



Advanced Data Sheet – JN513x

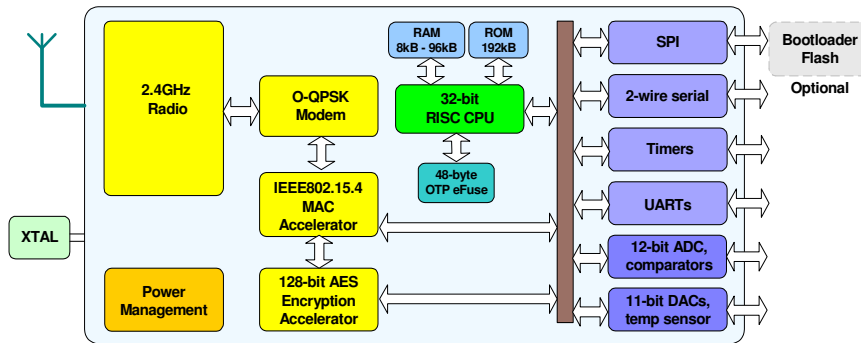
IEEE802.15.4 and ZigBee Wireless Microcontrollers

Overview

The JN513x are a family of low power, low cost wireless microcontrollers suitable for IEEE802.15.4 and ZigBee applications. Each device integrates a 32-bit RISC processor, with a fully compliant 2.4GHz IEEE802.15.4 transceiver, 192kB of ROM, a selection of RAM sizes from 8kB to 96kB, and a rich mixture of analogue and digital peripherals.

The cost sensitive ROM/RAM architecture supports the storage of system software, including protocol stacks, routing tables and application code/data. Each device has hardware MAC and AES encryption accelerators, power saving and timed sleep modes, and mechanisms for security key and program code encryption. These features all make for a highly efficient, low power, single chip wireless microcontroller for battery-powered applications.

Block Diagram



Features: Transceiver

- 2.4GHz IEEE802.15.4 compliant
- 128-bit AES security processor
- MAC accelerator with packet formatting, CRCs, address check, auto-acks, timers
- Integrated power management and sleep oscillator for low power
- On-chip power regulation for 2.2V to 3.6V battery operation
- Deep sleep current <0.4µA
- Sleep current with active beacon timer <1.5µA
- Needs minimum of external components (< US\$1 cost)
- Rx current 39mA
- Tx current 39mA
- Receiver sensitivity -97dBm
- Transmit power +3dBm

Features: Microcontroller

- 32-bit RISC processor sustains 32MIPs with low power
- 192kB ROM stores system code, including protocol stack
- 8kB, 16kB, 32kB or 96kB RAM stores system data and optionally bootloaded program code
- 48-byte OTP eFuse, stores MAC ID on-chip, offers AES based code encryption feature
- 4-input 12-bit ADC, 2 11-bit DACs, 2 comparators
- 2 Application timer/counters, 3 system timers
- 2 UARTs (one for debug)
- SPI port with 5 selects
- 2-wire serial interface
- Up to 21 GPIO

Industrial temperature range
(-40°C to +85°C)

8x8mm 56-lead QFN – pin compatible with JN5121

Lead-free and RoHS compliant

Benefits

- Single chip integrates transceiver and microcontroller for wireless sensor networks
- Cost sensitive ROM/RAM architecture, meets needs for volume application
- System BOM is low in component count and cost
- Hardware MAC ensures low power consumption and low processor overhead
- Extensive user peripherals
- Pin compatible with JN5121 for easy migration

Applications

- Robust and secure low power wireless applications
- Wireless sensor networks, particularly IEEE802.15.4 and ZigBee systems
- Home and commercial building automation
- Remote Control
- Toys and gaming peripherals
- Industrial systems
- Telemetry and utilities (e.g. AMR)

Contents

1 Introduction	1
1.1 Wireless Microcontroller	1
1.2 Wireless Transceiver	1
1.3 RISC CPU and Memory	1
1.4 Peripherals	2
1.5 Block Diagram	3
2 Pin Configurations	4
2.1 Pin Assignment	5
2.2 Pin Descriptions	6
2.2.1 Power Supplies	6
2.2.2 Reset	6
2.2.3 16MHz System Clock	6
2.2.4 Radio	6
2.2.5 Analogue Peripherals	6
2.2.6 Digital Input/Output	7
3 CPU	8
4 Memory Organisation	9
4.1 ROM	10
4.2 RAM	10
4.3 OTP eFuse Memory	10
4.4 External Memory	11
4.4.1 Secure External Memory Encryption	11
4.5 Peripherals	11
4.6 Unused Memory Addresses	11
5 System Clocks	12
5.1 16MHz Oscillator	12
5.2 32kHz Oscillator	12
6 Reset	13
6.1 Power-on Reset	13
6.2 External Reset	13
6.3 Software Reset	14
6.4 Brown-out Detect	14
7 Interrupt System	15
7.1 System Calls	15
7.2 Processor Exceptions	15
7.2.1 Bus Error	15
7.2.2 Alignment	15
7.2.3 Illegal Instruction	15
7.3 Hardware Interrupts	16
8 Wireless Transceiver	17
8.1 Radio	17
8.1.1 Radio External components	18
8.1.2 Antenna Diversity	18
8.2 Modem	19

8.3	Baseband Processor	20
8.3.1	Transmit	20
8.3.2	Reception	20
8.3.3	Auto Acknowledge	21
8.3.4	Beacon Generation	21
8.3.5	Security	21
8.4	Security Coprocessor	21
9	Digital Input/Output	22
10	Serial Peripheral Interface	23
10.1	Programming Example	25
11	Intelligent Peripheral Interface	27
11.1	Data Transfer Format	27
11.2	JN513x Initiated Data Transfer	28
11.3	Remote Processor Initiated Data Transfer	28
12	Timers	29
12.1	Peripheral Timer / Counters	29
12.1.1	Pulse Width Modulation Mode	30
12.1.2	Capture Mode	30
12.1.3	Counter / Timer Mode	31
12.1.4	Delta-Sigma Mode	31
12.1.5	Timer / Counter Application	32
12.2	Tick Timer	33
12.3	Wakeup Timers	34
12.3.1	RC Oscillator Calibration	34
12.3.2	External 32kHz Clock Source	35
13	Serial Communications	36
13.1	Interrupts	37
13.2	UART Application	37
13.3	Programming Example	38
14	Two-Wire Serial interface	39
14.1	Connecting Devices	40
14.2	Multi-Master Operation	40
14.3	Clock Stretching	41
14.4	Programming Example	41
15	Analogue Peripherals	43
15.1	Analogue to Digital Converter	44
15.1.1	Operation	44
15.1.2	Supply Monitor	44
15.1.3	Temperature Sensor	45
15.1.4	Programming Example	45
15.2	Digital to Analogue Converter	45
15.2.1	Operation	45
15.2.2	Programming Example	46
15.3	Comparators	46
16	Power Management and Sleep Modes	48

16.1	Operating Modes	48
16.1.1	Power Domains	48
16.2	Active Processing Mode	48
16.2.1	CPU Doze	48
16.3	Sleep Mode	48
16.3.1	Wakeup Timer Event	49
16.3.2	DIO Event	49
16.3.3	Comparator Event	49
16.4	Deep Sleep Mode	49
17	Electrical Characteristics	50
17.1	Maximum ratings	50
17.2	DC Electrical Characteristics	50
17.2.1	Operating Conditions	50
17.2.2	DC Current Consumption	51
17.2.3	I/O Characteristics	52
17.3	AC Characteristics	52
17.3.1	Reset	52
17.3.2	Brown-out Detect	53
17.3.3	SPI Timing	54
17.3.4	Two-wire serial interface	54
17.3.5	Power Down and Wake-Up timings	55
17.3.6	32kHz Oscillator	56
17.3.7	16MHz Crystal Oscillator	56
17.3.8	Analogue to Digital Converters	56
17.3.9	Digital to Analogue Converters	57
17.3.10	Comparators	58
17.3.11	Temperature Sensor	59
17.3.12	Radio Transceiver	59
Appendix A	Mechanical and Ordering Information	63
A.1	56pin QFN Package Drawing	63
A.2	Ordering Information	64
A.3	Device Package Marking	65
A.4	Tape and Reel Information	66
A.4.1	Tape Orientation and Dimensions	66
A.4.2	Reel Information: 7" Reel	67
A.4.3	Reel Information: 13" Reel	68
A.4.4	Dry Pack Requirement for Moisture Sensitive Material	68
A.5	PCB Design and Reflow Profile	69
Appendix B	Development Support	70
B.1	Crystal Oscillator	70
B.1.1	Crystal Equivalent Circuit	70
B.1.2	Crystal Load Capacitance	71
B.1.3	Crystal ESR and Required Transconductance	71
B.2	16MHz Oscillator	73
B.3	Applications Schematic	74
Appendix C		75
	Related Documents	75
	RoHS Compliance	75

Jennic

Status Information	75
Disclaimers	76
Version Control	76
Contact Details	77

1 Introduction

The JN513x is an IEEE802.15.4 wireless microcontroller that provides a fully integrated solution for applications using the IEEE802.15.4 standard in the 2.4 - 2.5GHz ISM frequency band [1], including ZigBee. It includes all of the functionality required to meet the IEEE802.15.4 specification and has additional processor capability to run a wide range of applications including but not limited to Remote Control, Home and Building Automation, Toys and Gaming.

The device includes a Wireless Transceiver, RISC CPU, on-chip memory and an extensive range of peripherals.

1.1 Wireless Microcontroller

Applications that transfer data wirelessly tend to be more complex than wired ones. Wireless protocols make stringent demands on frequencies, data formats, timing of data transfers, security and other issues. Application development must consider the requirements of the wireless network in addition to the product functionality and user interfaces. To minimise this complexity, Jennic provides a series of software libraries that control the transceiver and peripherals of the JN513x. These libraries, with functions called by an Application Programming Interface (API) remove the need for the developer to understand wireless protocols and greatly simplify the programming complexities of power modes, interrupts and hardware functionality. In addition, the JN513x is expected to be programmed in the C high-level language and debugged using the JN5 series software developer kit.

In view of the above, the register details of the JN513x are not provided in the datasheet and access to all peripherals is gained using API calls to the peripheral library. Extensive reference to such calls is made throughout the datasheet and the convention used is to format the function call in the courier font e.g. `vAHI_Init()`. Full details of these function calls can be found in the JN-RM-2001 Integrated Peripherals API [2].

An IEEE802.15.4 compliant wireless network can be developed using the IEEE802.15.4 MAC library described in JN-RM-2002 802.15.4 Stack [3]. Applications over simple (point-point, star or tree) wireless networks can use this library directly or more complex wireless mesh networks such as ZigBee or IPv6 can be built on top of the IEEE802.15.4 library.

1.2 Wireless Transceiver

The Wireless Transceiver is highly integrated and, together with the IEEE802.15.4 MAC library requires little knowledge of RF or wireless design.

The Wireless Transceiver comprises a low-IF 2.45GHz radio, an O-QPSK modem, a baseband controller and a security coprocessor. The radio has a 200Ω resistive differential antenna port that includes all the required matching components on-chip, allowing a differential antenna to be connected directly to the port, minimising the system BOM costs. Connection to a single ported antenna can be achieved using a 200/50Ω 2.45GHz balun. In addition, the radio also provides an output to control transmit-receive switching of external devices such as power amplifiers allowing applications that require increased transmit power to be realised very easily.

The Security coprocessor provides hardware-based 128-bit AES-CCM, CBC⁽¹⁾, CTR and CCM* processing as specified by the 802.15.4b standard. It does this in-band on packets during transmission and reception, requiring minimal intervention from the CPU. It is also available for off-line use under software control for encrypting and decrypting packets generated by software layers such as Zigbee and user applications. This means that these algorithms can be off-loaded by the CPU, increasing the processor bandwidth available for user applications.

The transceiver elements (radio, modem and baseband) work together to provide 802.15.4 Medium Access Control under the control of a protocol stack supplied with the device as a software library. Applications incorporating IEEE802.15.4 functionality can be rapidly developed by combining user-developed application software with this library. The facilities provided by this library to applications together with examples of their use are described in more detail in [3].

(1) AES-CBC processing is only available off-line for use under software control.

1.3 RISC CPU and Memory

A 32-bit RISC CPU allows software to be run on-chip, its processing power being shared between the IEEE802.15.4 MAC protocol, other higher layer protocols and the user application. The JN513x has a unified memory architecture, code memory, data memory, peripheral devices and I/O ports are organized within the same linear address space.

The device contains 192kBytes of ROM, a choice of 8k, 16k, 32k or 96kBytes of RAM and a 48-byte OTP eFuse memory.

1.4 Peripherals

The following peripherals are available on-chip:

- Master SPI port with five select outputs
- Two UARTs
- Two programmable Timer/Counters with capture/compare facility
- Two programmable Sleep Timers and a Tick Timer
- Two-wire serial interface (compatible with SMBus and I²C)
- Slave SPI port (shared with digital I/O)
- Twenty-one digital I/O lines (multiplexed with UARTs, timers and SPI selects)
- Four-channel, 12-bit, 100ksps Analogue-to-Digital converter
- Two 11-bit Digital-to-Analogue converters
- Two programmable analogue comparators
- Internal temperature sensor and battery monitor

User applications access the peripherals using the Hardware Peripheral Library with a simple API. This allows applications to use a tested and easily understood view of the peripherals allowing rapid system development. The JN-RM-2001 Integrated Peripherals API [2] describes this interface in more detail.

1.5 Block Diagram

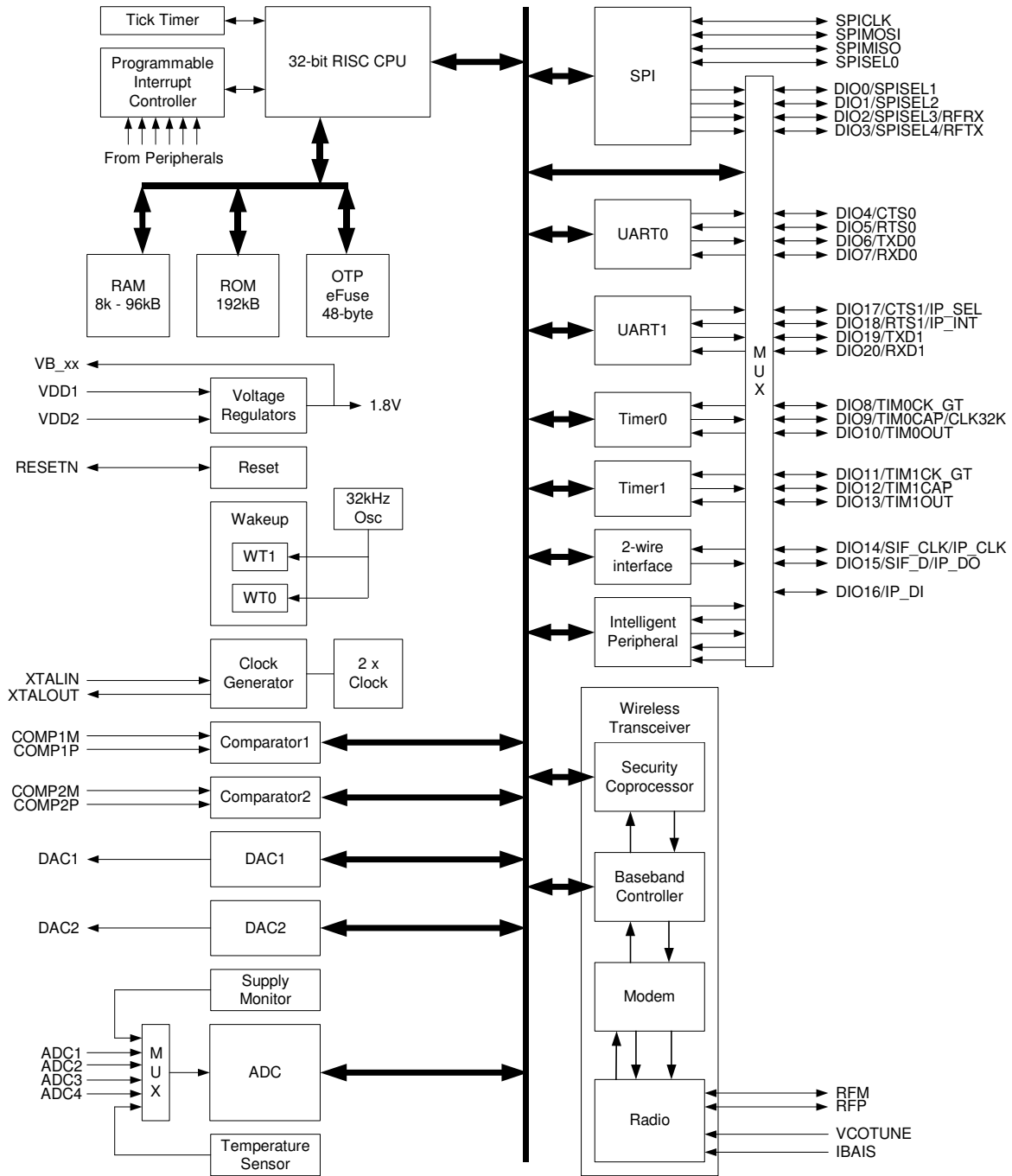


Figure 1: JN513x Block Diagram

2 Pin Configurations

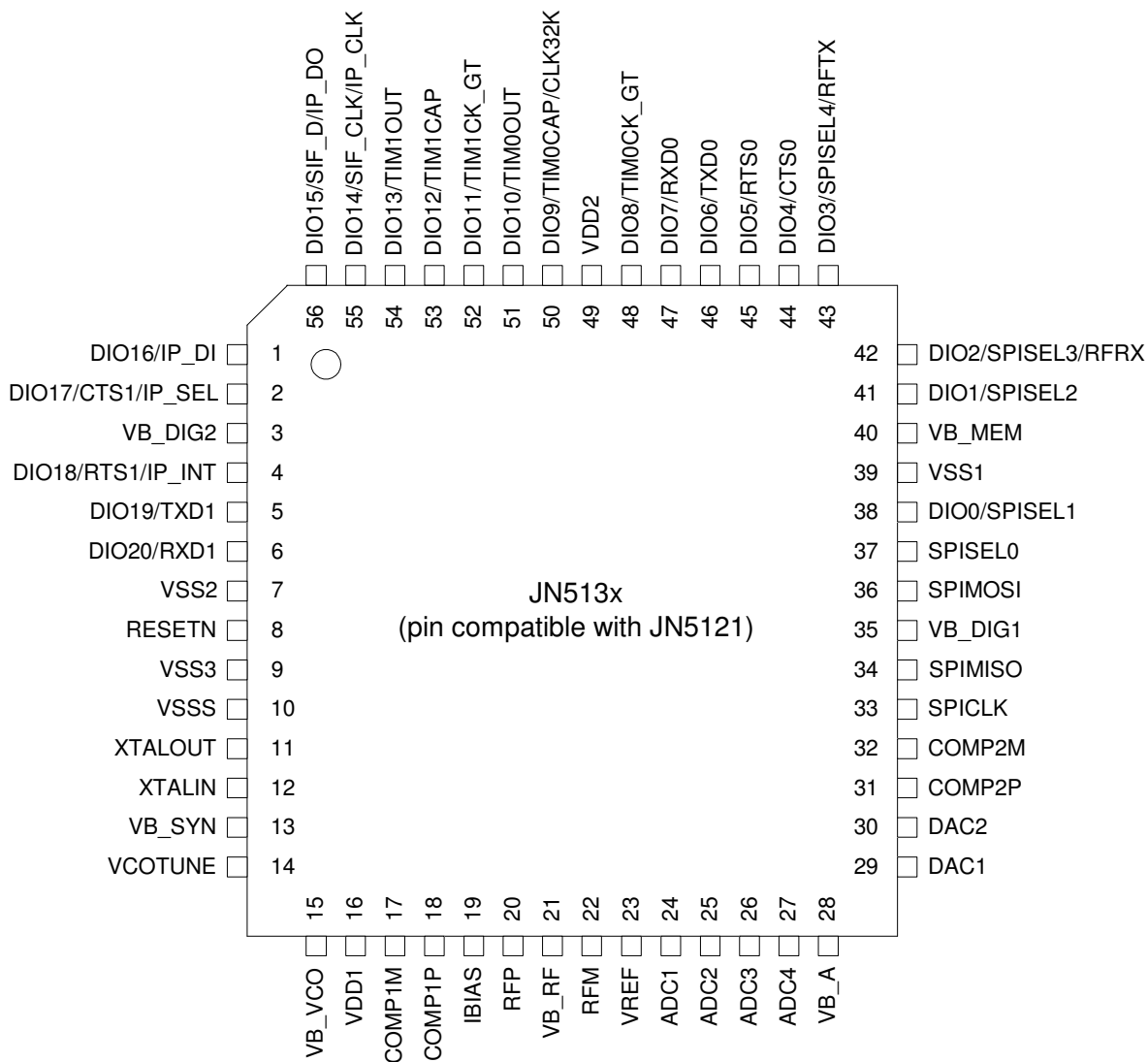


Figure 2: 56-pin QFN Configuration (top view)

2.1 Pin Assignment

Pin No	Power supplies		Description
3, 13, 15, 21 28, 35, 40	VB_DIG2, VB_SYN, VB_VCO, VB_RF, VB_A, VB_DIG1, VB_MEM		Regulated supply voltage
16, 49	VDD1, VDD2		Device supplies: VDD1 for analog, VDD2 for digital
7, 9, 10, 39 Paddle	VSS2, VSS3, VSSS, VSS1, VSSA		Device grounds (see appendix A.1 for paddle details)
General			
8	RESETN		Reset input
11, 12	XTALOUT, XTALIN		System crystal oscillator
Radio			
14	VCOTUNE		VCO tuning RC network
19	IBIAS		Bias current control
20, 22	RFP, RFM		Differential antenna port
Analogue Peripheral I/O			
24, 25, 26, 27	ADC1, ADC2, ADC3, ADC4		ADC inputs
23	VREF		Analogue peripheral reference voltage
29, 30	DAC1, DAC2		DAC outputs
17, 18, 31, 32	COMP1M, COMP1P, COMP2P, COMP2M		Comparator inputs
Digital I/O			
	Primary Function	Alternate Function	
33	SPICLK		SPI Clock
36	SPIMOSI		SPI Master Out Slave In
34	SPIMISO		SPI Master In Slave Out
37	SPISEL0		SPI Slave Select Input/Output 0
38	DIO0	SPISEL1	DIO0 or SPI Slave Select Output 1
41	DIO1	SPISEL2	DIO1 or SPI Slave Select Output 2
42	DIO2	SPISEL3, RFRX	DIO2 or SPI Slave Select Output 3 or Radio Receive Control Output
43	DIO3	SPISEL4, RFTX	DIO3 or SPI Slave Select Output 4 or Radio Transmit Control Output
44	DIO4	CTS0	DIO4 or UART 0 Clear To Send Input
45	DIO5	RTS0	DIO5 or UART 0 Request To Send Output
46	DIO6	TXD0	DIO6 or UART 0 Transmit Data Output
47	DIO7	RXD0	DIO7 or UART 0 Receive Data Input
48	DIO8	TIM0CK_GT	DIO8 or Timer0 Clock/Gate Input
50	DIO9	TIM0CAP, CLK32K	DIO9 or Timer0 Capture Input or CLK32K
51	DIO10	TIM0OUT	DIO10 or Timer0 PWM Output
52	DIO11	TIM1CK_GT	DIO11 or Timer1 Clock/Gate Input
53	DIO12	TIM1CAP	DIO12 or Timer1 Capture Input or Antenna Diversity
54	DIO13	TIM1OUT	DIO13 or Timer1 PWM Output or Antenna Diversity
55	DIO14	SIF_CLK, IP_CLK	DIO14 or Serial Interface Clock or Intelligent Peripheral Clock Input
56	DIO15	SIF_D, IP_DO	DIO15 or Serial Interface Data or Intelligent Peripheral Data Out
1	DIO16	IP_DI	DIO16 or Intelligent Peripheral Data In
2	DIO17	CTS1, IP_SEL	DIO17 or UART 1 Clear To Send Input or Intelligent Peripheral Device Select Input
4	DIO18	RTS1, IP_INT	DIO18 or UART 1 Request To Send Output or Intelligent Peripheral Interrupt Output
5	DIO19	TXD1	DIO19 or UART 1 Transmit Data Output
6	DIO20	RXD1	DIO20 or UART 1 Receive Data Input

2.2 Pin Descriptions

2.2.1 Power Supplies

The device is powered from the VDD1 and VDD2 pins, each being decoupled with a 100nF ceramic capacitor. VDD1 is the power supply to the analogue circuitry; it should be decoupled to analogue ground. VDD2 is the power supply for the digital circuitry; it should be decoupled to digital ground. A 10uF tantalum capacitor is required at the common ground star point of analogue and digital supplies. Decoupling pins for the internal 1.8V regulators are provided which require a 100nF capacitor located as close to the device as practical. VB_VCO, VB_RF, VB_A and VB_SYN should be decoupled to analogue ground, while VB_MEM, VB_DIG1 and VB_DIG2 should be decoupled to digital ground. See also Appendix B for connection details.

VSSA is the analogue ground, connected to the paddle of the device, while VSSS, VSS1, VSS2, VSS3 are digital ground pins.

2.2.2 Reset

RESETN is a bi-directional active low reset pin that is connected to a 45kΩ internal pull-up resistor. It may be pulled low by an external circuit, or can be driven low by the JN513x if an internal reset is generated. Typically, it will be used to provide a system reset signal. Refer to section 6.2, External Reset, for more details.

2.2.3 16MHz System Clock

A crystal connected between XTALIN and XTALOUT drives the system clock. A capacitor to analogue ground is required on each of these pins. Refer to section 5.1 16MHz Oscillator for more details.

