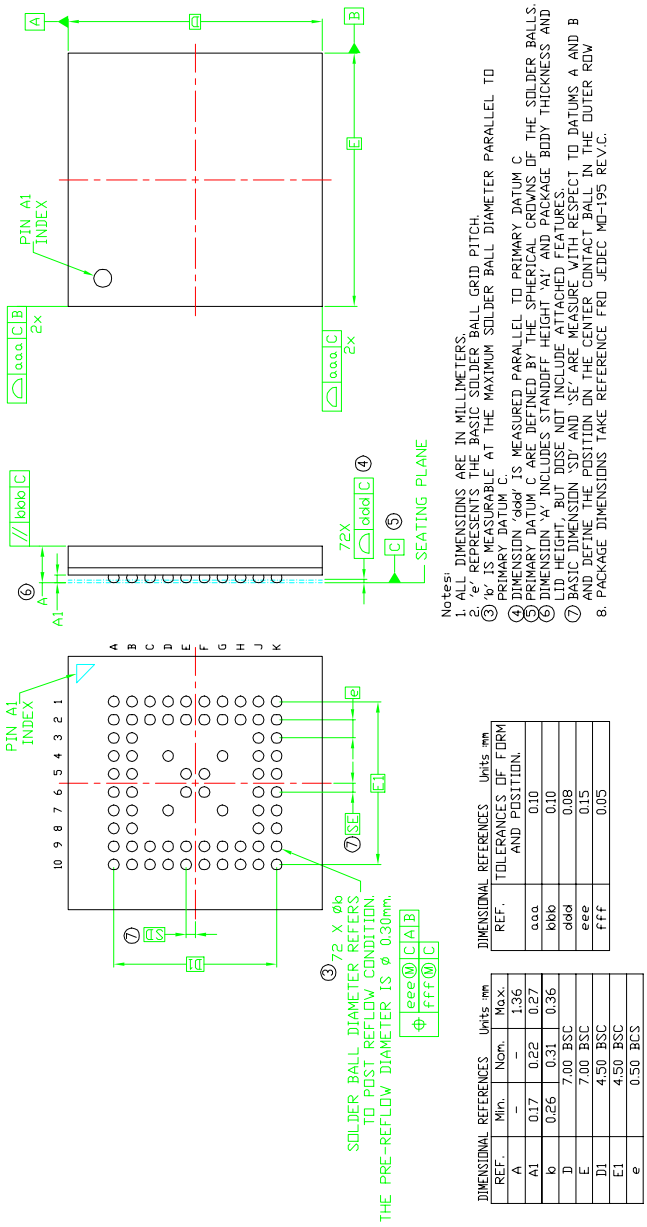


Figure 40 - Mechanical data for 72 pins LFBGA package (7mmX7mm)

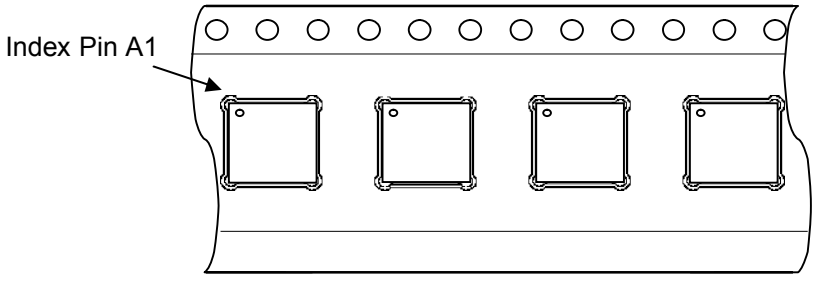


Figure 41 - Tape & Reel information

7 SOLDERING REFLOW PROFILE

The soldering reflow profile used by SX1441 is described in the standard IPC/JEDEC J-STD-020C. For detailed information click on the link http://www.jedec.org/download/search/jstd020c.pdf

8 REFERENCE DOCUMENTS

[1] Bluetooth Specification Version 1.2
[2] XE1413 - 1.8V ultra low power Bluetooth RF Transceiver datasheet, Semtech Neuchatel.
[3] CoolRISC 816 8-bit Microprocesor Core Hardware and Software Reference Manual, version 4.5, SEMTECH SA.

9 NOTICE, TRADEMARKS

Semtech reserves the right to make changes to its products or service without notice. Before using the product, Please make sure that the information being referred to is up-to-date.
Bluetooth is a SIG registered trademark, used under license by Semtech
EasyBlue is a trademark of Semtech
CoolRISC is a registered trademark of Semtech

EXHIBIT A – SYSTEM REGISTERS SUMMARY

See paragraphs 3.2, 3.3.2, 3.4.2, 3.5.2, 3.6.2, 3.7.2, 3.8.2, 3.9.2, 3.10.2, 3.11.2, 3.12.2, 3.13.9, 3.14.2, and 3.15.2 for detailed information on registers.