2.2.4 Radio

A 200Ω balanced antenna (such as a printed circuit antenna) can be connected directly to the radio interface pins RFM and RFP.

A single-ended 50Ω antenna such as a ceramic type or SMA connector for an external antenna requires the addition of a 200/50Ω 2.45GHz balun transformer connected to the antenna pins. The balun differential port should be connected to the antenna port with 200Ω balanced controlled impedance track. A 50Ω controlled impedance track should be used to connect the unbalanced port of the balun to the antenna to ensure good impedance matching and reduce losses and reflections.

A simple external loop filter circuit consisting of two capacitors and a resistor is connected to VCOTUNE. Refer to section 8.1 Radio for more details.

An external resistor (43kΩ) is required between IBIAS and analogue ground to set various bias currents and references within the radio.

2.2.5 Analogue Peripherals

Several of the analogue peripherals require a reference voltage to use as part of their operations. They can use either an internal reference voltage or an external reference connected to VREF. This voltage is referenced to analogue ground and the performance of the analogue peripherals is dependant on the quality of this reference.

There are four ADC inputs, two comparator inputs and two DAC outputs. The analogue I/O pins on the JN513x can have signals applied up to 0.3v higher than VDD1. A schematic view of the analogue I/O cell is shown in Figure 3 Analogue I/O Cell.

In reset and deep sleep the analogue peripherals are all off and the DAC outputs are in a high impedance state. During sleep the ADC and DAC's are off, with the DAC outputs in a high impedance state and the comparator may optionally be used as a wakeup.

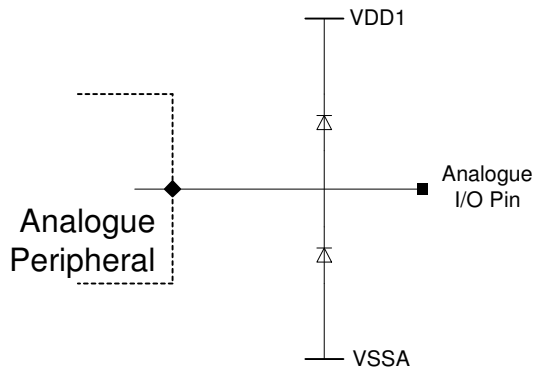


Figure 3 Analogue I/O Cell

2.2.6 Digital Input/Output

Digital I/O pins on the JN513x can have signals applied up to 2V higher than VDD2 and are therefore TTL-compatible with VDD2 > 3V. For other DC properties of these pins see section 17.2.3 I/O Characteristics.

When used in their primary function all Digital Input/Output pins are bi-directional and are connected to weak internal pull up resistors (45kΩ nominal) that can be disabled. When used in their secondary function (selected when the appropriate peripheral block is enabled) their direction is fixed by the function.

A schematic view of the digital I/O cell is in Figure 4: DIO Pin Equivalent Schematic.

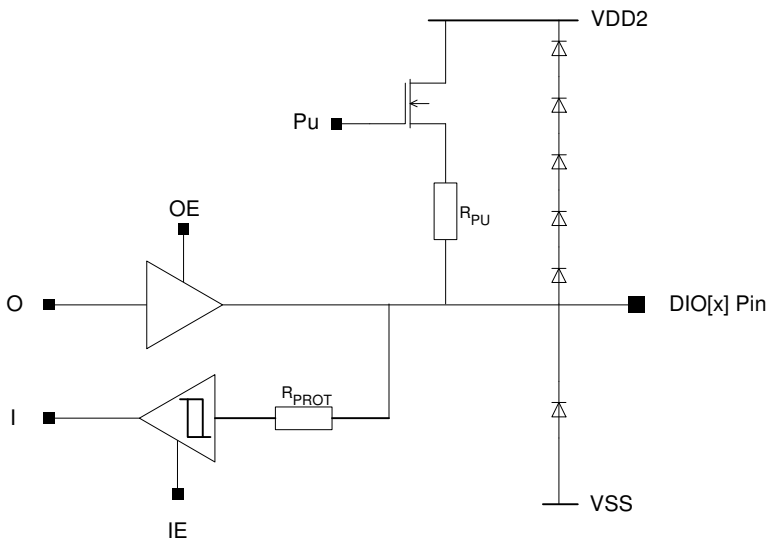


Figure 4: DIO Pin Equivalent Schematic

Each DIO pin configuration is programmed by functions in Hardware Peripheral Library. The pin direction is set by calling the `vAHI_DioSetDirection()` function that enables OE and IE as required, or by enabling a peripheral which uses the cell as part of its I/O. The use of the pull-up resistor R_{pu} for each pin is controlled through the `vAHI_DioSetPullup()` routine in the peripheral library.

In reset, the digital peripherals are all off and the DIO pins are set as high-impedance inputs. During sleep and deep sleep, the DIO pins retain both their input/output state and output level that was set as sleep commences. If the DIO pins were enabled as inputs and the interrupts were enabled these pins may be used to wake up the JN513x from sleep.

3 CPU

The CPU of the JN513x is a 32-bit load and store RISC processor. It has been architected for three key requirements:

- Low power consumption for battery powered applications
- High performance to implement a wireless protocol at the same time as complex applications
- Efficient coding of high-level languages such as C/C++ provided with the Jennic Software Developers Kit

It features a linear 32-bit logical address space with unified memory architecture, accessing both code and data in the same address space. Registers for peripheral units, such as the timers, UARTs and the baseband processor are also mapped into this space.

The CPU contains a block of 32 32-bit General-Purpose (GP) registers together with a small number of special purpose registers which are used to store processor state and control interrupt handling. The contents of any GP register can be loaded from or stored to memory, while arithmetic and logical operations, shift and rotate operations, and signed and unsigned comparisons can be performed either between two registers and stored in a third, or between registers and a constant carried in the instruction. Operations between general or special-purpose registers execute in one cycle (16/32MHz) while those that access memory require a further cycle to allow the memory to respond.

The instruction set manipulates 8, 16 and 32-bit data; this means that programs can use objects of these sizes very efficiently. Manipulation of 32-bit quantities is particularly useful for protocols and high-end applications allowing algorithms to be implemented in fewer instructions than on smaller word-size processors, and to execute in fewer clock cycles. In addition, the CPU supports hardware Multiply that can be used to efficiently implement algorithms needed by Digital Signal Processing applications.

The instruction set is designed for the efficient implementation of high-level languages such as C. Access to fields in complex data structures is very efficient due to the provision of several addressing modes, together with the ability to be able to use any of the GP registers to contain the address of objects. Subroutine parameter passing is also made more efficient by using GP registers rather than pushing objects on the stack. The recommended programming method for the JN513x is by using C, which is supported by a software developer kit comprising a C compiler, linker and debugger.

The CPU architecture also contains features that make the processor suitable for embedded, real-time applications. In some applications, it may be necessary to use a real-time operating system to allow multiple tasks to run on the processor. To provide protection for device-wide resources being altered by one task and affecting another, the processor can run in either supervisor or user mode, the former allowing access to all processor registers, while the latter only allows the GP registers to be manipulated. Supervisor mode is entered on reset or interrupt; tasks starting up would normally run in user mode in a RTOS environment.

Embedded applications require efficient handling of external hardware events. Exception processing (including reset and interrupt handling) is enhanced by the inclusion of a number of special-purpose registers into which the PC and status register contents are copied as part of the operation of the exception hardware. This means that the essential registers for exception handling are stored in one cycle, rather than the slower method of pushing them onto the processor stack. The PC is also loaded with the vector address for the exception that occurred, allowing the handler to start executing in the next cycle.

To improve power consumption a number of power-saving modes are implemented in the JN513x, described more fully in section 16 - Power Management and Sleep Modes. One of these modes is the CPU doze mode, under software control, the processor can be shut down and on an interrupt it will wake up to service the request.

The CPU clock may be optionally doubled using a 2x clock input. Using the 2x clock mode enables the CPU to clocked at 32MHz and therefore able to sustain 32 MIPs.

4 Memory Organisation

This section describes the different memories found within the JN513x. The device contains ROM, RAM, OTP memory, the wireless transceiver and peripherals all within the same linear address space.

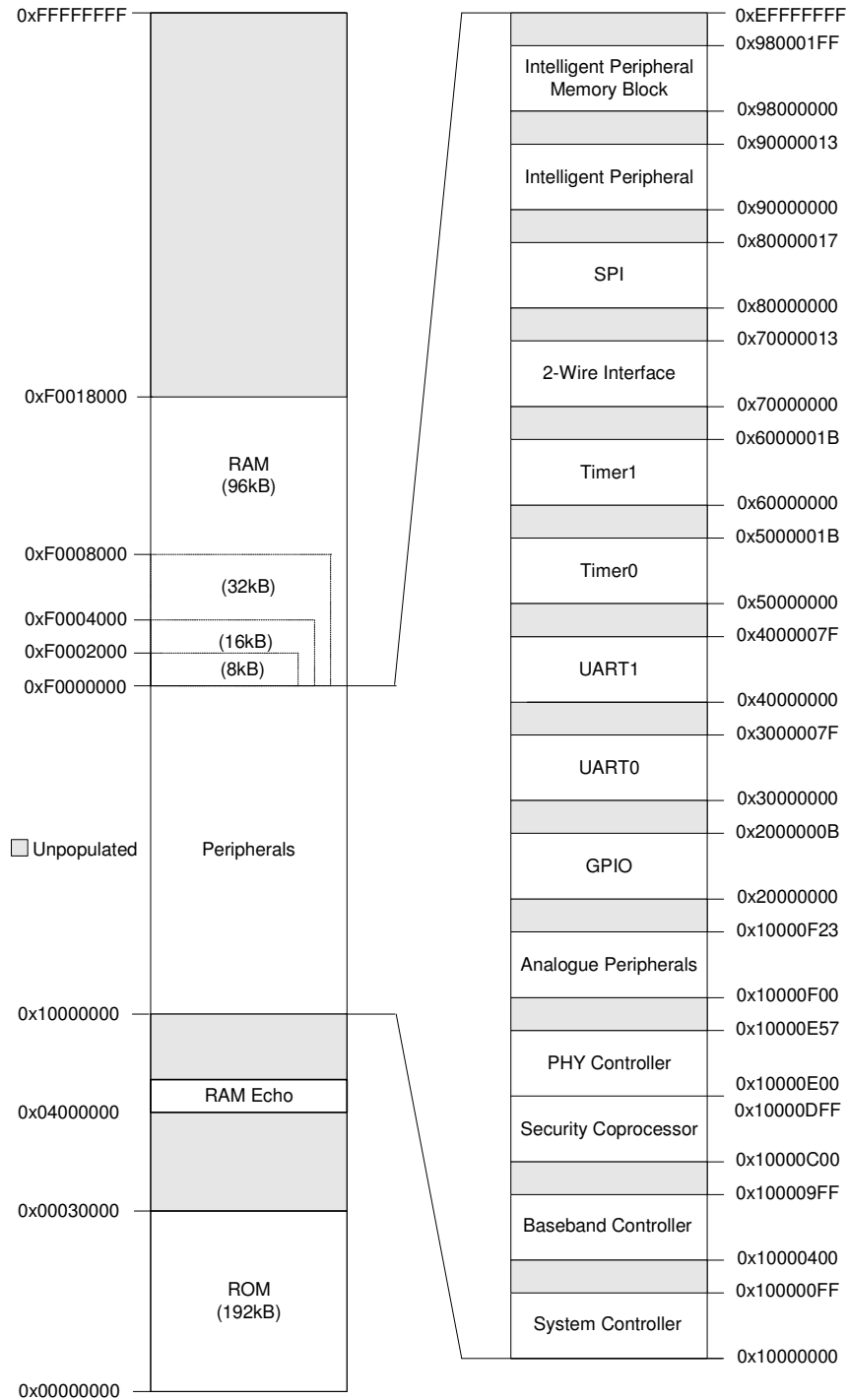


Figure 5: JN513x Memory Map

4.1 ROM

The ROM is 192K bytes in size, organized as 48k x 32-bit words and can be accessed by the CPU in a single clock cycle. The ROM contents change for different versions of the device to support differing protocol stacks and applications, all versions carry a default interrupt vector table and interrupt manager. Variants that can be used for application or protocol development carry a boot loader, to allow code from external Flash memory to be bootloaded into RAM at runtime. The operation of the boot loader is described in detail in Application Note JN-AN-1003 Boot Loader Operation [4]. For development variants the interrupt manager routes interrupt calls to the application's soft interrupt vector table contained within RAM. Section 7 contains further information regarding the handling of interrupts. Typical ROM contents for a development variant containing a ZigBee protocol stack is shown in Figure 6.

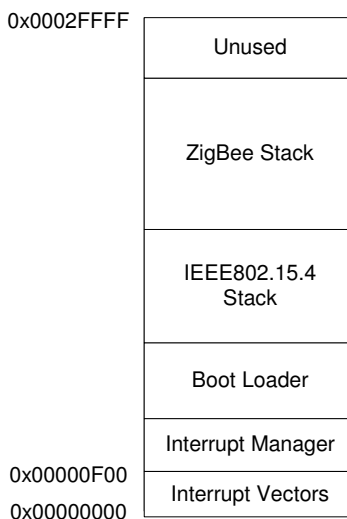


Figure 6: Typical ROM contents

4.2 RAM

The JN513x contains 8k, 16k, 32k or 96k bytes of high speed RAM organized as 2k, 4k, 8k or 24k x 32-bit words respectively. It can be used for both code and data storage and is accessed by the CPU in a single clock cycle. At reset, a boot loader controls the loading of segments of code and data from an external memory connected to the SPI port, into RAM. Software can control the power supply to the RAM allowing the contents to be maintained during a sleep period when other parts of the device are un-powered.

4.3 OTP eFuse Memory

The JN513x contains 48-bytes of eFuse memory; this is one time programmable memory that is organised as 12 x 32-bit words, 4 words are reserved by Jennic, 2 of which support on-chip MAC ID. The remaining 8 words are fully user programmable, designed to allow the storage of configuration and product information. If secure external memory encryption is enabled then 4 words of the user eFuse are used for this (see section 4.4.1)

At a low level, programming of the eFuse requires for a sequence of carefully controlled steps, therefore to simplify the procedure, a simple API function call through software is provided that handles the various sequences required, this is described in JN-RM-2001 Integrated Peripherals API [2]. For reliable programming operation, a minimum system supply voltage VDD2 of 3.3V must be provided. If this condition is not satisfied, then programming reliability cannot be guaranteed.

4.4 External Memory

An external memory with an SPI interface may be used to provide storage for program code and data for the device when external power is removed. The memory is connected to the SPI interface using select line SPISEL0; this select line is dedicated to the external memory interface and is not available for use with other external devices. See Figure 7 for connection details.

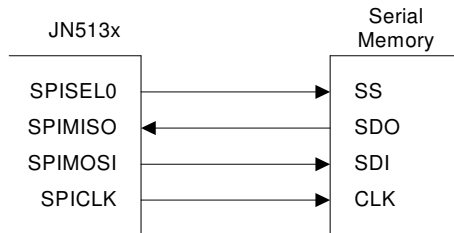


Figure 7: Connecting External Serial Memory

At reset, the contents of this memory are copied into RAM by the software boot loader. A number of types of memory device may be used with the JN513x boot loader so long as they conform to the format of read instructions issued by the boot loader over the SPI interface. See application note [4] JN-AN-1003 Boot Loader Operation for details on the format of the read command and other details of the boot loader.

4.4.1 Secure External Memory Encryption

The contents of the external serial memory may be securely encrypted to protect against system cloning or intrusion. The AES security processor combined with a user programmable 128-bit encryption key is used to encrypt the contents of the external memory. The encryption key is stored in eFuse and is programmed through software control. Initially after programming, the encryption feature is not active; this allows the system to continue to operate in an unsecured mode. Enabling of the encryption feature is through software control, once enabled all programming operations require authentication. Full details of the eFuse software functions may be found in JN-RM-2001 Integrated Peripherals API [2].

When bootloading program code from external serial memory, the JN513x automatically accesses the encryption key to execute the decryption process. User program code, does not need to handle any of the decryption process, it is a transparent process.

4.5 Peripherals

All peripherals have their registers mapped into the memory space. Access to these registers requires 3 clock cycles. Applications have access to the peripherals through the peripherals library, which presents a high-level view of the peripheral's functions through a series of dedicated software routines. These routines provide both a tested method for using the peripherals and operation of power and interrupts with the IEEE802.15.4 software protocol stack allowing bug-free application code to be developed more rapidly. See JN-RM-2001 Integrated Peripherals API [2] for more details.

4.6 Unused Memory Addresses

Any attempt to access an unpopulated memory area will result in a bus error exception (interrupt) being generated.

5 System Clocks

Two separate oscillators are used to provide system clocks: a crystal controlled 16MHz oscillator, using an external crystal and an internal, RC based 32kHz oscillator.

5.1 16MHz Oscillator

The JN513x contains the necessary on-chip components to build a 16 MHz reference oscillator with the addition of an external crystal resonator and two tuning capacitors. The schematic and layout of these components are shown in Figure 8. The two capacitors, C1 and C2, should be 15pF \pm 5% and use a COG dielectric. Due to the small size of these capacitors, it is important to keep the traces to the external components as short as possible. The on-chip transconductance amplifier is compensated for temperature variation, and is self-biasing by means of the internal resistor R1. The electrical specification of the oscillator can be found in section 17.3.6. For detailed application support and specification of the crystal required see Appendix B.1.

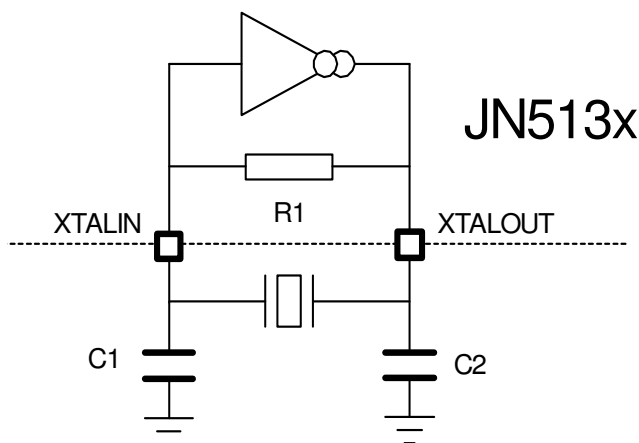


Figure 8: Crystal oscillator connections

The clock generated by this oscillator provides the reference for most of the JN513x subsystems, including the transceiver, processor, memory and digital and analogue peripherals. The clock for the processor, RAM and ROM may be optionally driven by a 2x clock that effectively clocks these at 32MHz.

5.2 32kHz Oscillator

The internal 32kHz RC oscillator requires no external components. It provides a low speed clock for use in sleep mode. The clock is used for timing the length of a sleep period (see section 16 Power Management and Sleep Modes) and also to generate the system clock used internally during reset. The internal timing components of the oscillator have a wide tolerance due to manufacturing process variation and so the oscillator runs nominally at 32kHz \pm 30%. To make this useful as a timing source for accurate wakeup from sleep, a frequency calibration factor derived from the more accurate 16MHz oscillator may be applied. The calibration factor is derived through software, details can be found in section 12.3.1. For detailed electrical specifications, see section 17.3.5.

6 Reset

A system reset initialises the device to a predefined state and forces the CPU to start program execution from the reset vector. The reset process that the JN513x goes through is as follows.

When power is applied, the 32kHz oscillator starts up and stabilises, which takes approximately 100µsec. At this point, the 16MHz crystal oscillator is enabled and power is applied to processor and peripheral logic. The logic blocks are held in reset until the 16MHz crystal oscillator stabilises, typically this takes 2.5ms.

Once the oscillator is up and running the internal reset is removed from the CPU and peripheral logic and the CPU starts to run code beginning at the reset vector, consisting of initialisation code and then optionally the resident Boot Loader (described in reference [4]).

Section 17.3.1 provides detailed electrical data and timing. Appendix B describes the JN513x pin states during and after reset.

The JN513x has four sources of reset:

- Power-on Reset
- External Reset
- Software Reset
- Brown-Out-Detect

Note: When the device exits a reset condition, device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, then the device must be held in reset until the operating conditions are met.

6.1 Power-on Reset

A power-on reset is generated by an on-chip detection circuit eliminating the need for an external reset circuit. The power-on reset is activated whenever VDD is below the detection level, and causes the JN513x to be held in reset. Once VDD has risen above this level, and the power supply and oscillator stabilization time t_{STAB} has elapsed, the reset is removed and the CPU is allowed to run. During the time that the internal reset is active the RESETN pin is driven low to provide a reset signal to any other devices in the system.

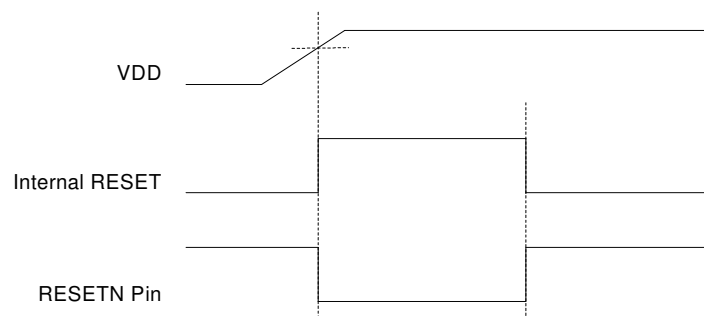


Figure 9: Power-on Reset

6.2 External Reset

An external reset is generated by a low level on the RESETN pin. Reset pulses longer than the minimum pulse width will generate a reset during active or sleep modes. Shorter pulses are not guaranteed to generate a reset. The JN513x is held in reset while the RESETN pin is low and when the applied signal reaches the Reset Threshold Voltage (V_{RST}) on its positive edge, the internal reset process starts.

Multiple devices may connect to the RESETN pin in an open-collector mode. The JN513x has an internal pull-up resistor although an external pull-up resistor is recommended when multiple devices connect to the RESETN pin. The pin is an input for an external reset, an output during the power-on reset and may optionally be an output during a software reset. No devices should drive the RESETN pin high.

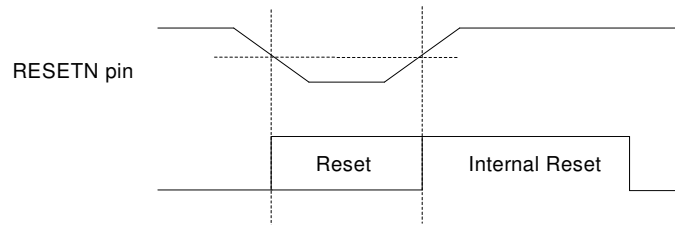


Figure 10: External Reset

6.3 Software Reset

A system reset can be triggered at any time by calling the Software Reset function, `vAHI_SwReset()` from the peripheral library. This function can be executed within a users application, upon detection of a system failure for example. The RESETN line can be driven low by the JN513x to provide a reset to other devices in the system (e.g. external sensors). The reset output feature can be enabled or disabled for the software generated reset using the function `vAHI_DriveResetOut()` within the peripheral library (the default state is disabled).

6.4 Brown-out Detect

A brown-out detect module is used to monitor the supply voltage to the JN513x; this can be used whilst the device is awake or is in CPU doze mode. Dips in the supply voltage below a variable threshold can be detected and can be used to cause the JN513x to perform a chip reset. Equally, dips in the supply voltage can be detected and used to cause an interrupt to the processor, when the voltage either drops below the threshold or rises above it. Hysteresis is built into the brown out detect module this is nominally 100mV.

The threshold voltage is selectable at levels of 2.1V, 2.4V, 2.5V or 2.6V through software control, this is described in JN-RM-2001 Integrated Peripherals API [2].

7 Interrupt System

The interrupt system on JN513x is a hardware-vectorized interrupt system. The JN513x provides several interrupt sources, some associated with CPU operations (CPU exceptions) and others which are used by hardware in the device. When an interrupt occurs the CPU stops executing the current program and loads its program counter with a fixed hardware address specific to that interrupt. The interrupt handler or interrupt service routine is stored at this location and is run on the next CPU cycle. Execution of interrupt service routines is always performed in supervisor mode. Interrupt sources and their vector locations are listed in Table 1 below:

Interrupt Source	Vector Location	Interrupt Definition
Reset	0x100	Software or hardware reset
Bus Error	0x200	Bus error or attempt to access invalid physical address
Tick Timer	0x500	Tick Timer expiry
Alignment	0x600	Load/Store to naturally not aligned location
Illegal Instruction	0x700	Illegal instruction in instruction stream
Hardware Interrupts	0x800	Hardware Interrupt
System Call	0xC00	System Call Initiated by software (l.sys instruction)
Trap	0xE00	Caused by l.trap instruction

Table 1: Interrupt Vectors

7.1 System Calls

Executing the `l.sys` instruction causes a system call interrupt to be generated. The purpose of this interrupt is to allow a task to switch into supervisor mode when a real time operating system is in use, see section 3 for further details. It also allows a software interrupt to be issued, as does execution of the `l.trap` instruction.

7.2 Processor Exceptions

7.2.1 Bus Error

A bus error exception is generated when software attempts to access a memory address that does not exist, or is not populated with memory or peripheral registers.

7.2.2 Alignment

Alignment exceptions are generated when software attempts to access objects that are not aligned to natural word boundaries. 16-bit objects must be stored on even byte boundaries, while 32-bit objects must be stored on quad byte boundaries. For instance, attempting to read a 16-bit object from address 0xFFF1 will trigger an alignment exception as will a read of a 32-bit object from 0xFFF1, 0xFFF2 or 0xFFF3. Examples of legal 32-bit object addresses are 0xFFF0, 0xFFF4, 0xFFF8 etc.

7.2.3 Illegal Instruction

If the CPU reads an unrecognised instruction from memory as part of its instruction fetch, it will cause an illegal instruction exception.

7.3 Hardware Interrupts

Hardware interrupts generated from the transceiver, analogue or digital peripherals and DIO pins are individually masked using the Programmable Interrupt Controller (PIC). Management of interrupts is provided in the peripherals library. Further details of interrupts are provided for the functions in their respective sections in this datasheet.

Interrupts are used to wake the JN513x from sleep. The peripherals, baseband controller, security coprocessor and PIC are powered down during sleep but the DIO interrupts and optionally the wake-up timers and analogue comparator interrupts remain powered to bring the JN513x out of sleep.

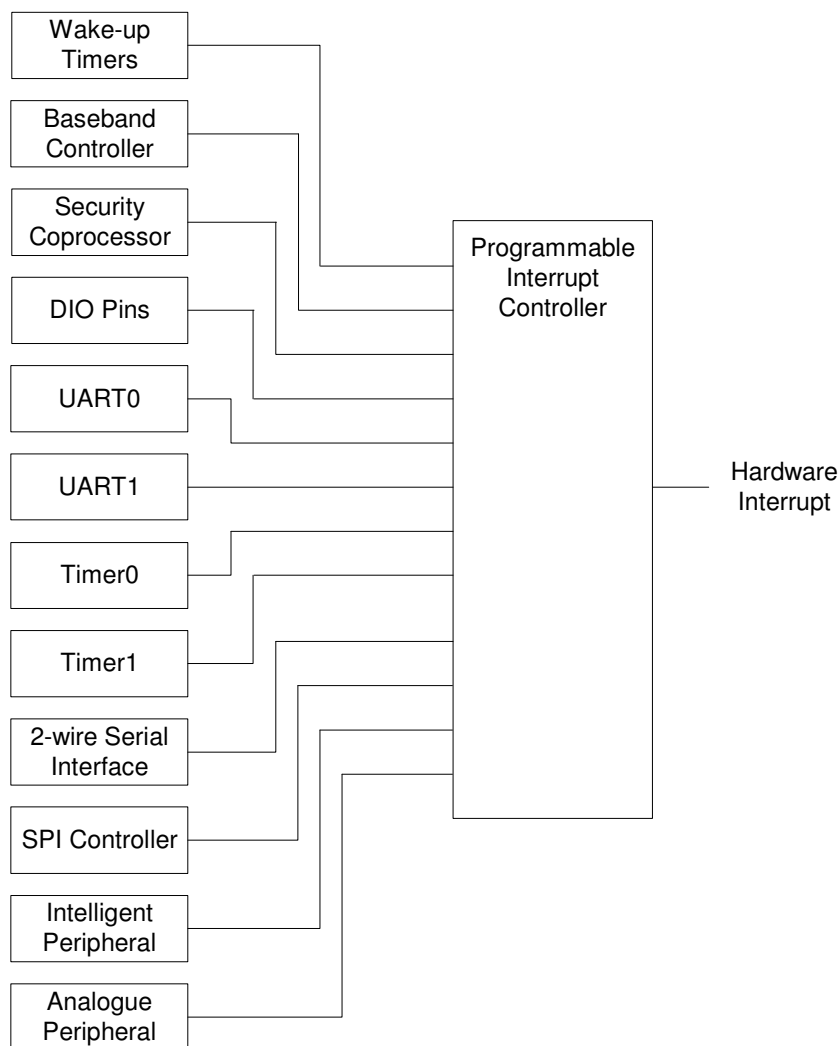


Figure 11: Programmable Interrupt Controller

8 Wireless Transceiver

The wireless transceiver comprises a 2.45GHz radio, an O-QPSK modem, a baseband processor, a security coprocessor and PHY controller. These blocks, with protocol software provided as a library, implement an IEEE802.15.4 standards-based wireless transceiver that transmits and receives data over the air in the unlicensed 2.4GHz band. IEEE802.15.4 wireless functionality is provided with the transceiver and the protocol software described in JN-RM-2002 802.15.4 Stack [3]. Applications interface to the protocol software via an API interface that corresponds to the SAP interfaces defined in the IEEE Std 802.15.4-2006 [1]

8.1 Radio

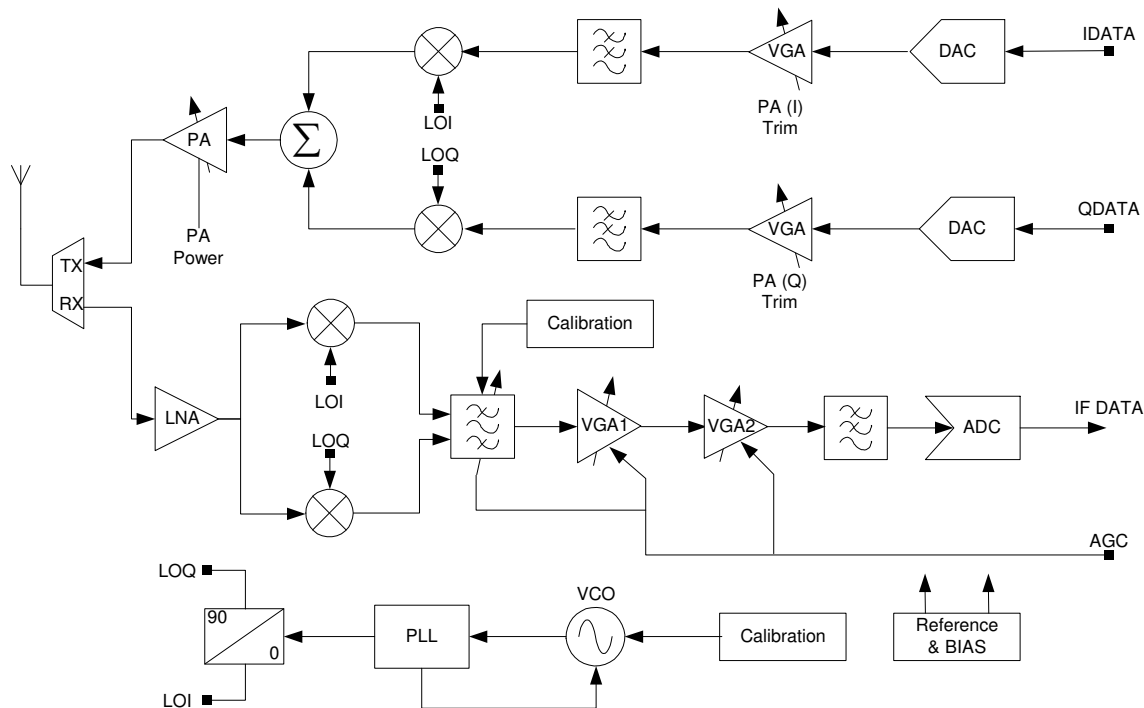


Figure 12: Radio Architecture

The radio comprises a low-IF receive path and a direct up-conversion transmit path, which converge at the TX/RX switch. This switch includes the necessary matching components such that a 200Ω differential antenna may be directly connected without external components. Alternatively, a balun can be used for single ended antennas.

The 16MHz crystal oscillator feeds a divider, which provides the frequency synthesiser with a reference frequency. The synthesiser contains programmable feedback dividers, phase detector, charge pump and internal Voltage Controlled Oscillator (VCO). The VCO has no external components, and includes calibration circuitry to compensate for differences in internal component values due to process and temperature variations. The VCO is controlled by a Phase Lock Loop (PLL) that has a loop filter comprising 3 external components. A programmable charge pump is also used to tune the loop characteristic. Finally, quadrature (I and Q) local oscillator signals for the mixer drives are derived.

The receiver chain starts with the low noise amplifier / mixer combination whose outputs are passed to the polyphase bandpass filter. This filter provides the channel definition as well as image frequency rejection. The signal is then passed to two variable gain amplifier blocks. The gain control for these stages, and the bandpass filter, is derived in the automatic gain control (AGC) block within the Modem. The signal is conditioned with the anti-alias low pass filter, before being converted to a digital signal with a flash ADC.

In the transmit direction, the digital I and Q streams from the Modem are passed to I and Q quadrature DAC blocks which are buffered and low-pass filtered, before being applied to the modulator mixers. The summed 2.4 GHz signal is then passed to the RF Power Amplifier (PA), whose power control can be selected from one of six settings. The output of the PA drives the antenna via the RX/TX switch.

8.1.1 Radio External components

The VCO loop filter requires three external components and the IBIAS pin requires one external component as shown in Figure 13. These components should be placed close to the JN513x pins and analogue ground.

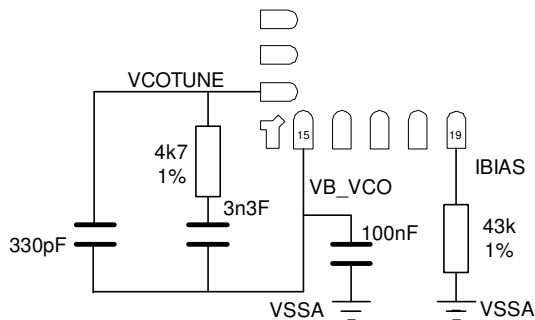


Figure 13: VCO Loop Filter and IBIAS

The radio is powered from a number of internal 1.8V regulators fed from the analogue supply VDD1, in order to provide good noise isolation between the digital logic of the JN513x and the analogue blocks. These regulators are also controlled by the baseband controller and protocol software to minimise power consumption. Decoupling for internal regulators is required as described in section 2.2.1, Power Supplies.

8.1.2 Antenna Diversity

Support is provided for antenna diversity. Antenna diversity is a technique that maximises the performance of an antenna system. It allows the radio to switch between two antennas that have very low correlation between their received signals. Typically, this is achieved by spacing two antennas around 0.25 wavelengths apart or by using two orthogonal polarisations. So, if a packet is transmitted and no acknowledgement is received, the radio system can switch to the other antenna for the retry, with a different probability of success.

The JN513x provides two outputs that can be used to control an antenna switch; this enables antenna diversity to be implemented easily. DIO12 is asserted on odd numbered retries and DIO13 is asserted on the first transmit and even numbered retries.

8.2 Modem

The Modem performs all the necessary modulation and spreading functions required for digital transmission and reception of data at 250kbps in the 2450MHz radio frequency band in compliance with the IEEE802.15.4 standard.

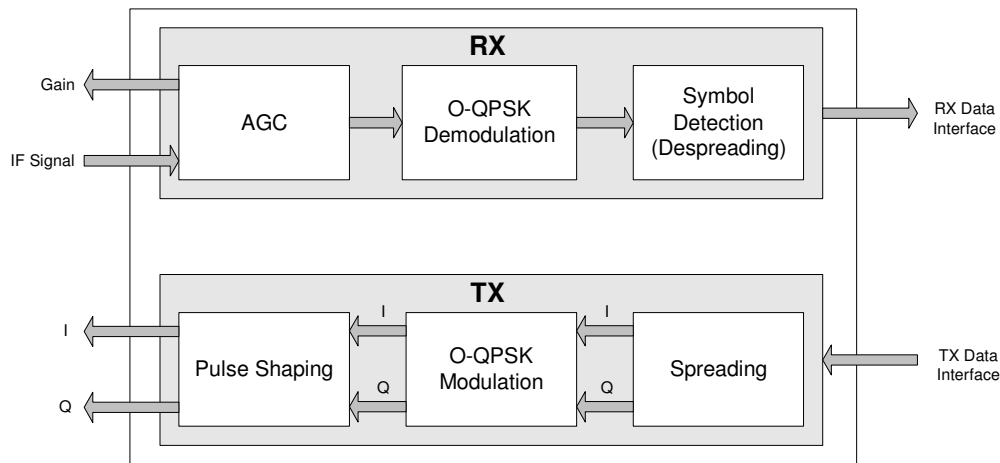


Figure 14: Modem Architecture

The transmitter receives symbols from the baseband processor and uses the spreading function to map each unique 4-bit symbol to a 32-chip Pseudo-random Noise (PN) sequence. Offset-QPSK modulation and half-sine pulse shaping is applied to the resultant spreading sequence to produce two independent quadrature phase signals (I and Q), which are subsequently converted to analogue voltages in the radio transmit path.

The Automatic Gain Control (AGC) monitors the received signal level and adjusts the gain of the amplifiers in the radio receiver to ensure that the optimum signal amplitude is maintained during reception.

The demodulator performs digital IF down-conversion and matched filtering and is extremely tolerant to carrier frequency offsets in excess of ± 80 ppm without suffering any significant degradation in performance.

Symbol detection and synchronization is performed using direct sequence correlation techniques in conjunction with searches for the Preamble and Start-of-Frame Delimiter (SFD) fields contained in the transmitted IEEE 802.15.4 Synchronization Header (SHR).

Features are provided to support network channel selection algorithms include Energy Detection (ED), Link Quality Indication (LQI) and fully programmable Clear Channel Assessment (CCA).

The Modem provides a digital Receive Signal Strength Indication (RSSI) that facilitates the implementation of the IEEE 802.15.4 ED function.

The LQI is defined in the IEEE 802.15.4 standard as a characterization of the strength and/or data quality of a received packet. The Modem produces a signal quality metric based upon correlation magnitudes, which may be used in conjunction with the ED value to formulate the LQI.

The CCA capability of the Modem supports all modes of operation defined in the IEEE 802.15.4 standard, namely Energy above ED threshold, Carrier Sense and Carrier Sense and/or energy above ED threshold.

8.3 Baseband Processor

The baseband processor provides all time-critical functions of the IEEE802.15.4 MAC layer. Dedicated hardware guarantees air interface timing is precise. The MAC layer hardware/software partitioning, enables software to implement the sequencing of events required by the protocol and to schedule timed events with millisecond resolution, and the hardware to implement specific events with microsecond timing resolution. The protocol software layer performs the higher-layer aspects of the protocol, sending management and data messages between endpoint and coordinator nodes, using the services provided by the baseband processor.

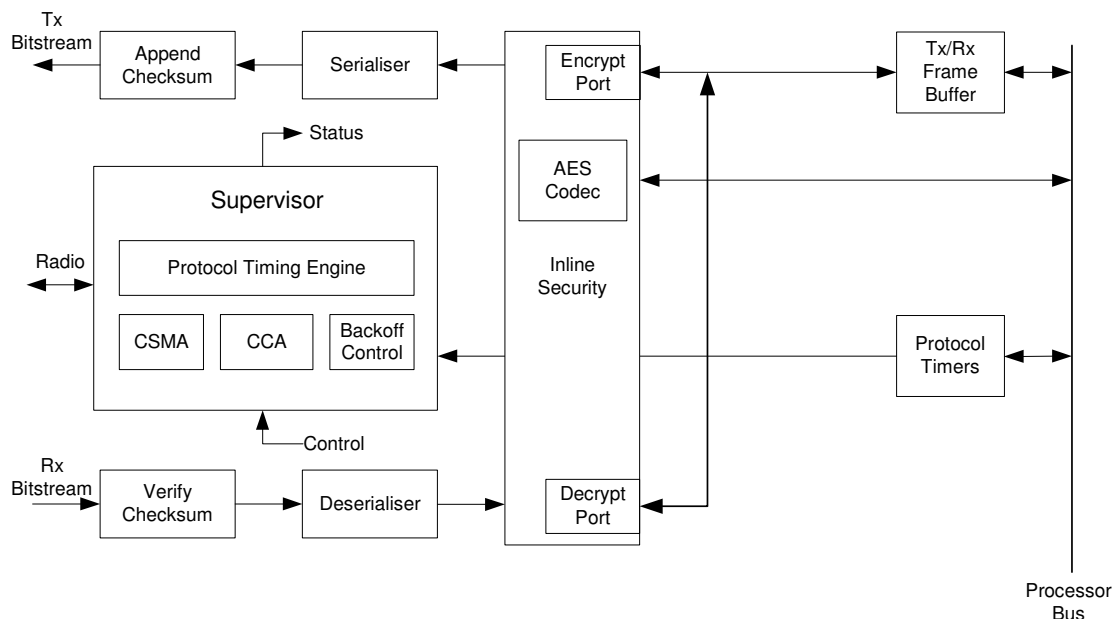


Figure 15: Baseband Processor

8.3.1 Transmit

A transmission is performed by software writing the data to be transferred into the Tx/Rx Frame Buffer, together with parameters such as the destination address and the number of retries allowed, and programming one of the protocol timers to indicate the time at which the frame is to be sent. This time will be determined by the software tracking the higher-layer aspects of the protocol such as superframe timing and slot boundaries. Once the packet is prepared and protocol timer set, the supervisor block controls the transmission. When the scheduled time arrives, the supervisor controls the sequencing of the radio and modem to perform the type of transmission required. It can perform all the algorithms required by IEEE802.15.4 such as CSMA/CA, GTS without processor intervention including retries and random backoffs.

When the transmission begins, the header of the frame is constructed from the parameters programmed by the software and sent with the frame data through the serialiser to the Modem. At the same time, the radio is prepared for transmission. During the passage of the bitstream to the modem, it passes through a CRC checksum generator that calculates the checksum on-the-fly, and appends it to the end of the frame.

If using slotted access, it is possible for a transmission to overrun the time in its allocated slot; the Baseband Processor handles this situation autonomously and notifies the protocol software via interrupt, rather than requiring it to handle the overrun explicitly.

8.3.2 Reception

In a reception, the radio is set to receive on a particular channel. On receipt of data from the modem, the frame is directed into the Tx/Rx Frame Buffer where both header and frame data can be read by the protocol software. An interrupt may be provided on receipt of the frame header. As the frame data is being received from the modem it is passed through a checksum generator; at the end of the reception the checksum result is compared with the checksum at the end of the message to ensure that the data has been received correctly.

During reception, the modem determines the Link Quality, which is made available at the end of the reception as part of the requirements of IEEE802.15.4.

8.3.3 Auto Acknowledge

Part of the protocol allows for transmitted frames to be acknowledged by the destination sending an acknowledge packet within a very short window after the transmitted frame has been received. The JN513x baseband processor can automatically construct and send the acknowledgement packet without processor intervention and hence avoid the protocol software being involved in time-critical processing within the acknowledge sequence. The JN513x baseband processor can also request an acknowledge for packets being transmitted and handle the reception of acknowledged packets without processor intervention.

8.3.4 Beacon Generation

In beaconing networks, the baseband processor can automatically generate and send beacon frames; the repetition rate of the beacons is programmed by the CPU, and the baseband then constructs the beacon contents from data delivered by the CPU. The baseband processor schedules the beacons and transmits them without CPU intervention.

8.3.5 Security

The baseband processor supports the transmission and reception of secured frames using the Advanced Encryption Standard (AES) algorithm transparently to the CPU. This is done by passing incoming and outgoing data through an in-line security engine that is able to perform encryption and decryption operations on-the-fly, resulting in minimal processor intervention. The CPU must provide the appropriate encrypt/decrypt keys for the transmission or reception. On transmission, the key can be programmed at the same time as the rest of the frame data and setup information.

During reception, the CPU must look up the key and provide it from information held in the header of the incoming frame. However, the hardware of the security engine can process data much faster than the incoming frame data rate. This means that it is possible to allow the CPU to receive the interrupt from the header of an incoming packet, read where the frame originated, look up the key and program it to the security hardware before the end of the frame has arrived. By providing a small amount of buffering to store incoming data while the lookup process is taking place, the security engine can catch up processing the frame so that when the frame arrives in the receive frame buffer it is fully decrypted.

8.4 Security Coprocessor

As well as being used during in-line encryption/decryption operations over a streaming interface and in external memory encryption, it is also possible to use the AES core as a coprocessor to the CPU of the JN513x. To allow the hardware to be shared between the two interfaces an arbiter ensures that the streaming interface to the AES core always has priority, to ensure that in-line processing can take place at any time.

Some protocols (for example ZigBee) require that security operations can be performed on buffered data rather than in-line. A hardware implementation of the encryption engine significantly speeds up the processing of the contents of these buffers. The AES library for the JN513x provides two operations `vAHI_SecurityEncode()` and `vAHI_SecurityDecode()` which utilise the encryption engine in the device and allow the contents of memory buffers to be transformed. Information such as the type of security operation to be performed and the encrypt/decrypt key to be used must also be provided.

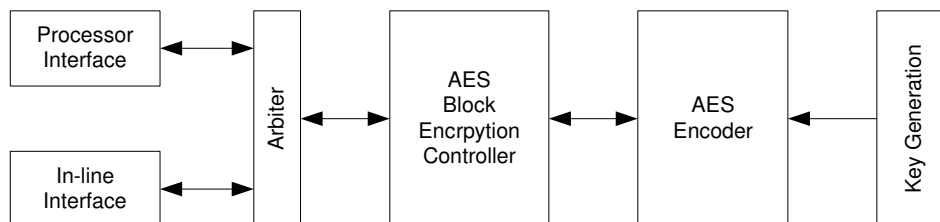


Figure 16: Security Coprocessor Architecture

9 Digital Input/Output

There are 21 Digital I/O (DIO) pins, which can be configured as either an input or an output, and each has a selectable internal pull-up resistor. Most DIO pins are multiplexed with alternate peripheral features of the device. Once a peripheral is enabled it takes precedence over the device pins. Refer to the individual module sections for a full description of the alternate peripherals functions. From reset, all peripherals are off and the DIO pins are configured as inputs with the internal pull-ups turned on.

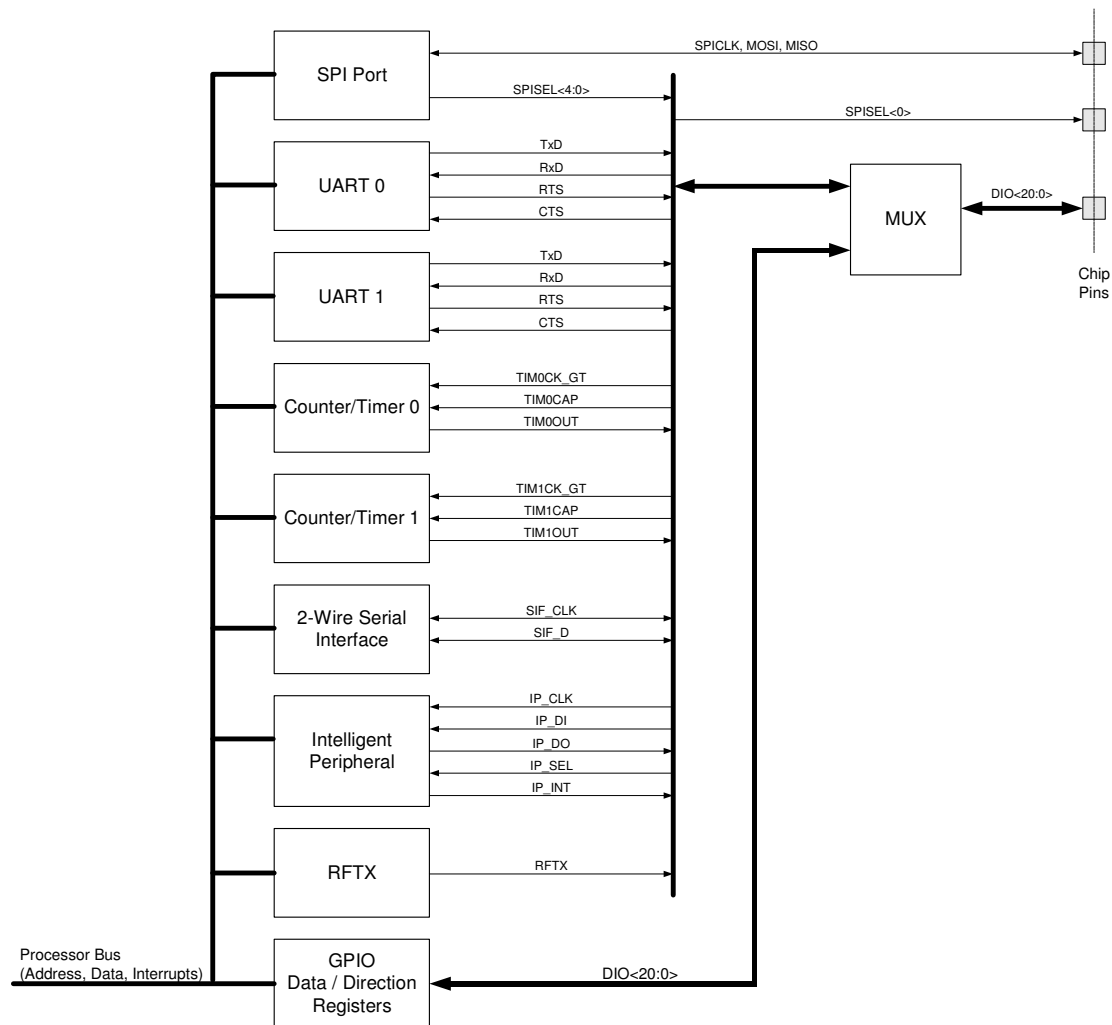


Figure 17: DIO Block Diagram

When a peripheral is not enabled, the DIO pins associated with it can be used as digital inputs or outputs. Each pin can be controlled individually with the direction being set using the `vAHI_DioSetDirection()` function. Reading and writing to the pins is controlled using the `vAHI_DioSetOutput()` and `u32AHI_DioReadInput()` functions.

The individual pull-up resistors, R_{PU} , are selected using the `vAHI_DioSetPullup()` function.

When configured as an input each pin can be used to generate an interrupt upon a change of state (selectable transition either from low to high or high to low); the interrupt can be enabled or disabled. When the device is sleeping, these interrupts become events that can be used to wake the device up. Selection of the interrupt transition is done using `vAHI_DioInterruptEdge()`. Enabling and masking of DIO interrupts is done using `vAHI_DioInterruptEnable()` while the status of a DIO interrupt is given by `u32AHI_DioInterruptStatus()`. See section 16 Power Management and Sleep Modes for further details on sleep and wakeup.