Subsystem	Register Name	Address (Hex)
Reset & Clock	RegSysCtrl	0x0010
	RegSysClock	0x0012
	RegSysMisc	0x0013
	RegSysWd	0x0014
	RegSysPre0	0x0015
	RegSysRcTrim1	0x001B
	RegSysRcTrim2	0x001C
-	reserved	0x001D-0x001F
GPIO's - Port A	RegPAIn	0x0020
	RegPADebounce	0x0021
	RegPAEdge	0x0022
	RegPAPullup	0x0023
	RegPARes0	0x0024
	RegPARes1	0x0025
	RegPACtrl	0x0026
	RegPASnapToRail	0x0027
GPIO's – Port B	RegPBOut	0x0028
	RegPBIn	0x0029
	RegPBDir	0x002A
	RegPBOpen	0x002B
	RegPBPullup	0x002C
-	reserved	0x002D-0x002F
Application UART	RegUartFifoCtrl	0x0030
	RegUartFifoBaud	0x0031
	RegUartFifoTx	0x0032
	RegUartFifoTxSta	0x0033
	RegUartFifoRx	0x0034
	RegUartFifoRxSta	0x0035
	RegUartFifoMisc	0x0036
-	reserved	0x0037-0x003B
Events Controller	RegEvn	0x003C
	RegEvnEn	0x003D
	RegEvnPriority	0x003E
	RegEvnEvn	0x003F
Interrupts Controller	RegIrqHig	0x0040
	RegIrqMid	0x0041
	RegIrqLow	0x0042
	RegIrqEnHig	0x0043
	RegIrqEnMid	0x0044
	RegIrqEnLow	0x0045
	ReqIrqPriority	0x0046
	ReqIrqIrq	0x0047
Power Management	RegPmgtVrega	0x0048
	RegPmgtVregd	0x0049
	RegPmgtEol	0x004A
-	reserved	0x004B-0x004F
HCI UART	RegHUartFifoCtrl	0x0050
	RegHUartFifoBaud	0x0051
	RegHUartFifoTx	0x0052

Subsystem	Register Name	Address (Hex)
	RegHUartFifoTxSta	0x0053
	RegHUartFifoRx	0x0054
	RegHUartFifoRxSta	0x0055
	RegHUartFifoMisc	0x0056
-	reserved	0x0057
Timers/Counters	RegCntA	0x0058
	RegCntB	0x0059
	RegCntC	0x005A
	RegCntD	0x005B
	RegCntCtrlCk	0x005C
	RegCntConfig1	0x005D
	RegCntConfig2	0x005E
	RegCntOn	0x005F
-	reserved	0x0060-0x0063
Codec Volume Control	RegVolCtrl	0x0064
	RegVolCmdADC	0x0065
	RegVolCmdDAC	0x0066
-	reserved	0x0067
SPI	RegSpiControl	0x0068
	RegSpiStatus	0x0069
	RegSpiDataOut	0x006A
	RegSpiDataIn	0x006B
	RegSpiPullup	0x006C
	RegSpiDir	0x006D
	RegSpiSlvSel	0x006E
-	reserved	0x006F-0x0077
Debug Interface	RegDbgDir	0x0078
	RegDbgOut	0x0079
	RegDbgIn	0x007A
	RegDbgMode	0x007B
Bluetooth Sequencer	RegBtmCtrl1	0x007C
	RegBtmCtrl2	0x007D
-	reserved	0x007E-0x00DF
Codec	RegCodecCtrl	0x00E0
	reserved	0x00E1
	RegDACSampleH	0x00E2
	RegDACSampleL	0x00E3
	RegADCSampleH	0x00E4
	RegADCSampleL	0x00E5
	RegDmaRdStartAddrH	0x00E6
	RegDmaRdStartAddrL	0x00E7
	RegDmaRdStopAddrH	0x00E8
	RegDmaRdStopAddrL	0x00E9
	RegDmaWrStartAddrH	0x00EA
	RegDmaWrStartAddrL	0x00EB
	RegDmaWrStopAddrH	0x00EC
	RegDmaWrStopAddrL	0x00ED
	RegDmaCtrl	0x00EE
	RegCodecDataFlow	0x00EF
	reserved	0x00F0
	RegADCGain	0x00F1
	RegCodecPaMute	0x00F9
-	reserved	0x00FA-0x00FE

Subsystem	Register Name	Address (Hex)
-	reserved	0x3FF0–0x3FFF

Contact Information

Semtech Corporation
Wireless and Sensing Products Division
200 Flynn Road, Camarillo, CA 93012
Phone (805) 498-2111 Fax : (805) 498-3804