10 Serial Peripheral Interface

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the JN513x and peripheral devices. The JN513x operates as a master on the SPI bus and all other devices connected to the SPI are expected to be slave devices under the control of the JN513x CPU. The SPI includes the following features:

- Full-duplex, three-wire synchronous data transfer
- Programmable bit rates
- Programmable transaction size of 8, 16 or 32 bits
- Selectable transmit on positive or negative edge of clock
- Selectable receive on positive or negative edge of clock
- Automatic slave select generation (up to 5 slaves)
- Maskable transaction complete interrupt
- LSB First or MSB First Data Transfer

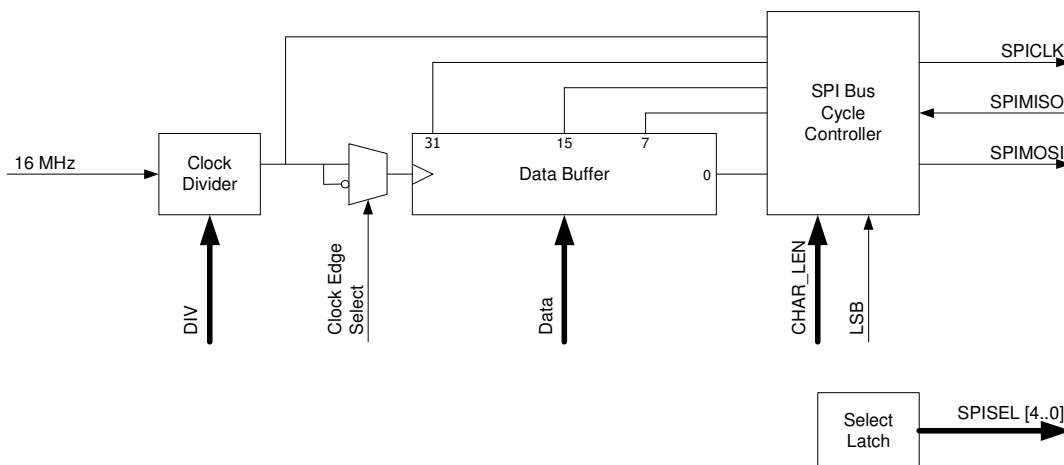


Figure 18: SPI Block Diagram

The SPI bus employs a simple shift register data transfer scheme. Data is clocked out of and into the active devices in a first-in, first-out fashion allowing SPI devices to transmit and receive data simultaneously.

There are three dedicated pins SPICLK, SPIMOSI, SPIMISO that are shared across all devices on the bus. Master-Out-Slave-In or Master-In-Slave-Out data transfer is relative to the clock signal SPICLK generated by the JN513x.

The JN513x provides five slave selects, SPISEL0 to SPISEL4 to allow five SPI peripherals on the bus. SPISEL0 is a dedicated pin and SPISEL1 to 4, are alternate functions of pins DIO0 to 3 respectively. This allows a serial flash memory to be connected to SPISEL0 and download to internal RAM via software from reset.

The interface can transfer 8, 16 or 32 bits without software intervention and can keep the slave select lines asserted between transfers when required, to enable longer transfers to be performed.

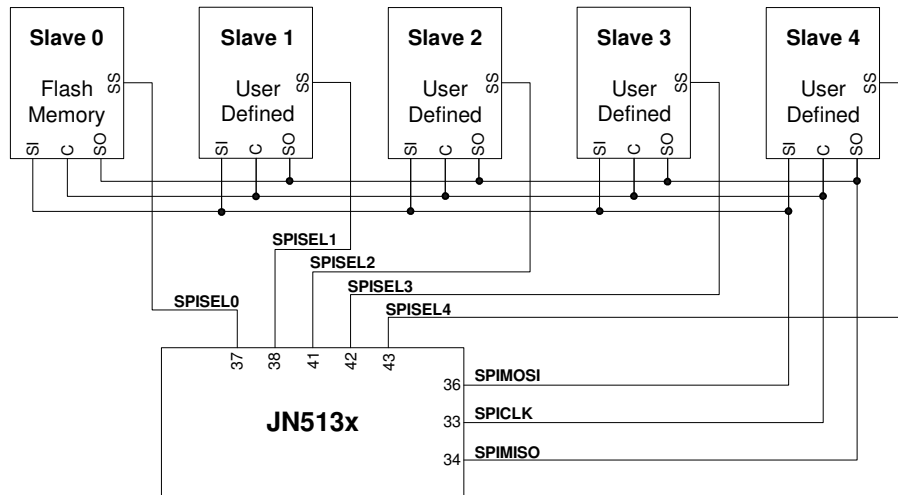


Figure 19: Typical JN513x SPI Peripheral Connection

The data transfer rate on the SPI bus is determined by the SPICLK signal. The JN513x supports transfers at selectable data rates from 16MHz to 250kHz selected by a clock divider. Both SPICLK clock phase and polarity are configurable. The SPICLK line is held low when the interface is not being used. The clock phase determines which edge of SPICLK is used by the JN513x to present new data on the SPIMOSI line; the opposite edge will be used to read data from the SPIMISO line. These options are specified using the `vAHI_SpiConfigure()` function.

The slave select outputs, SPISEL_n, are controlled using the `vAHI_SpiSelect()` function. If more than one SPISEL line is to be used in a system they must be used in numerical order, for instance if 3 SPI select lines are to be used, they must be SPISEL0, 1 and 2. The number of SPISEL lines to be used in a system is controlled using `vAHI_SpiConfigure()`. A SPISEL line can be automatically deasserted between transactions if required, or it may stay asserted over a number of transactions until removed by a call to `vAHI_SpiSelect()`. For devices such as memories where a large amount of data can be received by the master by continually providing SPICLK transitions, the ability for the select line to stay asserted is an advantage since it keeps the slave enabled over the whole of the transfer.

A transaction commences with the SPI bus being set to the correct configuration using `vAHI_SpiConfigure()`, and then the slave device being selected using `vAHI_SpiSelect()`. Transmit commences using the `vAHI_SpiStartTransferxx()` function (where xx is either 8, 16 or 32 bits). This will cause data to be placed in the FIFO data buffer and be clocked out, at the same time generating the corresponding SPICLK transitions. Since the transfer is full-duplex, the same number of data bits is being received from the slave as it transmits. The data that is received during this transmission can be read using `u32AHI_SpiReadTransferxx()` (again xx is either 8, 16 or 32 bits). If the master simply needs to provide a number of SPICLK transitions to allow data to be sent from a slave, it can perform a `vAHI_SpiStartTransferxx()` using dummy transmit data. An interrupt can be generated when the transaction has completed when enabled by `vAHI_SpiConfigure()`. Alternatively the interface can be polled using the `bAHI_SpiPollBusy()` or `vAHI_SpiWaitBusy()` functions.

If a slave device wishes to signal the JN513x indicating that it has data to provide, it may be connected to one of the DIO pins that can be enabled as an interrupt.

10.1 Programming Example

The following code example shows how to initialise the SPI and perform a simple read from a slave device. The device being read requires 40 clocks to send an 8-bit instruction, a 24-bit address and retrieve the 8-bit data. This cannot be achieved by a single transfer, so multiple transfers are combined without the automatic de-assertion of the selects. The waveforms generated by the example code are illustrated in Figure 20.

Programming Example	
<pre> PRIVATE void vReadFromFlash(uint32 u32Addr, uint32 u32NumWords, uint32 *pau32Buffer) { #define FLASHREADCMD 0x03 #define SPI_SLCT_NONE 0x00 uint32 u32Temp; uint32 i; vAHI_SpiConfigure(1, /* number of slave select lines in use */ E_AHI_SPIM_MSB_FIRST, /* send data MSB first */ E_AHI_SPIM_TXNEG_EDGE, /* TX data to change on negative edge */ E_AHI_SPIM_RXNEG_EDGE, /* RX data to change on negative edge */ 0, /* Generate 16MHz SPI clock */ E_AHI_SPIM_INT_DISABLE, /* Disable SPI interrupt */ E_AHI_SPIM_AUTOSLAVE_DSABL); /* Disable auto slave select */ /* combine read cmd & addr into single value to be sent over SPI */ u32Temp = (u32Addr & 0x00FFFFFF) (FLASHREADCMD << 24); /* select spi device */ vAHI_SpiSelect(E_AHI_SPIM_SLAVE_ENBLE_0); /* send read cmd and address location */ vAHI_SpiStartTransfer32(u32Temp); vAHI_SpiWaitBusy(); for (i=0; i<=u32NumWords; i++) { /* read data 4 bytes at a time */ vAHI_SpiStartTransfer32(u32Addr); vAHI_SpiWaitBusy(); /* copy into temp variable */ u32Temp = u32AHI_SpiReadTransfer32(); /* copy to buffer */ memcpy((pau32Buffer+i), &u32Temp, sizeof(u32Temp)); } /* deselect select spi device */ vAHI_SpiSelect(SPI_SLCT_NONE); } </pre>	

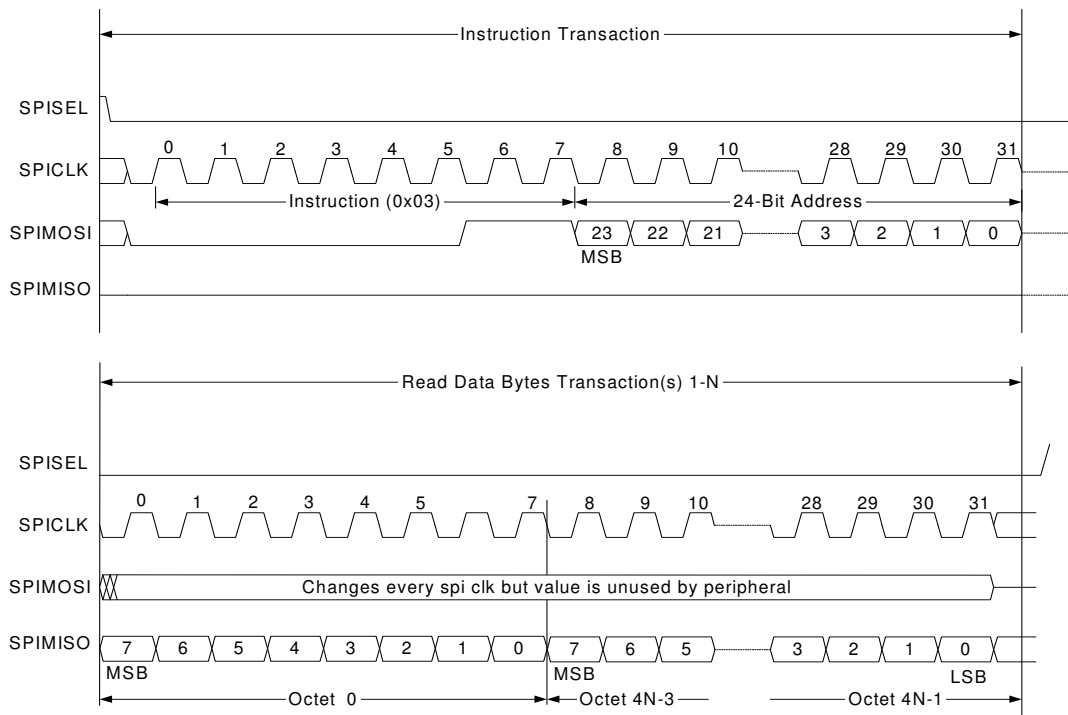


Figure 20: SPI Transaction Waveforms

11 Intelligent Peripheral Interface

The Intelligent Peripheral (IP) Interface is provided for systems that are more complex, where there is a processor that requires a wireless peripheral. As an example, the JN513x may provide a complete IEEE802.15.4, ZigBee or other wireless network to a phone, computer, PDA, set-top box or games console. No resources are required from the main processor compared to a transceiver as the complete wireless protocol may be run on the internal JN513x CPU. The wireless peripheral may be controlled via one of the UARTs but the IP interface is intended to provide a high-speed, low-processor-overhead interface.

The intelligent peripheral interface is a SPI slave interface and uses pins shared with other DIO signals. The interface is designed to allow message passing and data transfer. Data received and transmitted on the IP interface is copied directly to and from a dedicated area of memory without intervention from the CPU. This memory area, the intelligent peripheral memory block, contains 64 32-bit word receive and transmit buffers.

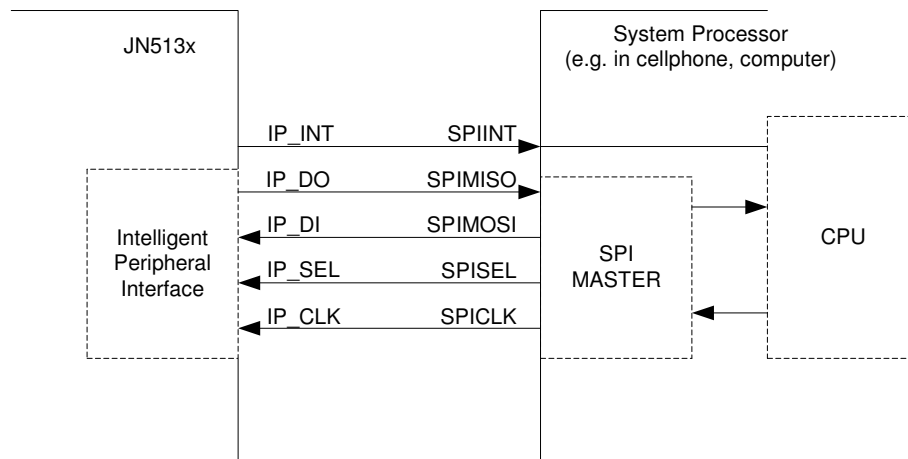


Figure 21: Intelligent Peripheral Connection

The interface conforms to the SPI protocol as described in section 10. It is possible to select the clock edge of IP_CLK on which data on the IP_DIN line of the interface is sampled, and the state of data output IP_DOUT is changed. The order of transmission is MSB first. The IP_DO data output is tri-stated when the device is inactive, i.e. the device is not selected via IP_SEL. An interrupt output line IP_INT is available so that the JN513x can indicate to an external master that it has data to transfer.

The IP interface signals IP_CLK, IP_DO, IP_DI, IP_SEL, IP_INT are alternate functions of pins DIO14 to 18 respectively.

11.1 Data Transfer Format

Transfers are started by the remote processor asserting the IPSEL line and terminated by the remote processor de-asserting IP_SEL.

Data transfers are bi-directional and traffic in both directions has a format of status byte, data length byte (of the number of 32-bit words to transfer) and data packet (from the receive and transmit buffers). The first byte transferred in either direction is a status byte with the following format:

Bit	Field	Description
7:2	RSVD	Reserved, set to 0.
1	TXQ	1: Data queued for transmission
0	RXRDY	1: Buffer ready to receive data

Table 2: IP Status Byte Format

If data is queued for transmission and the recipient has indicated that they are ready for it (RXRDY in incoming status byte was 1), the next byte to be transmitted is the data length in words. If either the JN513x or the remote processor

has no data to transfer, then the data length should set to zero. The transaction can be terminated by the master after the status byte has been sent if it is not possible to send data in either direction. This may be because neither party has data to send or because the receiver does not have a buffer available. If the data length is non-zero, the data in the JN513x transmit memory buffer is sent, beginning at the start of the buffer. At the same time that data bytes are being sent from the transmit buffer, the JN513x receive buffer is being filled with incoming data, beginning from the start of the buffer.

The remote processor, acting as the master must determine the larger of its incoming or outgoing data transfers and deassert IP_SEL when all of the transmit and receive data has been transferred. The data is transferred into or out of the buffers starting from the lowest address in the buffer, and each word is assembled with the MSB first on the serial data lines.

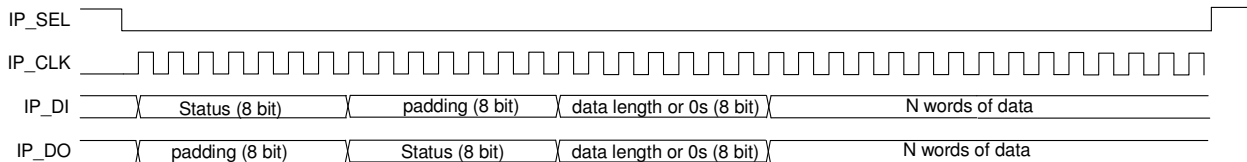


Figure 22: Intelligent Peripheral Data Transfer Waveforms

11.2 JN513x Initiated Data Transfer

To send data, the data is written into either buffer 0 or 1 of the intelligent peripheral memory area. Then the buffer number is written together with the data length using `bAHI_IpSendData()`. If the call is successful, the interrupt line IP_INT will signal to the remote processor that there is a message ready to be sent from the JN513x. When a remote processor starts a transfer to the JN513x by deasserting IP_SEL, then IP_INT is deasserted. If the transfer is unsuccessful and the data is not output then IP_INT is reasserted after the transfer to indicate that data is still waiting to be sent.

The interface can be configured to generate an internal interrupt whenever a transaction completes (for example IP_SEL becomes inactive after a transfer starts). It is also possible to mask the interrupt. The end of the transmission can be signalled by an interrupt, or the interface can be polled using the function `bAHI_IpTxDone()`

To receive data the interface must first be initialised using `vAHI_IpEnable()`. When this is done, the bit RXRDY sent in the status byte from the IP block will show that data can be received by the JN513x. Successful data arrival can be indicated by an interrupt, or the interface can be polled using `bAHI_IpRxDataAvailable()`. Data is then retrieved using `bAHI_IpReadData()`.

To send and receive at the same time, the transmit and the receive buffers must be set to be different.

11.3 Remote Processor Initiated Data Transfer

The remote processor (master) may initiate a transfer to send data to the JN513x by asserting the slave select pin, IP_SEL, and generating its status byte on IP_DI with TXRDY set. After receiving the status byte from the JN513x, it should check that the JN513x has a buffer ready by reading the RXRDY bit. If the RXRDY bit is 0 indicating that the JN513x cannot accept data, it should terminate the transfer by deasserting IP_SEL unless it is receiving data from the JN513x. If the RXRDY bit is 1, indicating that the JN513x can accept data, then the master should generate a further 8 clocks on IP_CLK in order to transfer its own message length on IP_DI. The master should continue clocking the interface until sufficient clocks have been generated to send all the data specified in the length field to the JN513x. The master should then deassert IP_SEL to show the transfer is complete.

The master may initiate a transfer to read data from the JN513x by asserting the slave select pin, IP_SEL, and generating its status byte on IP_DI with RXRDY set. After receiving the status byte from the JN513x, it should check that the JN513x has a buffer ready by reading the TXRDY bit. If the TXRDY bit is 0, indicating that the JN513x does not have data to send, it should terminate the transfer by deasserting IP_SEL unless it is transmitting data to the JN513x. If the TXRDY bit is 1, indicating that the JN513x can send data, then the master should generate a further 8 clocks on IP_CLK in order to receive the message length on IP_DO. The master should continue clocking the interface until sufficient clocks have been generated to receive all the data specified in the length field from the JN513x. The master should then deassert IP_SEL to show the transfer is complete.

Data can be sent in both directions at once and the master must ensure both transfers have completed before deasserting IP_SEL.

12 Timers

12.1 Peripheral Timer / Counters

Two general-purpose timer / counter units are available that can be independently configured to operate in one of five modes. The timers have the following features:

- 16-bit prescaler, divides system clock by of 2^{prescale} value as the clock to the timer
- Clocked from internal system clock
- 16-bit counter, 16-bit Rise and Fall (period) registers
- Timer: can generate interrupts off Rise and Fall counts. Can be gated by external signal
- Counter: counts number of transitions on external event signal. Can use low-high, high-low or both transitions
- PWM/Single pulse: outputs repeating Pulse Width Modulation signal or a single pulse. Can set period and mark-space ratio
- Capture: measures times between transitions of an applied signal.
- Delta-Sigma: Return-To-Zero (RTZ) and Non-Return-to-Zero (NRZ) modes

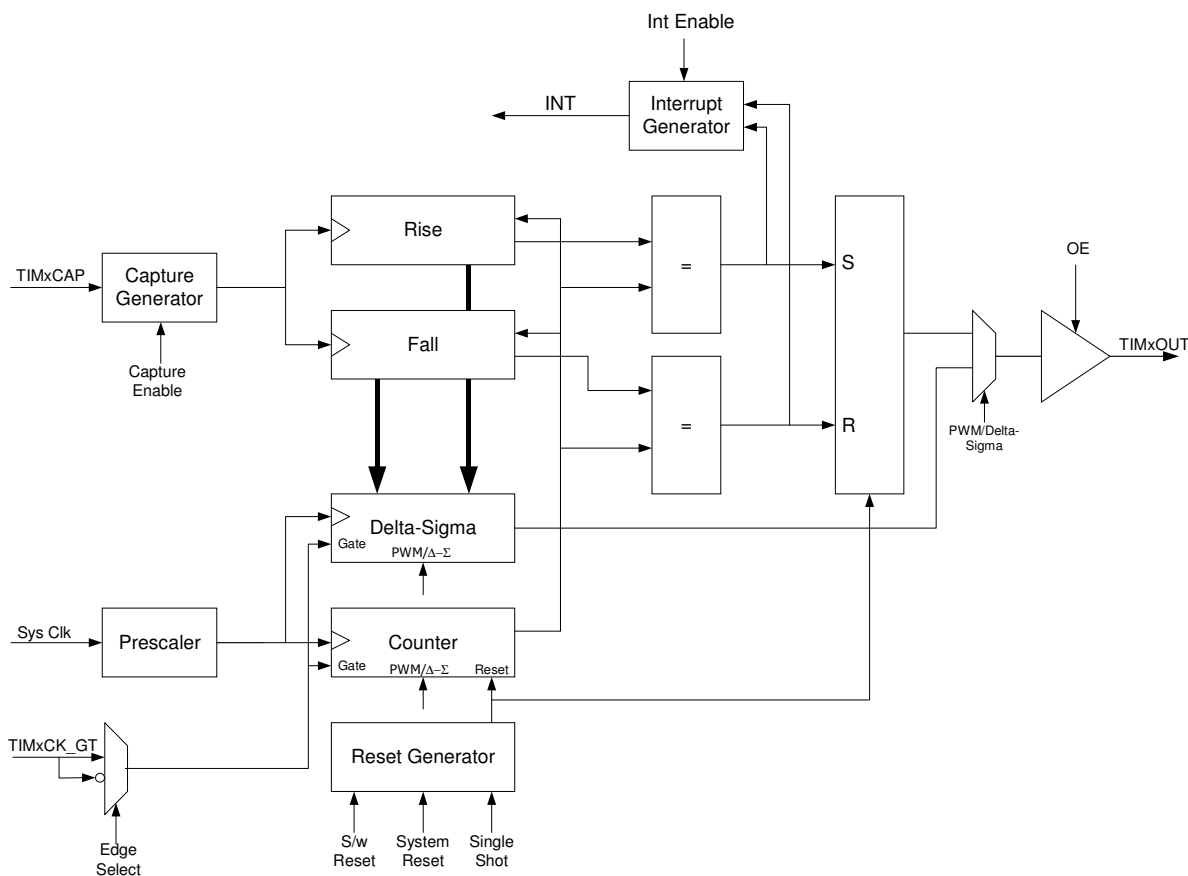


Figure 23: Timer Unit Block Diagram

The clock source for the timer unit is fed from the 16MHz system clock. This clock passes to a 16-bit prescaler where a value of 0 leaves the clock unmodified and other values divide it by 2^{prescale} value. For example, a prescale value of 2 applied to the 16MHz system clock source results in a timer clock of 4MHz. The value of the prescaler is set using the `vAHI_TimerEnable()` function.

The counter is optionally gated by a signal on the clock / gate input (TIMxCK_GT). If the gate function is selected (using `vAHI_TimerClockSelect()`) the counter is frozen when the clock/gate input is high.

If enabled using the `vAHI_TimerEnable()` function, an interrupt is generated whenever counter is equal to the value stored in either of the High or Low registers.

The internal Output Enable (OE) signal enables or disables the timer output.

The Timer 0 signals CK_GT, CAP and OUT are alternate functions of pins DIO8, 9 and 10 respectively and Timer 1 signals CK_GT, CAP and OUT are alternate functions of pins DIO11, 12, and 13 respectively. Selection of either the Timer or DIOx functionality is through software, in either case the timer still functions internally.

12.1.1 Pulse Width Modulation Mode

Pulse Width Modulation (PWM) mode allows the user to specify an overall cycle time and pulse length within the cycle. The pulse can be generated either as a single shot or as a train of pulses with a repetition rate determined by the cycle time.

In this mode, the cycletime and low periods of the PWM output signal can be set by the values of two independent 16-bit registers (Fall and Rise). The counter increments and its output is compared to the 16-bit Rise and Fall registers. When the counter is equal to the Rise register, the PWM output is set to high; when the counter reaches the Fall value, the output returns to low. In continuous mode, when the counter reaches the Fall value, it will reset and the cycle repeats. Depending upon the mode of operation either the `vAHI_TimerStartRepeat()` function or the `vAHI_TimerStartSingleShot()` is used to set the values of the High and Low registers. The PWM waveform is available on TIMxOUT when the output driver is enabled using `vAHI_TimerEnable()`.

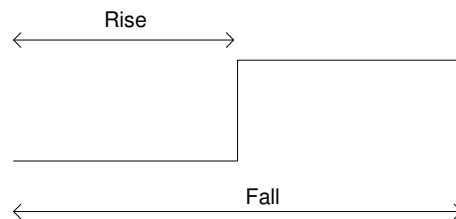


Figure 24: PWM Output Timings

12.1.2 Capture Mode

The capture mode can be used to measure the time between transitions of a signal applied to the capture input (TIMxCAP). The mode is selected and the counter started by `vAHI_TimerStartCapture()`. On the next low-to-high transition of the captured signal, the count value is stored in the Rise register, and on the following high-to-low transition, the counter value is stored in the Fall register. The pulse width is the difference in counts in the two registers multiplied by the driving clock (in all cases this must be the 16MHz clock and so the prescaler must be set to 0). The counter is stopped and Low and High registers read with `vAHI_TimerReadCapture()`. The values in the High and Low registers will be updated whenever there is a corresponding transition on the capture input, and the value stored will be relative to when the mode was started. Therefore, if multiple pulses are seen on TIMxCAP before the counter is stopped only the last pulse width will be stored.

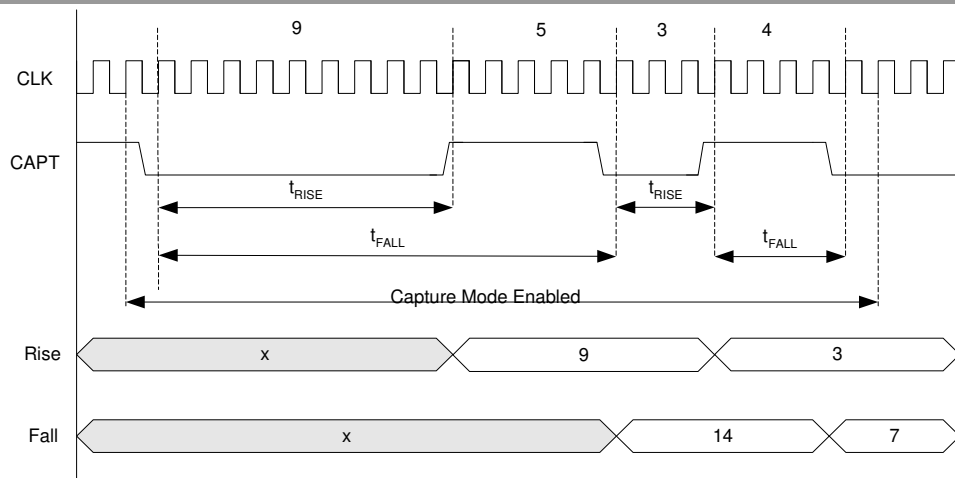


Figure 25: Capture Mode

12.1.3 Counter / Timer Mode

The counter/timer can be used to generate timing or count interrupts for software to use. As a timer the clock source is from the system clock, prescaled if required. The timer period is programmed into the Fall register and the Fall register match interrupt enabled. The timer is started either as a single-shot or repeating timer (`vAHI_TimerStartSingleShot()` or `vAHI_TimerStartRepeat()`), and generates an interrupt when the counter reaches the Fall register value.

When used to count external events on TIMxCK_GT the clock source is selected from the input pin and the number of events programmed into the Fall register. The Fall register match interrupt is enabled and the counter started, usually in single shot mode. An interrupt is generated when the programmed number of low-to-high transitions is seen on the input pin.

12.1.4 Delta-Sigma Mode

A separate delta-sigma mode is available, allowing a low speed delta-sigma DAC to be implemented with up to 16-bit resolution. This requires that a resistor-capacitor network is placed between the output DIO pin and digital ground. A stream of pulses with digital voltage levels is generated which is integrated by the RC network to give an analogue voltage. A conversion time is defined in terms of a number of clock cycles. The width of the pulses generated is the period of a clock cycle. The number of pulses output in the cycle, together with the integrator RC values will determine the resulting analogue voltage. For example, generating approximately half the number of pulses that make up a complete conversion period will produce a voltage on the RC output of $VDD1/2$, provided the RC time constant is chosen correctly. During a conversion, the pulses will be pseudo-randomly dispersed throughout the cycle in order to produce a steady voltage on the output of the RC network.

The output signal is asserted for the number of clock periods defined in the High register set by `vAHI_TimerStartDeltaSigma()`, with the total period being 2^{16} cycles. For the same value in the High register the pattern of pulses on subsequent cycles is different, due to the pseudo-random distribution.

The delta-sigma convertor output can operate in a Return-To-Zero (RTZ) or a Non-Return-to-Zero (NRZ) mode. The NRZ mode will allow several pulses to be output next to each other. The RTZ mode ensures that each pulse is separated from the next by at least one period. This improves linearity if the rise and fall times of the output are different to one another. Essentially, the output signal is low on every other output clock period, and the conversion cycle time is twice the NRZ cycle time ie 2^{17} clocks. The integrated output will only reach half $VDD2$ in RTZ mode, since even at full scale only half the cycle contains pulses. Figure 26 and Figure 27 illustrate the difference between RTZ and NRZ for the same programmed number of pulses.

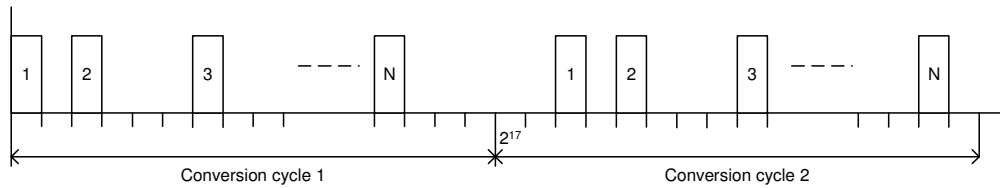


Figure 26: Return To Zero Mode in Operation

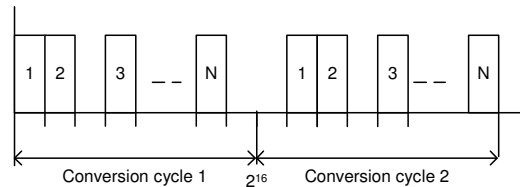


Figure 27: Non-Return to Zero Mode

12.1.5 Timer / Counter Application

Figure 28 shows an application of the JN513x timers to provide closed loop speed control. Timer 0 is configured in PWM mode to provide a variable mark-space ratio switching waveform to the gate of the NFET. This in turn controls the power in the DC motor.

Timer 1 is configured to count the rising edge events on the clk/gate pin over a constant period. This converts the tacho pulse stream output into a count proportional to the motor speed. This value is then used by the application software executing the control algorithm.

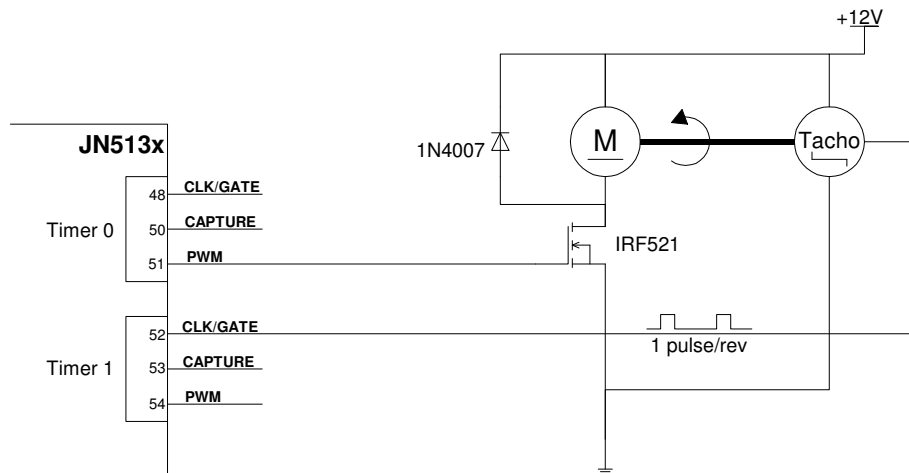


Figure 28: Closed Loop PWM Speed Control Using JN513x Timers

12.2 Tick Timer

The JN513x contains a hardware timer that can be used for generating timing interrupts to software. It may be used to implement regular events such as ticks for software timers or an operating system, as a high-precision timing reference or can be used to implement system monitor timeouts as used in a watchdog timer. Features include:

- 32-bit counter
- 28-bit match value
- Maskable timer interrupt
- Single-shot, Restartable or Continuous modes of operation

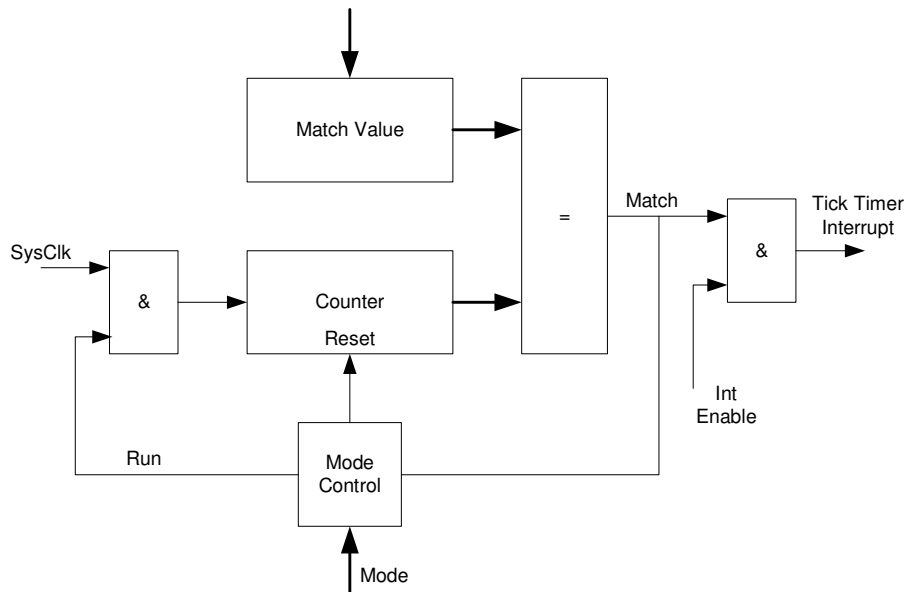


Figure 29: Tick Timer

The Tick Timer is clocked from the system clock (16 MHz), which is fed to a 32-bit wide resettable up-counter, gated by a signal from the mode control block. A match register allows comparison between the counter and a programmed value. The match value, measured in 16MHz clock cycles can be programmed using `vAHI_TickTimerInterval()`, in the range 0 to 0x0FFFFFFF. The output of the comparison can be used to generate an interrupt if the interrupt is enabled and used in controlling the counter in the different modes. The mode is programmed using `vAHI_TickTimerConfigure()`, which also resets the counter to zero.

The interrupt is enabled by `vAHI_TickTimerIntEnable()`. The interrupt state is returned by `bAHI_TickTimerIntStatus()` and if an interrupt is generated it can be cleared by `vAHI_TickTimerIntPndClr()`.

If the mode is programmed as single shot, the counter begins to count from zero until the match value is reached. The match signal will be generated which will cause an interrupt if enabled, and the counter will stop counting. The counter can be restarted by reprogramming the mode using `vAHI_TickTimerConfigure()`.

If the mode is programmed as restartable, the operation of the counter is the same as for the single shot mode, except that when the match value is reached the counter is reset and begins counting from zero. An interrupt will be generated when the match value is reached if it is enabled.

Continuous mode operation is similar to restartable, except that when the match value is reached, the counter is not reset but continues to count. An interrupt will be generated when the match value is reached if enabled.

12.3 Wakeup Timers

Two 32-bit wakeup timers are available in the JN513x driven from the 32kHz internal clock. They may run during sleep periods when the majority of the rest of the device is powered down, to time sleep periods or other long period timings that may be required by the application. The wakeup timers do not run during deep sleep and may optionally be disabled in sleep mode through software control. When a wakeup timer expires it typically generates an interrupt, if the device is asleep then the interrupt may be used as an event to end the sleep period. See Section 16 for further details on how they are used during sleep periods. Features include:

- 32-bit down-counter
- Optionally runs during sleep periods
- Clocked from 32 kHz RC oscillator

A wakeup timer consists of a 32-bit down counter clocked from the 32 kHz internal clock. An interrupt or wakeup event can be generated when the counter reaches zero. On reaching zero the counter will continue to count down until stopped, which allows the latency in responding to the interrupt to be measured. If an interrupt or wakeup event is required, the timer interrupt should be enabled using `vAHI_WakeTimerEnable()` before loading the count value for the period. The count value is loaded using `vAHI_WakeTimerStart()` and causes the counter to begin to count down to zero; the counter can be stopped at any time using `vAHI_WakeTimerStop()`. The counter will remain at the value it contained when the timer was stopped and no interrupt will be generated. The status of the timers can be checked using the `u8AHI_WakeTimerStatus()` function, which indicates if the timers are running. The timers can be checked to see if they have expired using `u8AHI_WakeTimerFiredStatus()` which is useful when the timer interrupts are masked. If a timer has expired then the fired status will be reset by the function.

12.3.1 RC Oscillator Calibration

The RC oscillator used to time sleep periods is designed to require very little power to operate and be self-contained, requiring no external timing components and hence is lower cost. As a consequence of using on-chip resistors and capacitors, the inherent absolute accuracy and temperature coefficient is lower than that of a crystal oscillator, but once calibrated the accuracy approaches that of a crystal oscillator. Sleep time periods should be as close to the desired time as possible in order to allow the device to wake up in time for important events, for example beacon transmissions in the IEEE802.15.4 protocol. If the sleep time is accurate, the device can be programmed to wake up very close to the calculated time of the event and so keep current consumption to a minimum. If the sleep time is less accurate, it will be necessary to wake up earlier in order to be certain the event will be captured. If the device wakes earlier, it will be awake for longer and so reduce battery life.

In order to allow sleep time periods to be as close to the desired length as possible, the true frequency of the RC oscillator needs to be determined to better than the initial 30% accuracy. The calibration factor can then be used to calculate the true number of nominal 32kHz periods needed to make up a particular sleep time. A calibration reference timer, clocked from the crystal oscillator, is provided to allow comparisons to be made between the RC clock and the 16MHz crystal oscillator when the JN513x is awake. Operation is as follows:

- Wakeup timer0 is disabled and programmed with a number of 32kHz ticks
- Calibration mode is enabled which causes the Calibration Reference counter to be zeroed. Both counters start counting, the wakeup timer decrementing and the calibration counter incrementing
- When the wakeup timer reaches zero the Reference Counter is stopped, allowing software to read the number of 16MHz clock ticks generated during the time represented by the number of 32kHz ticks programmed in the wakeup timer. The true period of the 32kHz clock can thus be determined and used when programming a wakeup timer to achieve a better accuracy and hence more accurate sleep periods

The RC oscillator has a good temperature coefficient for an oscillator of its class (see section 17.3.5) however this should be taken into account for any given application, when planning the wake up events and the time interval between calibrations.

A calibration can be performed by calling `u32AHI_WakeTimerCalibrate()`, which calibrates over twenty 32kHz ticks and returns the number of 16MHz ticks recorded. For a RC oscillator running at exactly 32kHz the value returned should be 10000. If the oscillator is running faster than 32kHz the count will be less than 10000, if running slower the value will be higher. For a calibration count of 9000, indicating that the RC oscillator period is running at approximately 35kHz, to time for a period of 2 seconds the timer should be loaded with $71,112 ((10000/9000) * (32000*2))$ rather than 64000.

12.3.2 External 32kHz Clock Source

It is possible to change the source of 32kHz clock used for the sleep timers, to an externally supplied 32kHz reference clock on the CLK32K input (DIO9). This mode could allow the timer clock to be sourced from a very stable oscillator model, allowing more accurate sleep cycle timings.

13 Serial Communications

The JN513x has two independent Universal Asynchronous Receiver / Transmitter (UART) serial communication interfaces. These provide similar operating features to the industry standard 16550A device operating in FIFO mode. Each interface performs serial-to-parallel conversion on incoming serial data and parallel-to-serial conversion on outgoing data from the CPU to external devices. In both directions, a 16-byte deep FIFO buffer allows the CPU to read and write multiple characters on each transaction. This means that the CPU is freed from handling data on a character-by-character basis, with the associated high processor overhead. The UARTs have the following features:

- Emulates behaviour of industry standard NS16450 and NS16550 UARTs
- 16 byte transmit and receive FIFO buffers reduce interrupts to CPU
- Adds / deletes standard start, stop and parity communication bits to or from the serial data
- Independently controlled transmit, receive, status and data sent interrupts
- Optional modem flow control signals CTS and RTS
- Fully programmable data formats: baud rate, start, stop and parity settings
- False start bit detection
- Internal diagnostic capabilities: loop-back controls for communications link fault isolation

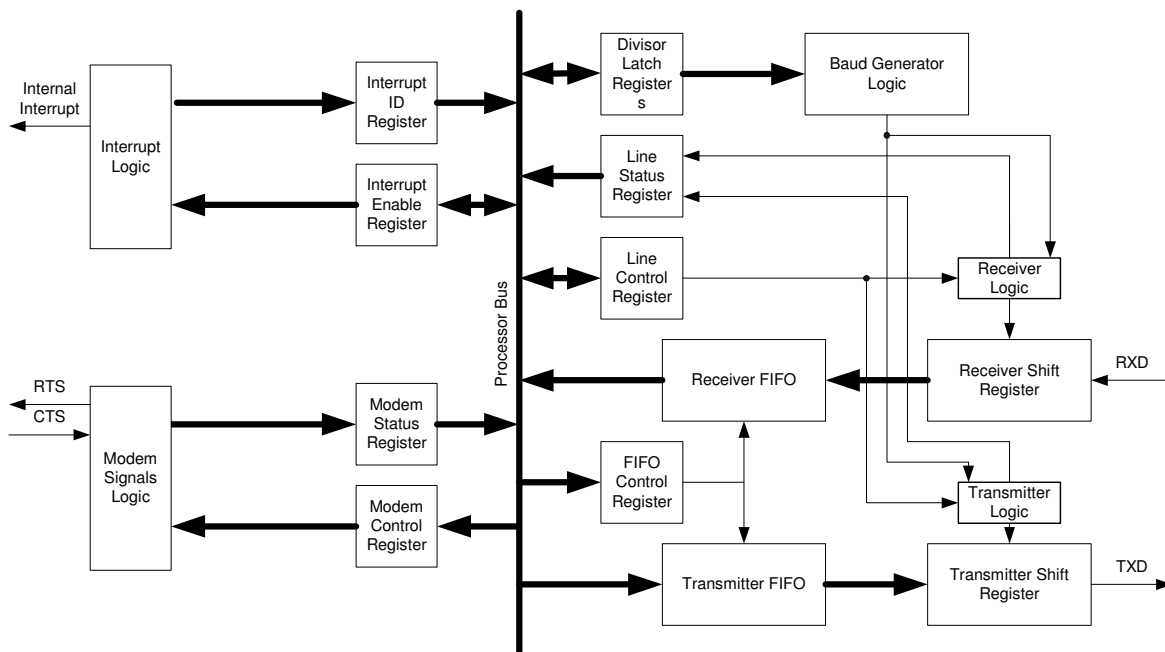


Figure 30 UART Block Diagram

The serial interface characteristics are programmed using the peripheral library call `vAHI_UartSetControl()`. This sets the number of data bits (5, 6, 7 or 8), even, odd, set-at-1, set-at-0 or no-parity detection and generation and single or multiple stop bit generation (for 5 bit data, multiple is 1.5 stop bits; for 6, 7 or 8 data bits, multiple is 2 bits).

The baud rate is programmable between 4800, 9600, 19.2k, 38.4k, 76.8k and 115.2 kbaud via the `vAHI_UartSetClockDivisor()` function. For higher or non-standard baud rates, the registers of the UART may be accessed directly to achieve the desired programming.

Two hardware flow control signals are provided: Clear-To-Send (CTS) and Request-To-Send (RTS). CTS is an indication sent by an external device to the UART that it is ready to receive data. RTS is an indication sent by the UART to the external device that it is ready to receive data. Both signals are active low. RTS is controlled from software using the `vAHI_UartSetControl()` function, while the value of CTS can be read using `u8AHI_UartReadModemStatus()`. The result of this routine also indicates if the state of CTS has changed, indicating that the connected device has signalled the UART that it can begin transmitting. Monitoring and control of

CTS and RTS is a software activity, normally performed as part of interrupt processing. The signals do not control parts of the UART hardware, but simply indicate to software the state of the UART external interface.

Characters are read one byte at a time from the Receive FIFO using the `u8AHI_UartReadData()` routine and are written to the Transmit FIFO using `vAHI_UartWriteData()`. The Transmit and Receive FIFOs can be cleared and reset independently of each other using `vAHI_UartReset()`. The status of the transmitter can be checked using `u8AHI_UartReadLineStatus()`, which indicates if the transmit FIFO is empty, and if there is a character being transmitted. The status of the receiver is also checked using this call, which can indicate if conditions such as parity error, framing error or break indication have occurred. It also shows if an overrun error occurred (receive buffer full and another character arrives) and if there is data held in the receive FIFO.

UART 0 signals CTS, RTS, TXD and RXD are alternate functions of pins DIO4, 5, 6 and 7 respectively and UART 1 signals CTS, RTS, TXD and RXD are alternate functions of pins DIO17, 18, 19 and 20 respectively. If CTS and RTS are not required on the devices external pins, then they may be disabled through software control, this allows the alternate DIOx to be used instead

13.1 Interrupts

Interrupt generation is controlled for the UART block using the `vAHI_UartSetInterrupt()` routine, and are divided into four categories:

- Received Data Available: Is set when data in the Rx FIFO queue reaches a particular level. The trigger level can be configured as 1, 4, 8 or 14.
- Transmit Character Buffer Empty: Is set when the current character transmission has completed.
- Receiver Line Status: Is set when one of the following occur (1) Parity Error - the character at the head of the receive FIFO has been received with a parity error, (2) Overrun Error - the FIFO is full and another character has been received at the Receiver shift register, (3) Framing Error - the character at the head of the receive FIFO does not have a valid stop bit and (4) Break Interrupt – occurs when the RxD line has been held low for an entire character. The source of the interrupt is determined using `u8AHI_UartReadLineStatus()`
- Modem Status: Generated when the CTS (Clear To Send) input control line changes.

13.2 UART Application

The following example shows the UART connected to a 9-pin connector compatible with a PC. The software developer kit uses such an interface as the debugger interface between the JN513x and a PC. As the JN513x device pins do not provide the RS232 line voltage a level shifter is used.

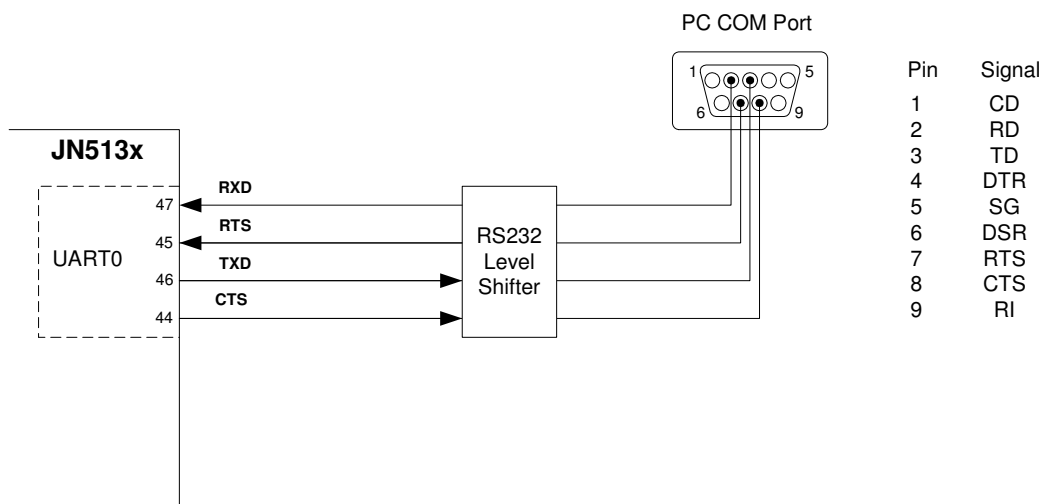


Figure 31 JN513x Serial Communication Link

13.3 Programming Example

The following code shows the peripheral library calls to configure UART0 and output the message 'Hello World'

Programming Example

```
/* Set up uart0 */
vAHI_UartEnable(E_AHI_UART_0);

/* Reset the Tx and Rx */
vAHI_UartReset(E_AHI_UART_0, E_AHI_UART_TX_RESET, E_AHI_UART_RX_RESET);

/* set baud rate */
vAHI_UartSetClockDivisor(0, E_AHI_UART_RATE_38400);

/* set parity, start bits, number data bits */
vAHI_UartSetControl(E_AHI_UART_0,
                    E_AHI_UART_EVEN_PARITY,
                    E_AHI_UART_PARITY_DISABLE,
                    E_AHI_UART_WORD_LEN_8,
                    E_AHI_UART_1_STOP_BIT,
                    E_AHI_UART_RTS_HIGH);

/* clear reset */
vAHI_UartReset(E_AHI_UART_0, E_AHI_UART_TX_ENABLE, E_AHI_UART_RX_ENABLE);

/* output message */
char acstring[] = "Hello World";
char *pcstring = acstring;

while (*pcstring)
{
    vAHI_UartWriteData(E_AHI_UART_0, *pcstring);
    pcstring++;
}
```

14 Two-Wire Serial interface

The JN513x includes an industry standard two-wire synchronous serial interface (SIF) that provides a simple and efficient method of data exchange between devices. The system uses a serial data line (SIF_D) and a serial clock line (SIF_CLK) to perform bi-directional data transfers and includes the following features:

- Compatible with both I²C and SMBus peripherals
- Multi-master operation
- Software programmable clock frequency
- Clock stretching and wait state generation
- Software programmable acknowledge bit
- Interrupt or bit-polling driven byte-by-byte data-transfers
- Bus busy detection
- Support for 7 and 10 bit addressing modes

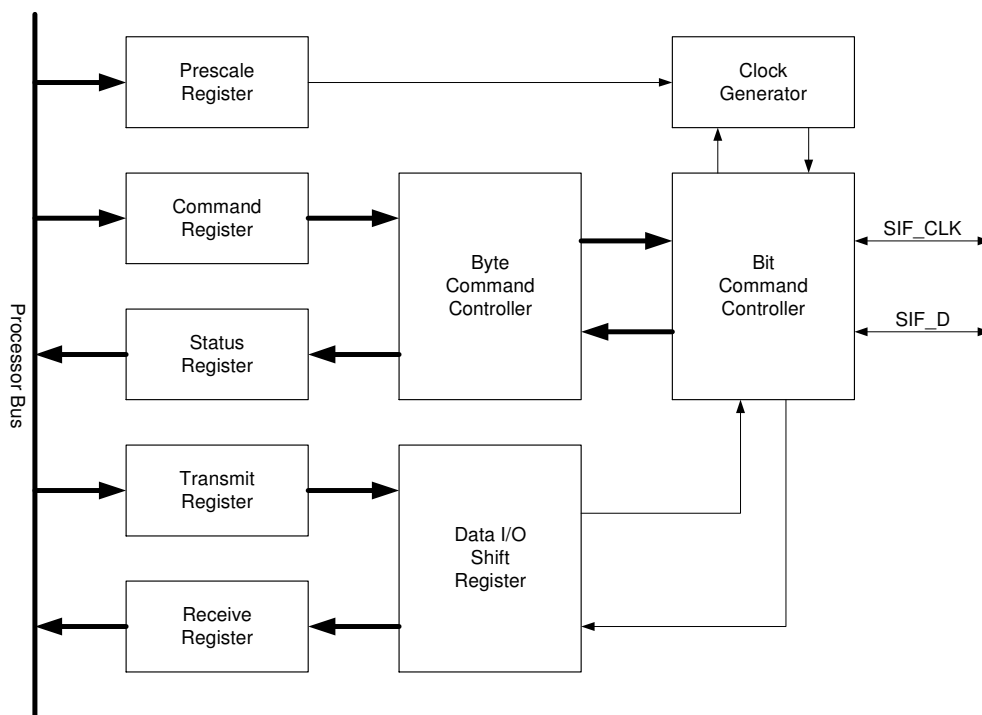


Figure 32: SIF Block Diagram

The prescale register, set using the `vAHI_SiConfigure()` function, allows the interface to be configured to operate at up to 400kbit/s. The clock generator handles the clock stretching required by some slave devices.

The Byte Command Controller handles traffic at the byte level. It takes data from the Command Register and translates it into sequences based on the transmission of a single byte. By setting the start, stop, read, write and acknowledge control bits in the command register using the `vAHI_SiSetCmdReg()` function it is possible to generate read or write sequences on the bus.

The data I/O shift register contains the data associated with the current transfer. During a read operation, data is shifted into this register from the SIF_D line. When the read is complete the byte is copied into the receive register and can be accessed using the `u8AHI_SiReadData8()` function.

During a write operation the contents of the transmit register are copied into the shift register and then onto the SIF_D line. The transmit register can be accessed using the `vAHI_SiWriteData8()` function. It is possible to generate

an interrupt upon the completion of a byte transmission or reception. If required this interrupt can be enabled by using the `vAHI_SiConfigure()` function.

If interrupt-driven communication is not desired it is possible to poll the status of the interface by using the `bAHI_SiPollBusy()` and `bAHI_SiPollTransferInProgress()` functions.

The first byte of data transferred by the device after a start bit is the slave address. The JN513x supports both 7-bit and 10-bit slave addresses by generating either one or two address transfers. Only the slave with a matching address will respond by returning an acknowledge bit. The slave address to be used is set using the `vAHI_SiWriteSlaveAddr()` function.

The SIF signals SIF_CLK, SIF_D are alternate functions of pins DIO14 and 15 respectively.

14.1 Connecting Devices

The clock and data lines, SIF_D and SIF_CLK, are alternate functions of DIO lines 14 and 15 respectively. The serial interface function of these pins is selected when the interface is enabled using the `vAHI_SiConfigure()` function. They are both bi-directional lines, connected internally to the positive supply voltage via weak (45kΩ) programmable pull-up resistors. However, it is recommended that external 4.7kΩ pull-ups be used for reliable operation at high bus speeds, as shown in Figure 33. When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collector in order to perform the wired-AND function. The number of devices connected to the bus is solely dependent on the bus capacitance limit of 400pF.

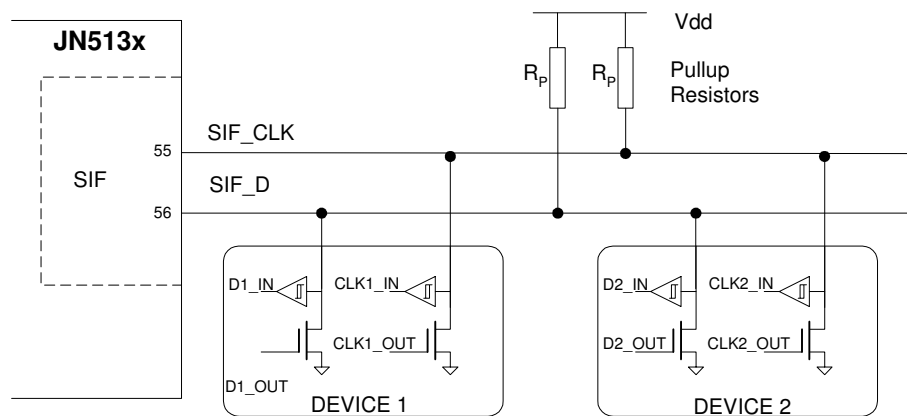


Figure 33: Connection Details

14.2 Multi-Master Operation

The interface provides a true multi-master bus including collision detection and arbitration that prevents data corruption. If two or more masters simultaneously try to control the bus, a clock synchronization procedure determines the bus clock. Because of the wired-AND connection of the interface, a high-to-low transition on the bus affects all connected devices. This means a high-to-low transition on the SIF_CLK line causes all concerned devices to count off their low period. Once a devices clock input has gone low, it will hold the SIF_CLK line in that state until the clock high state is reached. Due to the wired-AND connection, the SIF_CLK line will therefore be held low by the device with the longest low period, and held high by the device with the shortest high period.

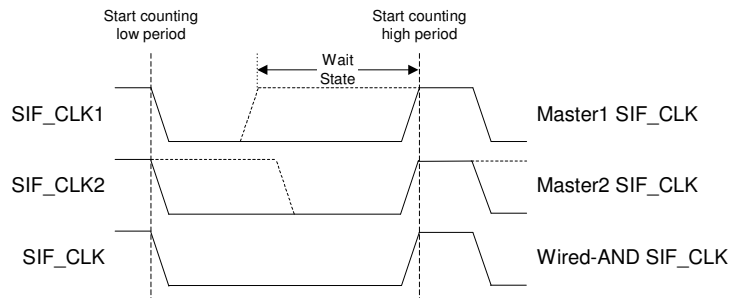


Figure 34: Multi-Master Clock Synchronization

14.3 Clock Stretching

Slave devices can use clock stretching to slow down the transfer bit rate. After the master has driven SIF_CLK low, the slave can drive SIF_CLK low for the required period and then release it. If the slave's SIF_CLK low period is greater than the master's low period the resulting SIF_CLK bus signal low period is stretched thus inserting wait states.

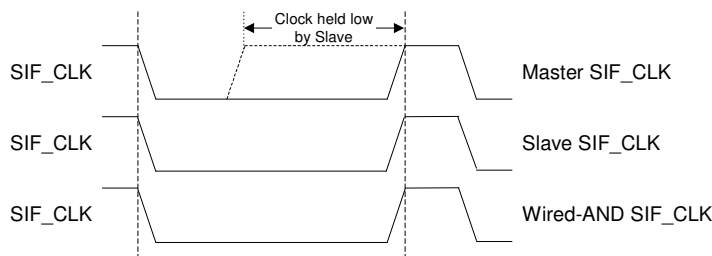


Figure 35: Clock Stretching

14.4 Programming Example

The two-wire serial interface protocol is implemented by a combination of hardware and software. Normally, a standard communication cycle consists of four parts:

- Start signal generation
- Slave address transfer
- Data transfer
- Stop signal generation

The hardware API supports several calls to support the protocol on the interface. All bit-level timing is implemented by dedicated hardware within the JN513x. The following code example shows how to read a set of values for a slave device into a buffer. A typical application would be data logging from a sensor.

Note that `bAHI_SiPollTransferInProgress()` function is used to block execution until a byte has been transferred. Higher performance applications should use interrupts to detect end of transfer, running the two-wire interface as a background task outside the main program thread.

The waveforms below illustrate the operation of the `bSIFRead()` function listed on the following page.

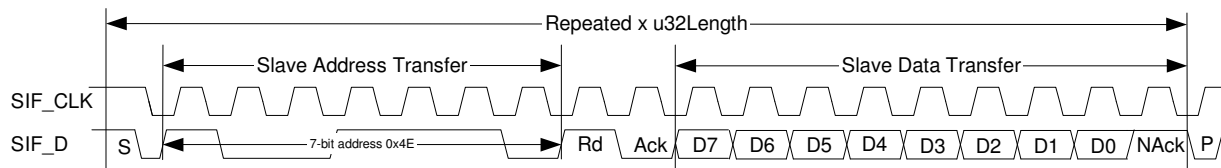


Figure 36: Read From Slave Device

Programming Example

```

PRIVATE bool_t bSIFRead(uint8 u8SlaveAddress, uint8 *pau8ReadBuffer, uint32
u32Length)
{
    int i;
    for (i=0; i<u32Length; i++)
    {
        /* set slave address */
        vAHI_SiWriteSlaveAddr(u8SlaveAddress, E_AHI_SI_SLAVE_RW_SET);

        /* send read command */
        vAHI_SiSetCmdReg(E_AHI_SI_START_BIT, E_AHI_SI_NO_STOP_BIT,
            E_AHI_SI_NO_SLAVE_READ, E_AHI_SI_SLAVE_WRITE,
            E_AHI_SI_SEND_ACK, E_AHI_SI_NO_IRQ_ACK);

        while(bAHI_SiPollTransferInProgress()); /* busy wait */

        if (bAHI_SiPollArbitrationLost() | bAHI_SiPollRxNack())
        {
            /* release bus & abort */
            vAHI_SiSetCmdReg(E_AHI_SI_NO_START_BIT, E_AHI_SI_STOP_BIT,
                E_AHI_SI_NO_SLAVE_READ, E_AHI_SI_SLAVE_WRITE,
                E_AHI_SI_SEND_ACK, E_AHI_SI_NO_IRQ_ACK);

            return FALSE;
        }
        if (i < u32Length - 1)
        {
            /* read and ack */
            vAHI_SiSetCmdReg(E_AHI_SI_NO_START_BIT, E_AHI_SI_NO_STOP_BIT,
                E_AHI_SI_SLAVE_READ, E_AHI_SI_NO_SLAVE_WRITE,
                E_AHI_SI_SEND_ACK, E_AHI_SI_NO_IRQ_ACK);
        }
        else /* last byte */
        {
            /* read, stop, nack */
            vAHI_SiSetCmdReg(E_AHI_SI_NO_START_BIT, E_AHI_SI_STOP_BIT,
                E_AHI_SI_SLAVE_READ, E_AHI_SI_NO_SLAVE_WRITE,
                E_AHI_SI_SEND_NACK, E_AHI_SI_NO_IRQ_ACK);
        }

        while(bAHI_SiPollTransferInProgress()); /* busy wait */

        if (bAHI_SiPollArbitrationLost())
        {
            /* release bus & abort */
            vAHI_SiSetCmdReg(E_AHI_SI_NO_START_BIT, E_AHI_SI_STOP_BIT,
                E_AHI_SI_NO_SLAVE_READ, E_AHI_SI_NO_SLAVE_WRITE,
                E_AHI_SI_SEND_ACK, E_AHI_SI_NO_IRQ_ACK);

            return FALSE;
        }

        /* Store data read from device */
        pau8ReadBuffer[i] = u8AHI_SiReadData8();
    }
    /* transfer complete */
    vAHI_SiSetCmdReg(E_AHI_SI_NO_START_BIT, E_AHI_SI_STOP_BIT,
        E_AHI_SI_NO_SLAVE_READ, E_AHI_SI_NO_SLAVE_WRITE,
        E_AHI_SI_SEND_ACK, E_AHI_SI_NO_IRQ_ACK);

    return TRUE;
}

```

15 Analogue Peripherals

The JN513x contains a number of analogue peripherals allowing the direct connection of a wide range of external sensors, switches and actuators.

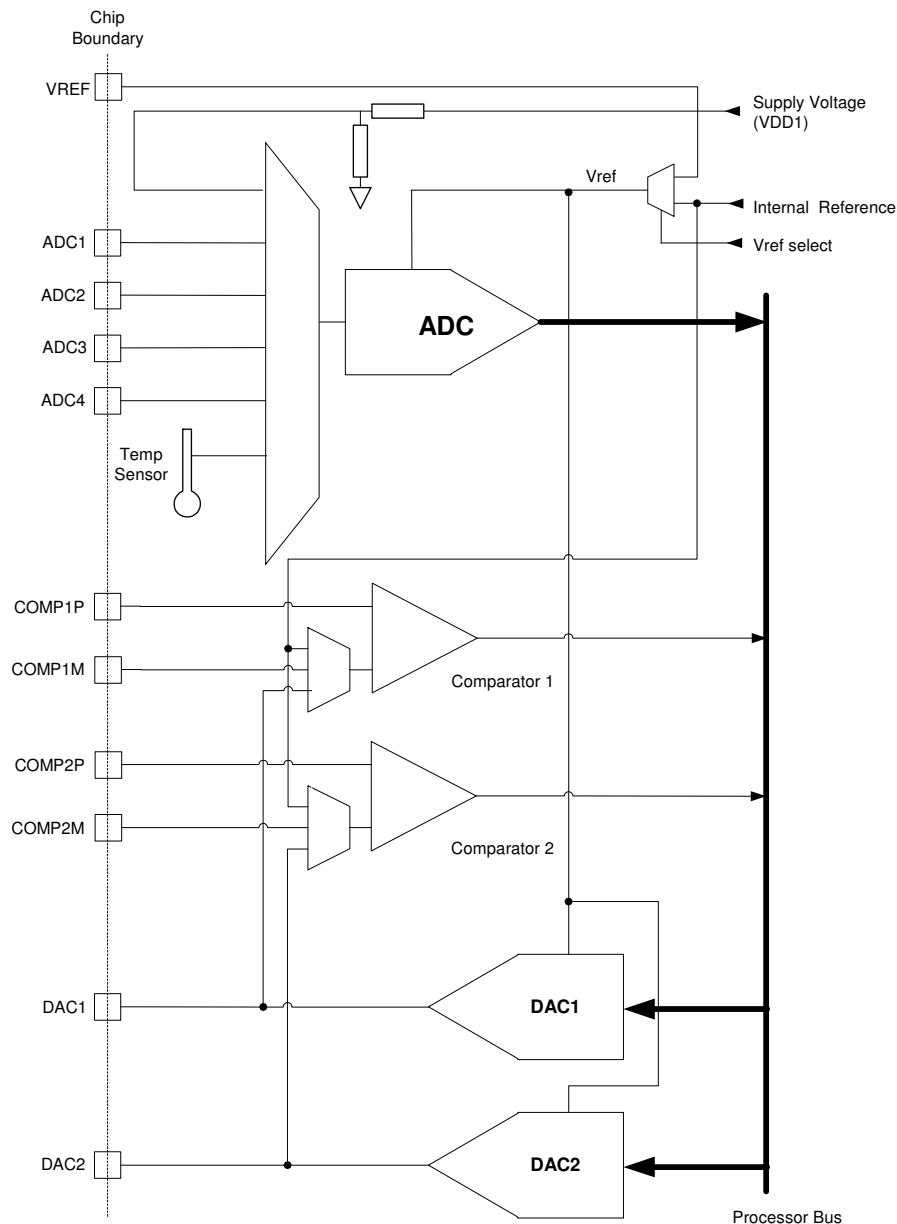


Figure 37: On-chip Analogue Peripherals

In order to provide good isolation from digital noise, the analogue peripherals are powered by a separate regulator, supplied from the analogue supply VDD1 and referenced to analogue ground VSSA.

The ADC and DAC reference Vref can be selected by `vAHI_ApConfigure()` between an internal bandgap reference or an external voltage reference supplied to the VREF pin.

15.1 Analogue to Digital Converter

The 12-bit analogue to digital converter (ADC) uses a successive approximation design to perform high accuracy conversions as typically required in wireless sensor network applications. It has six multiplexed single-ended input channels: four available externally, one connected to an internal temperature sensor, and one connected to an internal supply monitoring circuit.

15.1.1 Operation

The input range of the ADC can be set between 0V to either the reference voltage or twice the reference voltage. The reference can be either taken from the internal voltage reference or from the external voltage applied to the VREF pin. For example, an external reference of 1.2V supplied to VREF may be used to set the ADC range between 0V and 2.4V.

VREF	Gain Setting	Maximum Input Range	Supply Voltage Range (VDD)
1.2V	0	1.2V	2.2V - 3.6V
1.6V	0	1.6V	2.2V - 3.6V
1.2V	1	2.4V	2.6V - 3.6V
1.6V	1	3.2V	3.4V - 3.6V

The input clock to the ADC is 16MHz and is divided down to 500kHz. During an ADC conversion the selected input channel is sampled for a fixed period and then held. This sampling period is defined as a number of ADC clock periods and can be programmed to 2, 4, 6 or 8. The conversion rate is $(3 \times \text{sampling interval}) + (14 \times \text{Clock periods})$, for example if the sampling period is set to 2 clock periods, with the 500kHz clock the conversion rate will be $3 \times 2 + 14 = 20$ clock periods, 40µsecs or 25kHz.

If the source resistance of the input voltage is 1kΩ or less, then the default sampling time of 2 clocks should be used. The input to the ADC can be modelled as a resistor of 5kΩ to represent the on-resistance of the switches and the sampling capacitor 8pF. The sampling time required can then be calculated, by adding the sensor source resistance to the switch resistance, multiplying by the capacitance giving a time constant. Assuming normal exponential RC charging, the number of time constants required to give an acceptable error can be calculated, 7 time constants gives an error of 0.1%, so for 12-bit accuracy 10 time constants should be the target. For a source with zero resistance, 10 time constants is 800 nsecs, hence the smallest sampling window of 2 clock periods can be used.

The ADC sampling period is set with `vAHI_ApConfigure()`. The ADC input range and input is selected and the ADC enabled in either single shot mode with `vAHI_AdcStartSample()` or continuous mode using `vAHI_AdcEnable()`.

When the ADC conversion is complete, an interrupt is generated. This is enabled using `vAHI_ApConfigure()`. Alternatively the conversion status can be monitored using `bAHI_AdcPoll()`. When operating in continuous mode, it is recommended that the interrupt is used to signal the end of a conversion, since conversion times may range from 36 to 60 µsecs. Polling over this period would be wasteful of processor bandwidth. The result of a conversion can be read using `u16AHI_AdcRead()` function.

The ADC also has an accumulation feature that allows the results of several samples to be accumulated with no CPU intervention and once completed an interrupt generated.

For detailed electrical specifications, see section 17.3.7.

15.1.2 Supply Monitor

The internal supply monitor allows the voltage on the analogue supply pin VDD1 to be measured. This is achieved with a potential divider that reduces the voltage by a factor of 0.666, allowing it to fall inside the input range of the ADC when set with an input range twice the internal voltage reference. The resistor chain that performs the voltage reduction is disabled until the measurement is made to avoid a continuous drain on the supply.

15.1.3 Temperature Sensor

The on-chip temperature sensor can be used either to provide an absolute measure of the device temperature or to detect changes in the ambient temperature. In common with most on-chip temperature sensors, it is not trimmed and so the absolute accuracy variation is large; the user may wish to calibrate the sensor prior to use. The sensor forces a constant current through a forward biased diode to provide a voltage output proportional to the chip die temperature which can then be measured using the ADC. The measured voltage has a linear relationship to temperature as described in section 17.3.11.

Because this sensor is on-chip, any measurements taken must account for the thermal time constants. For example if the device just came out of sleep mode the user application should wait until the temperature has stabilized before taking a measurement.

15.1.4 Programming Example

The following example illustrates data logging using the ADC1 input channel.

Programming Example

```
PRIVATE void vAdcDataLogger(uint16 *paul6DataBuffer, uint32 u32Length)
{
    int i;

    /* configure Analogue Peripheral timings, interrupt & ref voltage */
    vAHI_ApConfigure(    E_AHI_AP_REGULATOR_ENABLE,
                        E_AHI_AP_INT_DISABLE,
                        E_AHI_AP_SAMPLE_2,
                        E_AHI_AP_CLOCKDIV_500KHZ,
                        E_AHI_AP_INTREF);

    while (!bAHI_APRegulatorEnabled);

    /* configure & enable DAC */
    vAHI_AdcEnable(E_AHI_ADC_CONVERT_ENABLE,
                  E_AHI_AP_INPUT_RANGE_1,
                  E_AHI_ADC_SRC_ADC_1);

    while(TRUE)
    {
        for (i=0;i<u32Length;i++)
        {
            vAHI_AdcStartSample();
            /* start capture */
            while(bAHI_AdcPoll());
            /* busy wait until capture complete */
            paul6DataBuffer[i] = u16AHI_AdcRead();
            /* store in buffer */
        }
    }
}
```

15.2 Digital to Analogue Converter

The Digital to Analogue Converter (DAC) provides two output channels and is capable of producing voltages of 0 to Vref or 0 to 2Vref where Vref is selected between the internal reference and the VREF pin, with a resolution of 11 bits and a minimum conversion time of 80µsecs (250kHz clock).

15.2.1 Operation

The output range of each DAC can be set independently to swing between 0V to either the reference voltage or twice the reference voltage. The reference voltage is selected from the internal reference or the VREF pin. For example, an external reference of 0.8V supplied to VREF may be used to set DAC1 maximum output of 0.8V and DAC2 maximum output of 1.6V.

The DAC output amplifier is capable of driving a capacitive load up to that specified in section 17.3.9.

Programmable clock periods set with `vAHI_ApConfigure()` allow a trade-off between conversion speed and resolution. The full 11-bit resolution is achieved with the 250kHz clock rate. See section 17.3.7, electrical characteristics, for more details.

The conversion period of the DACs are given by the same formula as the ADC conversion time and so can vary between 72 and 120uS. The DAC values may be updated at the same time as the ADC is active.

The clock divider ratio, interrupt enable and reference voltage select are all controlled by the `vAHI_ApConfigure()` function which is for options common to both the ADC and DAC. The DAC output range and value is set with `vAHI_DacEnable()` and subsequent updates may use `vAHI_DacOutput()`, which only requires the new DAC value. The call `bAHI_DacPoll()` can be used to determine if a DAC channel is busy performing a conversion. The `vAHI_DacDisable()` function is used to power down a DAC cell.

15.2.2 Programming Example

The following code example illustrates how to generate a sawtooth waveform on pin 29 (DAC1)

Programming Example

```
PRIVATE void vDacSawtooth(void)
{
    uint16 u16InitalValue = 0;
    int i;

    /* configure Analogue Peripheral timings, interrupt & ref voltage */
    vAHI_ApConfigure(
        E_AHI_AP_REGULATOR_ENABLE,
        E_AHI_AP_INT_DISABLE,
        E_AHI_AP_SAMPLE_2,
        E_AHI_AP_CLOCKDIV_2MHZ,
        E_AHI_AP_INTREF);

    while (!bAHI_APRegulatorEnabled);

    /* configure & enable DAC */
    vAHI_DacEnable(E_AHI_AP_DAC_1,
        E_AHI_AP_INPUT_RANGE_1,
        E_AHI_DAC_RETAIN_DISABLE,
        u16InitalValue);

    while(TRUE)
    {
        for (i=0;i<2048;i++)
        {
            /* value to output */
            vAHI_DacOutput(E_AHI_DAC_1, i);

            /* wait until conversion completes */
            while(bAHI_DacPoll());
        }
    }
}
```

15.3 Comparators

The JN513x contains two analogue comparators COMP1 and COMP2 that are designed to have true rail-to-rail inputs and operate over the full voltage range of the analogue supply VDD1. The hysteresis level (common to both comparators) can be set to a nominal value of 0mV, 5mV, 10mV or 20mV using the `vAHI_ComparatorEnable()` function. In addition, the source of the negative input signal for each comparator (COMP1M and COMP2M) can be set to one of the internal voltage reference, the output of DAC1 or DAC2 (COMP1 or COMP2 respectively) or the external pin, using `vAHI_ComparatorEnable()`. The comparator outputs are routed to internal registers and can be polled using the `u8AHI_ComparatorStatus()` function, or can be used to generate interrupts controlled by

`vAHI_ComparatorIntEnable()`. The comparators can be disabled using the `vAHI_ComparatorDisable()` function to reduce power consumption.

The comparators have a low power mode where the response time of the comparator is slower than normal and is specified in section 17.3.10. This mode may be used during non-sleep operation however it is particularly useful in sleep mode to wake up the JN513x from sleep where low current consumption is important. The function `vAHI_ComparatorIntEnable()` enables the wakeup action and sets which edge of the comparator output will be active. In sleep mode the negative input signal source defaults to the external pins.

16 Power Management and Sleep Modes

16.1 Operating Modes

Three operating modes are provided in the JN513x that enable the system power consumption to be controlled carefully to maximise battery life.

- Active Processing Mode
- Sleep Mode
- Deep Sleep Mode

The variation in power consumption of the three modes is a result of having a series of power domains within the chip that may be controllably powered on or off.

16.1.1 Power Domains

The JN513x has the following power domains:

- VDD Supply Domain: supplies the wake-up timers and controller, DIO blocks, Comparators and 32kHz RC oscillator. This domain is driven from the external supply (battery) and is always powered. The wake-up timers and controller, and the 32kHz RC oscillator may be powered on or off in sleep mode through software control.
- Digital Logic Domain: supplies the SPI interface, CPU, ROM, Baseband controller, Modem and Encryption processor. It is powered off during sleep mode.
- Analogue Domain: supplies the ADC, DACs and the temperature sensor. It is powered off during sleep mode and may be powered on or off in active processing mode through software control.
- RAM Domain: supplies the RAM during sleep mode to retain the memory contents. It may be powered on or off for sleep mode through software control.
- Radio Domain: supplies the radio interface. It is powered during transmit and receive and controlled by the baseband processor.

16.2 Active Processing Mode

Active processing mode in the JN513x is where all of the application processing takes place. By default, the CPU will execute in full speed mode allowing 16/32MIPs performance to be achieved. All of the peripherals are available to the application as are options to actively enable or disable them to control power consumption; see specific peripheral sections for details.

Whilst in Active processing mode there is the option to doze the CPU but keep the rest of the chip active, this is particularly useful for radio transmit and receive operations, where the CPU operation is not required.

16.2.1 CPU Doze

Whilst in doze mode, CPU operation is stopped but the chip remains powered and the digital peripherals continue to run. Doze mode is entered by executing the `vAHI_CpuDoze()` function and is terminated by any interrupt request. Once the interrupt service routine has been executed, the `vAHI_CpuDoze()` function returns and normal program execution resumes. Doze mode uses more power than sleep and deep sleep modes but requires less time to restart and can therefore be used as a low power alternative to an idle loop.

16.3 Sleep Mode

The JN513x enters sleep mode under control of the CPU using the `vAHI_PowerDown()` function. In this mode many of the internal chip functions are shutdown to save power, however the state of DIO pins are retained, including

the output values, and this therefore preserves any interface to the outside world. The DAC outputs are placed into a high impedance state.

When entering into sleep mode, there is an option to retain the RAM contents throughout the sleep period, this is determined by `vAHI_MemoryHold()`. If wakeup timers or comparator event are not to be used for a wakeup event, then power can be saved by switching off the 32kHz oscillator through software control.

Whilst in sleep mode one of three possible events can cause a wakeup to occur, transitions on DIO inputs, expiry of wakeup timers or comparator events. If any of these events occur, an interrupt is generated that will cause a wakeup from sleep. It is possible for multiple wakeup sources to trigger an event at the same instant and only one of them will be accountable for the wakeup period. It is therefore necessary in software to remove all other pending wakeup events prior to requesting entry back into sleep mode; otherwise, the device will re-awaken immediately.

When wakeup occurs, a similar sequence of events to the reset process described in section 6.1 happens. The 16MHz oscillator is started up, once stable the power to CPU system is enabled and the reset is removed. Software determines that this is a reset from sleep and so commences with the wakeup process.

16.3.1 Wakeup Timer Event

The JN513x contains two 32-bit wakeup timers that are counters clocked from the 32kHz oscillator, and can be programmed to generate a wake-up event. These timers are described in section 12.3.

Timer events can be generated from both of the two timers; one is intended for use by the 802.15.4 protocol, the other being available for use by the Application running on the CPU. These timers are available to run at any time, even during sleep mode, and are controlled by API calls as detailed in the Jennic document JN513x JN-RM-2001 Integrated Peripherals API [2].

16.3.2 DIO Event

Any DIO pin when used as an input has the capability, by detecting a transition, to generate a wake-up event. Once this feature has been enabled using the `vAHI_DioInterruptEnable()` function the type of transition can be specified (rising or falling edge) by using the `vAHI_DioInterruptEdge()` function. Even when groups of DIO lines are configured as alternative functions such as the UARTs or Timers etc, any input line in the group can still be used to provide a wakeup event. This means that an external device communicating over the UART can wakeup a sleeping device by asserting its RTS signal pin.

16.3.3 Comparator Event

The comparator can generate a wakeup interrupt when a change in the relative levels of the positive and negative inputs occurs, the negative input being selectable between the external pin COMPxN or the internal voltage reference. The ability to wakeup when continuously monitoring analogue signals is useful in ultra-low power applications. The JN513x can remain in sleep mode until the voltage drops below a threshold and then be woken up to deal with the alarm condition.

16.4 Deep Sleep Mode

Deep sleep mode gives the lowest power consumption. All switchable power domains are off and certain functions in the VDD supply power domain, including the 32kHz oscillator are stopped. It is entered by executing the `vAHI_PowerDown()` function. This mode can be exited by a power down, a hardware reset on the RESETN pin, or a DIO event. The DIO event in this mode causes a chip reset to occur.

17 Electrical Characteristics

17.1 Maximum ratings

Exceeding these conditions may result in damage to the device.

Parameter		Min	Max
Device supply voltage VDD1, VDD2		-0.3V	3.6V
Supply voltage at voltage regulator bypass pins VB_XXX		-0.3V	1.98V
Voltage on analogue pins XTALOUT, XTALIN, VCOTUNE, RFP, RFM,		-0.3V	VB_XXX + 0.3V
Voltage on analogue pins VREF, ADC1-4, DAC1-2, COMP1M, COMP1P, COMP2M, COMP2P, IBIAS		-0.3V	VDD1 + 0.3V
Voltage on 5v tolerant digital pins SPICLK, SPIMOSI, SPIMISO, SPISEL0, GPIO0-GPIO20, RESETN		-0.3V	Lower of (VDD2 + 2V) and 5.5V
Storage temperature		-40°C	150°C
Reflow soldering temperature according to IPC/JEDEC J-STD-020C			260°C
ESD rating (see note 1)	Human Body Model		1.5kV
	Machine Model		150V

Note 1: The Human body model is a 100pF capacitor discharged through a 1.5kΩ resistor into each pin. (MIL-STD-883 3015.7) The machine model is a 200pF capacitor discharged directly into each pin.

17.2 DC Electrical Characteristics

17.2.1 Operating Conditions

Supply	Min	Max
VDD1, VDD2	2.2V	3.6V
Ambient temperature range	-40°C	85°C

17.2.2 DC Current Consumption

VDD = 2.2 to 3.6V, -40 to +85° C

17.2.2.1 Active Processing

Mode:	Min	Typ	Max	Unit	Notes
CPU processing		4250 + 310/MHz		μA	SPI, GPIOs enabled
CPU processing (2 x clock)		TBD		μA	SPI, GPIOs enabled
Radio transmit [boost mode]		39 [43.5]		mA	CPU in software doze – radio transmitting
Radio receive [boost mode]		39 [42]		mA	CPU in software doze – radio in receive mode
The following current figures should be added to those above if the feature is being used					
ADC		580		μA	Temperature sensor and battery measurements require ADC
DAC		220 / 250		μA	One / both
Comparator		67 / 1.2		μA	Fast response time / low-power
UART		310		μA	For each UART
Timer		45		μA	For each Timer
2-wire serial interface		86		μA	

17.2.2.2 Sleep Mode

Mode:	Min	Typ	Max	Unit	Notes
Sleep mode with I/O wakeup		0.2		μA	Waiting on I/O event.
Sleep mode with I/O and timer wakeup		1.3		μA	As above, but also waiting on timer event. If both wakeup timers are enabled then add another 0.3μA
The following current figures should be added to those above if the feature is being used					
RAM retention		2.0		μA	For full 96kB retained.
Comparator (low-power mode)		1.2		μA	Reduced response time.

17.2.2.3 Deep Sleep Mode

Mode:	Min	Typ	Max	Unit	Notes
Deep sleep mode		0.2	0.4	μA	Waiting on chip RESET or I/O event.

17.2.3 I/O Characteristics

VDD = 2.2 to 3.6V, -40 to +85° C

Parameter	Min	Typ	Max	Unit	Notes
Internal DIO pull – up resistors	24 27 38	35 42 59	53 63 92	kΩ	VDD2 = 3.6V, 25C VDD2 = 3.0V, 25C VDD2 = 2.2V, 25C
Digital I/O High Input	VDD2 x 0.7		Lower of (VDD2 + 2V) and 5.5V	V	5V Tolerant
Digital I/O low Input	-0.3		VDD x 0.27	V	
Digital I/O input hysteresis	0.175		0.4V	V	
DIO High O/P (2.7-3.6V)	VDD2 x 0.8		VDD2	V	With 3mA load
DIO Low O/P (2.7-3.6V)	0		0.4V	V	With 3mA load
DIO High O/P (2.2-2.7V)	VDD2 x 0.8		VDD2	V	With 4mA load
DIO Low O/P (2.2-2.7V)	0		0.4V	V	With 4mA load

17.3 AC Characteristics

17.3.1 Reset

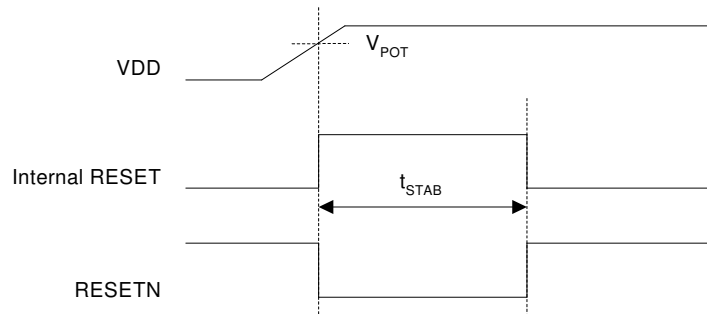


Figure 38: Power-on Reset

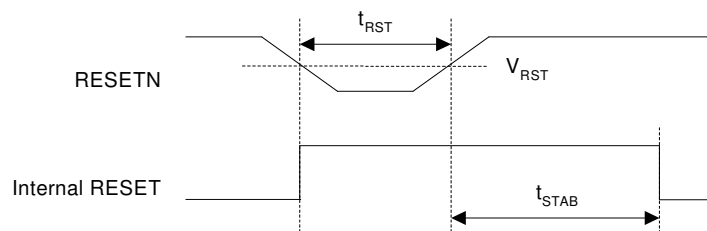


Figure 39: External Reset

Parameter	Min	Typ	Max	Unit	Notes
External Reset pulse width	1			μs	Assumes internal pullup resistor value of 100K worst case and ~5pF external capacitance .
External Reset threshold voltage	$V_{DD2} \times 0.7$			V	
Internal Power-on Reset threshold voltage (V_{POT})		2.0 2.15 2.25		V	$V_{DD2} = 2.2\text{V}$ $V_{DD2} = 3.0\text{V}$ $V_{DD2} = 3.6\text{V}$ Note 1

¹ The power-on reset will not operate unless VDD has fallen below V_{POT} (falling)

17.3.2 Brown-out Detect

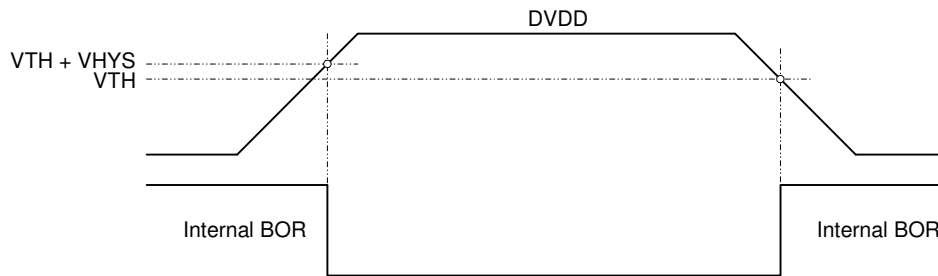


Figure 40: Brown-out Detect

Parameter	Min	Typ	Max	Unit	Notes
Brown-out Threshold Voltage (V_{TH})		2.1 2.4 2.5 2.6		V	Configurable threshold with 4 levels
Brown-out Hysteresis (V_{HYS})		100		mV	

17.3.3 SPI Timing

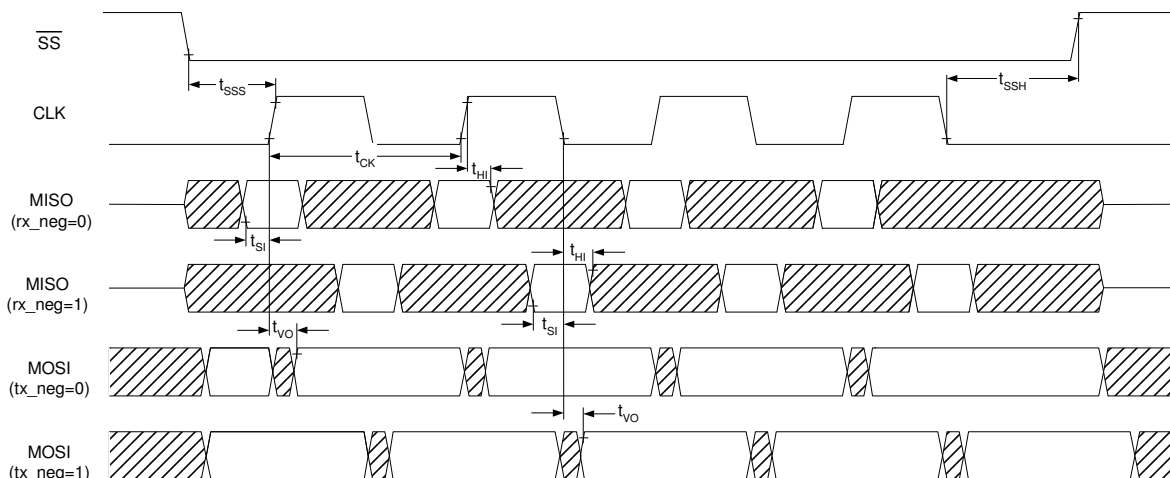


Figure 41: SPI Timing (Master)

Parameter	Symbol	Min	Max	Unit
Clock period	t_{CK}	62.5	-	ns
Data setup time	t_{SI}	5	-	ns
Data hold time	t_{HI}	10	-	ns
Data invalid period	t_{VO}	-	15	ns
Select set-up period	t_{SSS}	10	-	ns
Select hold period	t_{SSH}	10	-	ns

17.3.4 Two-wire serial interface

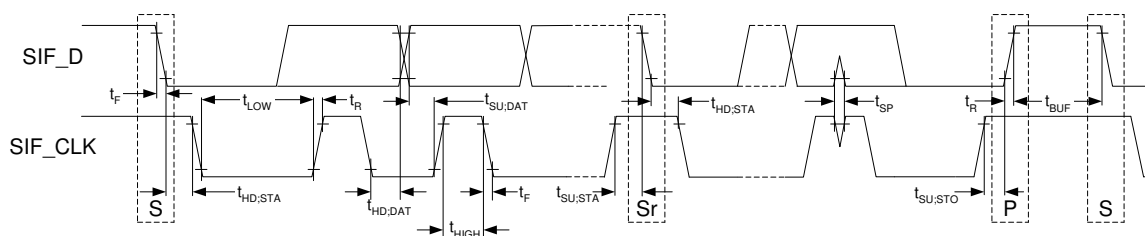


Figure 42: Two-wire serial Interface Timing

Parameter	Symbol	Min	Max	Unit
SIF_CLK clock frequency	f_{SCL}	0	400	kHz
Hold time (repeated) START condition. After this period, the first clock pulse is generated	$t_{HD,STA}$	0.6	-	μ s
LOW period of the SIF_CLK clock	t_{LOW}	1.3	-	μ s
HIGH period of the SIF_CLK clock	t_{HIGH}	0.6	-	μ s
Set-up time for repeated START condition	$t_{SU,STA}$	0.6	-	μ s
Data hold time SIF_D	$t_{HD,DAT}$	0	0.9	μ s
Data setup time SIF_D	$t_{SU,DAT}$	0.1	0	μ s

Rise Time SIF_D and SIF_CLK	t_R	$20+0.1C_b$	300	ns
Fall Time SIF_D and SIF_CLK	t_F	$20+0.1C_b$	300	ns
Set-up time for STOP condition	$t_{SU:STO}$	0.6	-	μs
Bus free time between a STOP and START condition	t_{BUF}	1.3	-	μs
Capacitive load for each bus line	C_b	-	400	pF
Noise margin at the LOW level for each connected device (including hysteresis)	V_{nl}	0.1VDD	-	V
Noise margin at the HIGH level for each connected device (including hysteresis)	V_{nh}	0.2VDD	-	V
Pulse width of spikes which must be suppressed by input filter	t_{SP}	N/a	50	ns

17.3.5 Power Down and Wake-Up timings

Parameter	Min	Typ	Max	Unit	Notes
Wake up from Deep Sleep		$2.5 + 0.84^*$ program size in kBytes		ms	Assumes SPI clock to external Flash is 16MHz
Wake up from Sleep (memory not held)		$2.5 + 0.84^*$ program size in kBytes		ms	Assumes SPI clock to external Flash is 16MHz
Wake up from Sleep (Memory held)		2.5		ms	
Wake up from CPU Doze mode		0.2		μs	

17.3.6 32kHz Oscillator

VDD = 2.2 to 3.6V, -40 to +85 °C

Parameter	Min	Typ	Max	Unit	Notes
Current consumption of cell and counter logic		1.05 0.9 0.82		μA	3.3V 2.7V 2.2v
32kHz clock native accuracy	-30%	32kHz	+30%		At 3.0V 25°C
Calibrated 32kHz accuracy		±65		ppm	
Variation with temperature		-0.02		%/°C	
Variation with VDD2		-2.5		%/V	

17.3.7 16MHz Crystal Oscillator

VDD = 2.2 to 3.6V, -40 to +85°C

Parameter	Min	Typ	Max	Unit	Notes
Current consumption		150		μA	Excluding bandgap ref.
Start – up time		3.0		ms	Assuming xtal with ESR of 40ohms and CL= 9pF External caps = 15pF (150mV pk-pk)
Input capacitance		1.4		pF	Bondpad and package
Transconductance		1.25		mA/V	
DC voltages, XTALIN, XTALOUT		400		mV	
External Capacitors		15		pF	Total external capacitance needs to be 2*CL, allowing for stray capacitance from chip, package and PCB

17.3.8 Analogue to Digital Converters

VDD = 3.0V, VREF = 1.2V, -40 to +85°C

Parameter	Min	Typ	Max	Unit	Notes
Resolution			12	bits	500kHz Clock
Current consumption		580		μA	
Integral nonlinearity		± 2		LSB	
Differential nonlinearity			± 1	LSB	Guaranteed monotonic

Parameter	Min	Typ	Max	Unit	Notes
Offset error		± 20		mV	
Gain error		± 20		mV	
Internal clock		500		kHz	16MHz input clock, ÷32
No. internal clock periods to sample input		(2, 4, 6 or 8) x 3			Programmable x 3
Conversion time	36			µs	500KHz Clock with sample period of 2
Input voltage range			0 to Vref or 0 to 2*Vref	V	Switchable
Vref (Internal)	1.15	1.2	1.25	V	Bandgap voltage
Vref (External)	1.15	1.2	1.6	V	Allowable range into VREF pin
Input capacitance		8		pF	In series with 5K ohms

17.3.9 Digital to Analogue Converters

VDD = 3.0V, VREF = 1.2V, -40 to +85°C

Parameter	Min	Typ	Max	Unit	Notes
Resolution		11		bits	
Current consumption		220 (single) 250 (both)		µA	
Integral nonlinearity		± 2		LSB	
Differential nonlinearity			± 1	LSB	Guaranteed monotonic
Offset error		-56		mV	
Gain error		15		mV	
Internal clock		250		kHz	16MHz input clock, ÷64
Output settling time to 0.5LSB		5		µs	With 10k ohms & 20pF load
Minimum Update time	72			µs	250KHz Clock with sample period of 2
Output voltage swing		0 to VREF or 0 to 2xVREF		V	Switchable
Vref (Internal)	1.15	1.2	1.25	V	Bandgap voltage
VREF (External)	0.8	1.2	1.6	V	Allowable range into VREF pin

Parameter	Min	Typ	Max	Unit	Notes
Resistive load	10kΩ				To ground
Capacitive load			20	pF	
Digital input coding	Binary				

17.3.10 Comparators

VDD = 2.2 to 3.6V -40 to +85°C

Parameter	Min	Typ	Max	Unit	Notes
Analogue response time (normal)		80	120	ns	+/- 250mV overdrive 10pF load
Total response time (normal) including delay to Interrupt controller			120 + 125	ns	Digital delay can be up to a max. of two 16MHz clock periods
Analogue response time (low power)		2.0		μs	+/- 250mV overdrive No digital delay
Hysteresis		10 20 40		mV	Programmable in 3 steps and zero.
Vref (Internal)	1.15	1.2	1.25	V	
Common Mode input range	0		Vdd	V	
Current (normal mode)		67		μA	
Current (low power mode)		1.2		μA	

17.3.11 Temperature Sensor

Parameter	Min	Typ	Max	Unit	Notes
Operating Range	-40	-	85	°C	
Sensor Gain	-1.55	-1.6	-1.63	mV/°C	
Accuracy	-	-	±10	°C	
Non-linearity	-	-	2.5	°C	
Output Voltage Range	620	750	845	mV	
Resolution	0.756	0.733	0.719	°C/LSB	0 to Vref ADC I/P Range

17.3.12 Radio Transceiver

This JN513x meets all the requirements of the IEEE802.15.4 standard over 2.2 - 3.6V and offers the following improved RF characteristics. All RF characteristics are measured single ended and include the losses of a ceramic balun.

Parameter	Min	Typical	Max	Notes
RF Port Characteristics				
Type				Differential
Impedance		200ohm		2.4-2.5GHz
Frequency range	2.4 GHz		2.4835GHz	

17.3.12.1 Radio parameters: 2.2-3.6V, +25°C

Parameter	Min	Typical	Max	Unit	Notes
Receiver Characteristics					
Receive sensitivity		-95.5		dBm	Nominal for 1% PER, as per 802.15.4 section 6.5.3.3
Receive sensitivity (boost)		-96.5		dBm	Nominal for 1% PER, as per 802.15.4 section 6.5.3.3
Maximum input signal		-10		dBm	For 1% PER, measured as sensitivity
Adjacent channel rejection -1 channel / +1 channel [CW Interferer]		31 / 33 [35 / 38]		DB	For 1% PER with wanted signal 3dB above sensitivity, as per 802.15.4 section 6.5.3.4 (modulated interferer)
Alternate channel rejection [CW Interferer]		41 [45]		DB	For 1% PER with wanted signal 3dB above sensitivity, as per 802.15.4 section 6.5.3.4 (modulated interferer)
Other in band rejection 2.4 to 2.4835 GHz, excluding adj channels		46		DB	For 1% PER with wanted signal 3dB above sensitivity, measured as per 802.15.4 section 6.5.3.4
Out of band rejection		>45		DB	
Spurious emissions (RX)			-57 -47	dBm	30MHz to 1GHz 1GHz to 12GHz
Intermodulation protection		40		DB	For 1% PER at with wanted signal 3dB above sensitivity. Modulated Interferers at 2 & 4 channel separation
RSSI linearity	-3		+3	DB	-95 to -10dBm. Available through Hardware API
Transmitter Characteristics					
Transmit power		0.5		dBm	Nominal
Transmit power (boost)		+2.5		dBm	
Output power control range		-30		DB	in 5 6dB steps
Spurious emissions (TX)			-36 -43 -47	dBm	30MHz to 1GHz, 1GHz to 12.5GHz, The following exceptions apply 1.8 to 1.9GHz & 5.15 to 5.3GHz
EVM		15	25	%	At maximum output power
Transmit Power Spectral Density		-48	-20	dBc	At greater than 3.5MHz offset, as per 802.15.4, section 6.5.3.1

17.3.12.2 Radio parameters: 2.2-3.6V, -40°C

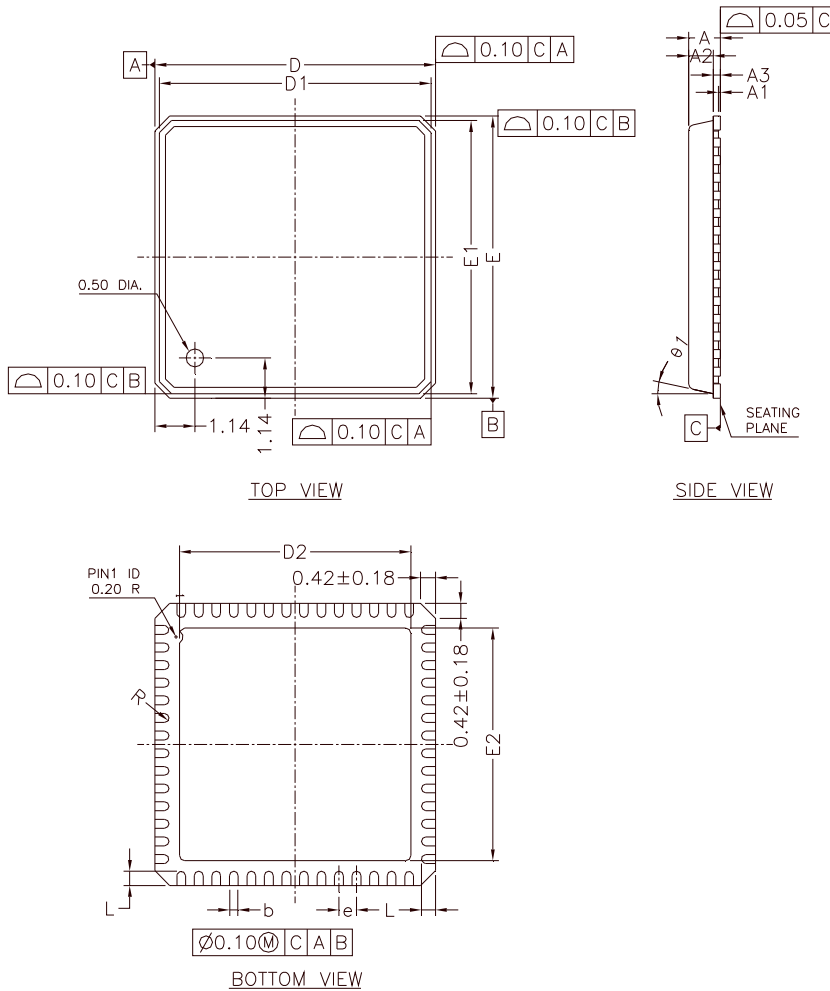
Parameter	Min	Typical	Max	Unit	Notes
Receiver Characteristics					
Receive sensitivity		-97		dBm	Nominal for 1% PER, as per 802.15.4 section 6.5.3.3
Maximum input signal		-10		dBm	For 1% PER, measured as sensitivity
Adjacent channel rejection -1 channel / +1 channel		19 / 29		dB	For 1% PER with wanted signal 3dB above sensitivity, as per 802.15.4 section 6.5.3.4
Alternate channel rejection		40		dB	For 1% PER with wanted signal 3dB above sensitivity, as per 802.15.4 section 6.5.3.4
Other in band rejection		45		dB	2.4 to 2.4835 GHz, excluding adjacent channels For 1% PER with wanted signal 3dB above sensitivity, measured as per 802.15.4 section 6.5.3.4
Out of band rejection		TBA			
Spurious emissions (RX)			-57 -47	dBm	30MHz to 1GHz 1 to 12GHz
Intermodulation protection		35		dB	For 1% PER at with wanted signal 3dB above sensitivity. Modulated Interferers at 2 & 4 channel separation
RSSI linearity	-3		+3	dB	-95 to -10dBm. Available through Hardware API
Transmitter Characteristics					
Transmit power		1.8		dBm	Nominal
Transmit power (boost)		+3.0		dBm	
Output power control range		-30		dB	in 5 6dB steps
Spurious emissions (TX)			-36 -43 -47	dBm	30MHz to 1GHz, 1GHz to 12.5GHz, The following exceptions apply 1.8 to 1.9GHz & 5.15 to 5.3GHz
EVM		20	25	%	At maximum output power
Transmit Power Spectral Density		-50	-20	dBc	At greater than 3.5MHz offset, as per 802.15.4, section 6.5.3.1

17.3.12.3 Radio parameters: 2.2-3.6V, +85°C

Parameter	Min	Typical	Max	Unit	Notes
Receiver Characteristics					
Receive sensitivity		-92		dBm	Nominal for 1% PER, as per 802.15.4 section 6.5.3.3
Maximum input signal		-10		dBm	For 1% PER, measured as sensitivity
Adjacent channel rejection -1 channel / +1 channel		19 / 29		dB	For 1% PER with wanted signal 3dB above sensitivity, as per 802.15.4 section 6.5.3.4
Alternate channel rejection		39		dB	For 1% PER with wanted signal 3dB above sensitivity, as per 802.15.4 section 6.5.3.4
Other in band rejection		43		dB	2.4 to 2.4835 GHz, excluding adjacent channels For 1% PER with wanted signal 3dB above sensitivity, measured as per 802.15.4 section 6.5.3.4
Out of band rejection		TBA			
Spurious emissions (RX)			-57 -47	dBm	30MHz to 1GHz 1GHz to 12GHz
Intermodulation protection		35		dB	For 1% PER at with wanted signal 3dB above sensitivity. Modulated Interferers at 2 & 4 channel separation
RSSI linearity	-3		+3	dB	-95 to -10dBm. Available through Hardware API
Transmitter Characteristics					
Transmit power		-3.0		dBm	Nominal
Transmit power (boost)		TBD		dBm	
Output power control range		-30		dB	in 5 steps of 6dB
Spurious emissions (TX)			-36 -43 -47	dBm	30MHz to 1GHz, 1GHz to 12.5GHz, The following exceptions apply 1.8 to 1.9GHz & 5.15 to 5.3GHz
EVM		12	25	%	At maximum output power
Transmit Power Spectral Density		-46	-20	dBc	At greater than 3.5MHz offset, as per 802.15.4, section 6.5.3.1

Appendix A Mechanical and Ordering Information

A.1 56pin QFN Package Drawing



Controlling Dimension: mm

Symbol	Millimeter		
	Min.	Nom.	Max.
A	-----	-----	0.9
A1	0.00	0.01	0.05
A2	-----	0.65	0.7
A3		0.20 Ref.	
b	0.2	0.25	0.3
D	8.00 bsc		
D1	7.75 bsc		
D2	6.20	6.40	6.60
E	8.00 bsc		
E1	7.75 bsc		
E2	6.20	6.40	6.60
L	0.30	0.40	0.50
e	0.50 bsc		
v1	0°	-----	12°
R	0.09	-----	-----
Tolerances of Form and Position			
aaa	0.10		
bbb	0.10		
ccc	0.05		

A.2 Ordering Information

Ordering Format:

JN513x - XXX - Y₁Y₂Y₃

Part Numbers:

JN5131	Wireless microcontroller - 8kB RAM
JN5132	Wireless microcontroller - 16kB RAM
JN5133	Wireless microcontroller - 32kB RAM
JN5139	Wireless microcontroller - 96kB RAM

XXX: ROM Variant:

001	IEEE802.15.4 stack
Z01	ZigBee stack

Y₁: Package Variant:

A	Punched 56 lead, 0.5mm pitch 8x8mm Quad Flat No Leads (QFN)
---	---

Y₂: Temperature Range / Device Status:

I	-40°C to +85°C - Industrial Temperature Range
---	---

Y₃: Shipping:

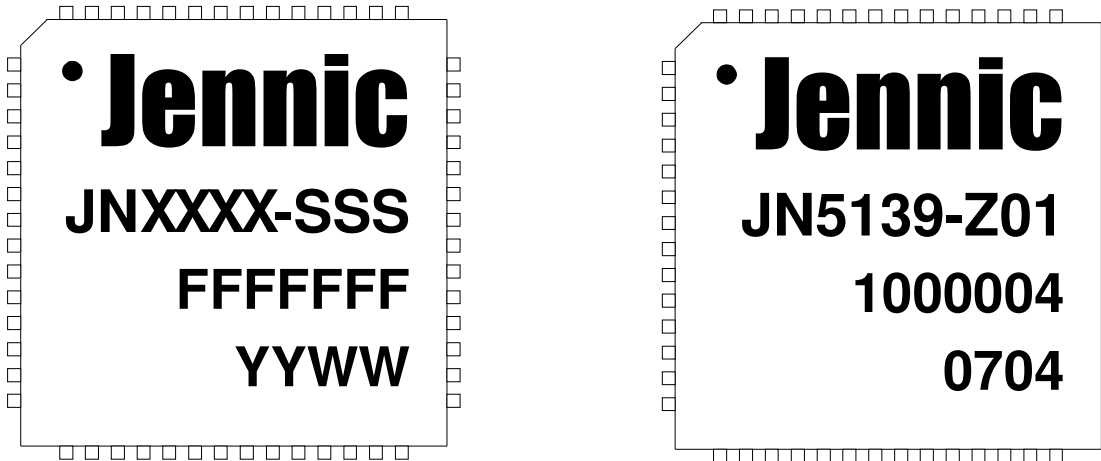
R	Trays (up to 260 devices)
T	Tape mounted 2500 devices on a 13" reel
V	Tape mounted 1000 devices on a 7" reel
X	Tape mounted 500 devices on a 7" reel
Y	Tape mounted upto 100 devices (no reel)

Ordering Examples:

Part Number	Description
JN5131-001-AIR	JN5131 IEEE802.15.4 Wireless Microcontroller – up to 260 devices in a tray
JN5133-Z01-AIV	JN5133 ZigBee Wireless Microcontroller - 1000 devices on a 7" reel

A.3 Device Package Marking

The diagram below shows the package markings for JN513x devices. The package on the left along with the legend information below it, shows the general format of package marking. The package on the right shows the specific markings for a JN5139-Z01 device, that came from assembly build number 1000004 and was manufactured week 4 of 2007.



Legend:

JN	Jennic
XXXX	4 digit part number, for example 5139, 5132
SSS	3 digit software ROM identifier
FFFFFFF	7 digit assembly build number
YY	2 digit year number
WW	2 digit week number

Where this Data Sheet is denoted as “Advanced” or “Preliminary”, devices will be either Engineering or Prototype Samples. Devices of this status have an R suffix after the software ROM identifier, for example JN5139-Z01R.

A.4 Tape and Reel Information

A.4.1 Tape Orientation and Dimensions

The general orientation of the 56QFN package in the tape is as shown in Figure 42.

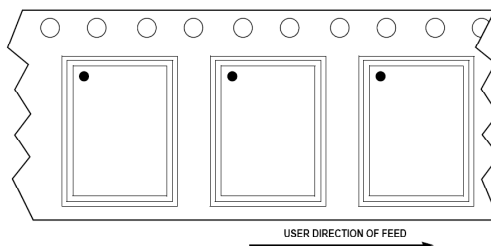
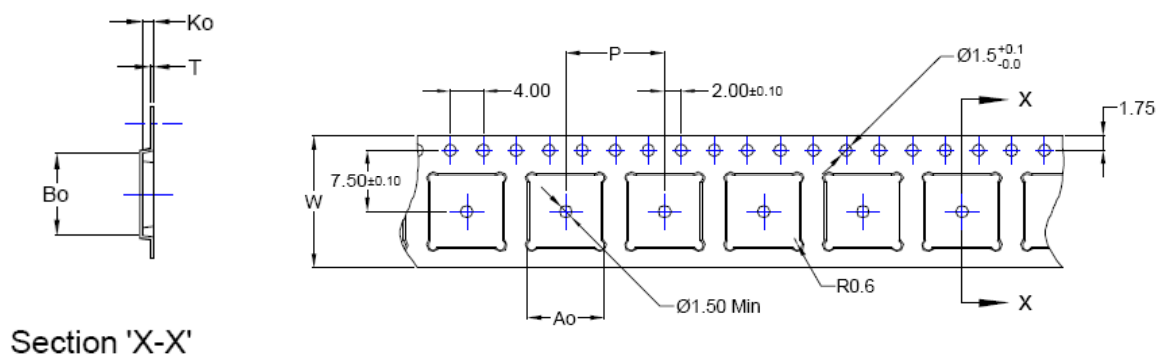


Figure 43: Tape and Reel orientation

Figure 43 shows the detailed dimensions of the tape used for 8x8mm 56QFN devices.



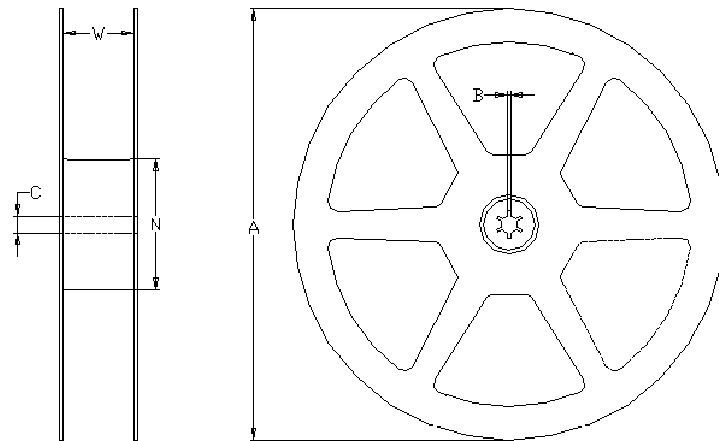
Reference	Dimensions (mm)
A ₀	8.30 ± 0.10
B ₀	8.30 ± 0.10
K ₀	1.10 ± 0.10
P	12.00 ± 0.10
T	0.30 ± 0.10
W	16.00 + 0.30/-0.10

Figure 44: Tape Dimensions

A.4.2 Reel Information: 7" Reel

Surface Resistivity	Between $10e^9$ – $10e^{11}$ Ohms Square
Material	High Impact Polystyrene, environmentally friendly, recyclable

All dimensions and tolerances are fully compliant with EIA-481-B and are specified in millimetres.
 6 window design with one window on each side blanked to allow adequate labelling space.



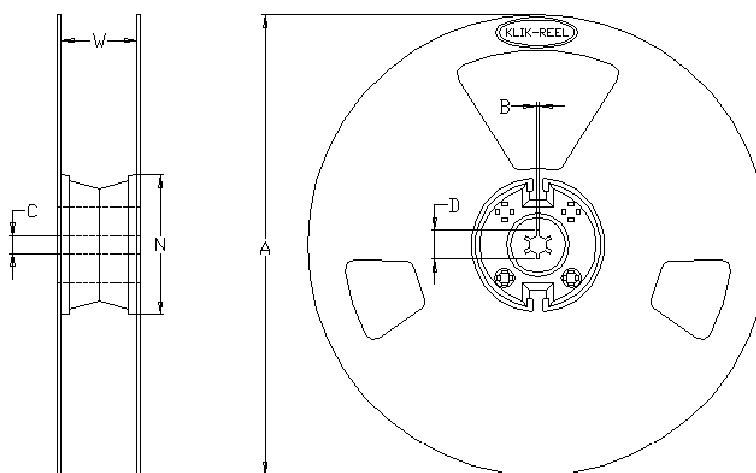
Tape Width	A	B (min)	C	N	W (min)	W (max)
16	180	1.5min	13 ± 0.2	$60 + 0.1 - 0.0$	16.40	17.90

Figure 45: Reel Dimensions

A.4.3 Reel Information: 13" Reel

Surface Resistivity	Between $10e^9 - 10e^{11}$ Ohms Square
Material	High Impact Polystyrene with Antistatic Additive

All dimensions and tolerances are fully compliant with EIA-481-B and are specified in millimetres.
 3 window design to allow adequate labelling space.



Tape Width	A	B (min)	C	D (min)	N (min)	W (min)	W (max)
16	330	1.5	13 +0.5 -0.2	20.2	100	15.90	19.40

Figure 46: Reel Dimensions

A.4.4 Dry Pack Requirement for Moisture Sensitive Material

Moisture sensitive material, as classified by JEDEC standard J-STD-033, must be dry packed. The 56 lead QFN package is MSL2A/260°C, and is dried before sealing in a moisture barrier bag (MBB) with desiccant bag weighing at 67.5 grams of activated clay and a 6 spot humidity indicator card (HIC) meeting MIL-L-8835 specification. The MBB has a moisture-sensitivity caution label to indicate the moisture-sensitive classification of the enclosed devices.

A.5 PCB Design and Reflow Profile

PCB and land pattern designs are key to board level reliability, and Jennic strongly recommends that users follow the design rules listed in IPC-SM-782. For reflow profiles, it is recommended to follow the reflow profile in Figure 47 as a guide, as well as the paste manufacturers guidelines on peak flow temperature, soak times, time above liquidus and ramp rates.

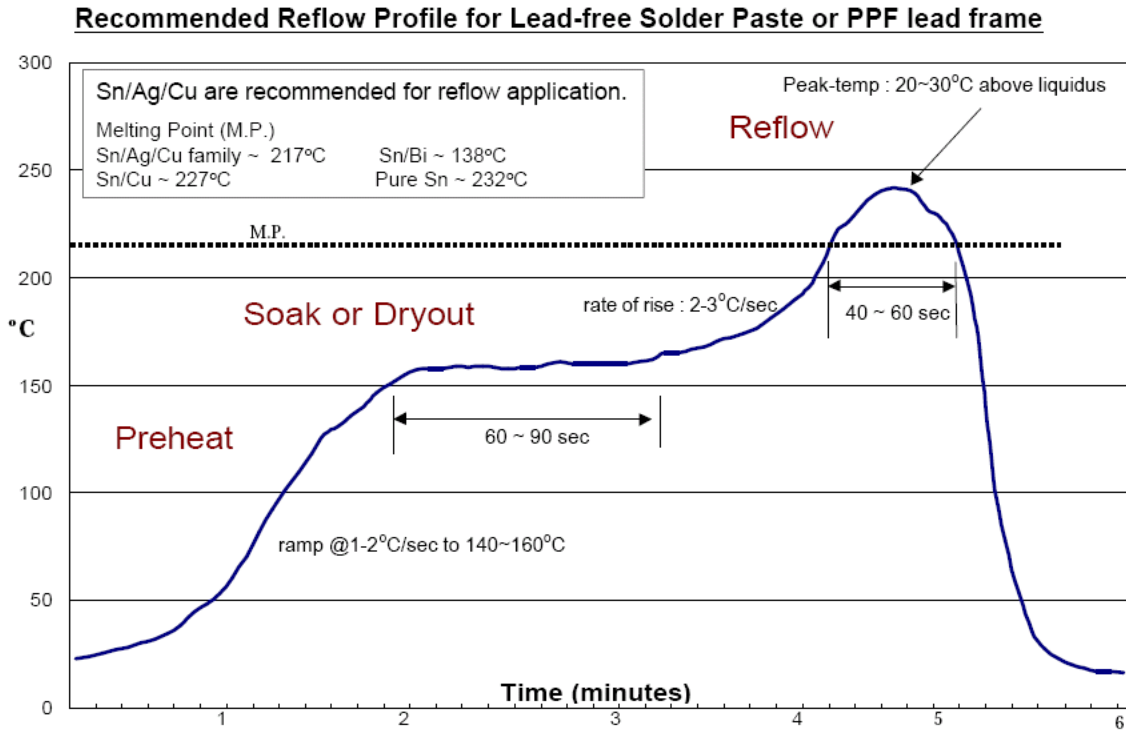


Figure 47: Reflow Profile

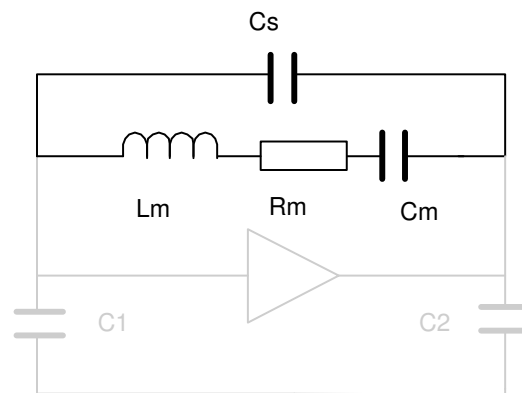
Appendix B Development Support

B.1 Crystal Oscillator

16MHz Crystal Requirements

Parameter	Min	Typ	Max	Notes
Crystal Frequency		16MHz		
Crystal Tolerance			40ppm	Including temperature and aging
Crystal ESR (R_m) ¹	20Ω		60Ω	See below for more details
Crystal Load Capacitance (CL)		9pF		See below for more details
External Capacitors (C1 & C2)		15pF		Total external capacitance needs to be $2 \cdot CL$, allowing for stray capacitance from chip, package and PCB

B.1.1 Crystal Equivalent Circuit



Where C_m is the motional capacitance

L_m is the motional inductance. This together with C_m defines the oscillation frequency (series)

R_m is the equivalent series resistance (ESR).

C_s is the shunt or package capacitance and this is a parasitic

B.1.2 Crystal Load Capacitance

The crystal load capacitance is the total capacitance seen at the crystal pins, from all sources. As the load capacitance (CL) affects the oscillation frequency by a process known as 'pulling', crystal manufacturers specify the frequency for a given load capacitance only. A typical pulling coefficient is 15ppm/pF, to put this into context the maximum frequency error in the IEEE802.15.4 specification is +/-40ppm for the transmitted signal. Therefore, it is important for resonance at 16MHz exactly, that the specified load capacitance is provided.

The load capacitance can be calculated using:

$$CL = \frac{C_{T1} \times C_{T2}}{C_{T1} + C_{T2}}$$

Total capacitance $C_{T1} = C_1 + C_{1P} + C_{1in}$

Where C_1 is the capacitor component

C_{1P} is the PCB parasitic capacitance. With the recommended layout this is about 1.6pF

C_{1in} is the on-chip parasitic capacitance and is about 1.4pF typically.

Similarly for C_{T2}

Hence for a 9pF load capacitance, and a tight layout the external capacitors should be 15pF

B.1.3 Crystal ESR and Required Transconductance

The resistor in the crystal equivalent circuit represents the energy lost. To maintain oscillation, power must be supplied by the amplifier, but how much? Firstly, the Pi connected capacitors C_1 and C_2 with C_S from the crystal, apply an impedance transformation to R_m , when viewed from the amplifier. This new value is given by:

$$\hat{R}_m = R_m \left(\frac{C_S + C_L}{C_L} \right)^2$$

The amplifier is a transconductance amplifier, which takes a voltage and produces an output current. The amplifier together with the capacitors C_1 and C_2 , form a circuit, which provides a negative resistance, when viewed from the crystal. The value of which is given by:

$$R_{NEG} = \frac{g_m}{C_{T1} \times C_{T2} \times \omega^2}$$

Where g_m is the transconductance

ω is the frequency in rad/s

Derivations of these formulas can be easily found in textbooks.

In order to give quick and reliable oscillator start-up, a common rule of thumb is to set the amplifier negative resistance to be a minimum of 4 times the effective crystal resistance. This gives

$$\frac{g_m}{C_{T1} \times C_{T2} \times \omega^2} \geq 4R_m \left(\frac{C_S + C_L}{C_L} \right)^2$$

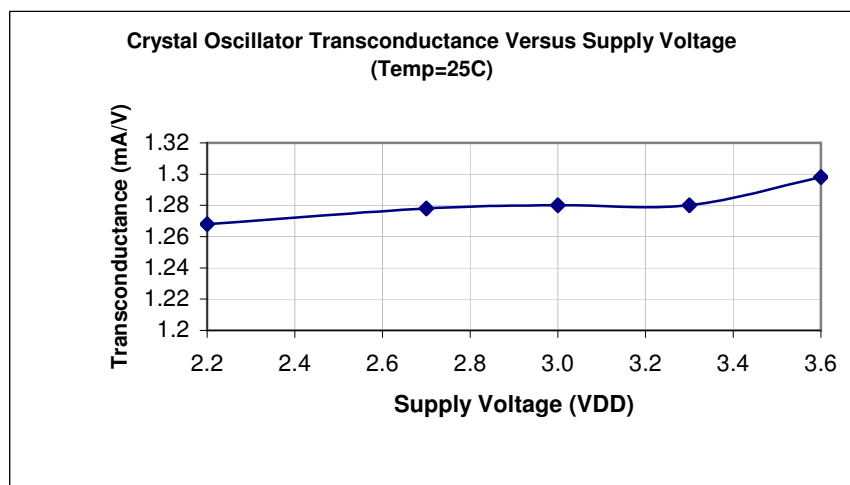
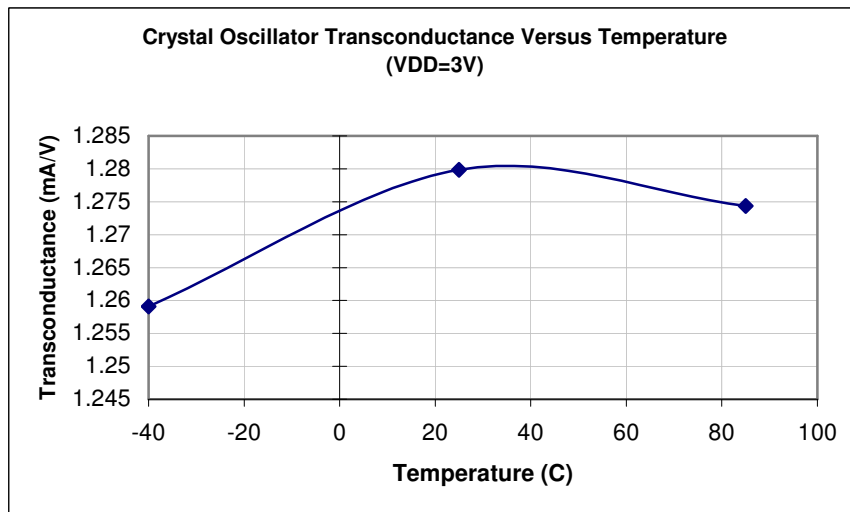
This can be used to give an equation for the required transconductance.

$$g_m \geq \frac{4R_m \times \omega^2 [C_s(C_{T1} + C_{T2}) + C_{T1} \times C_{T2}]^2}{C_{T1} \times C_{T2}}$$

Example: Using typical parameters of $R_m = 40\Omega$, $C_s = 1\text{pF}$ and $C_{T1} = C_{T2} = 18\text{pF}$ (for a load capacitance of 9pF), the equation above gives the required transconductance (g_m) as 647uA/V. The JN513x has a typical value for transconductance of 1.25mA/V

The example and equation illustrate the trade-off that exists between the load capacitance and crystal ESR. For example, a crystal with a higher load capacitance can be used, but the value of max. ESR that can be tolerated is reduced. Also note, that the circuit sensitivity to external capacitance [C_1, C_2] is a square law.

Below is measurement data showing the variation of the crystal oscillator amplifier transconductance with temperature and supply voltage, notice how small the variation is. Circuit techniques have been used to apply compensation, such that the user need only design for nominal conditions.



B.2 16MHz Oscillator

The JN513x contains the necessary on-chip components to build a 16 MHz reference oscillator with the addition of an external crystal resonator and two tuning capacitors. The schematic and layout of these components are shown in Figure 48. The two capacitors, C1 and C2, should be 15pF \pm 5% and use a COG dielectric. For a detailed specification of the crystal required see Appendix B.1.

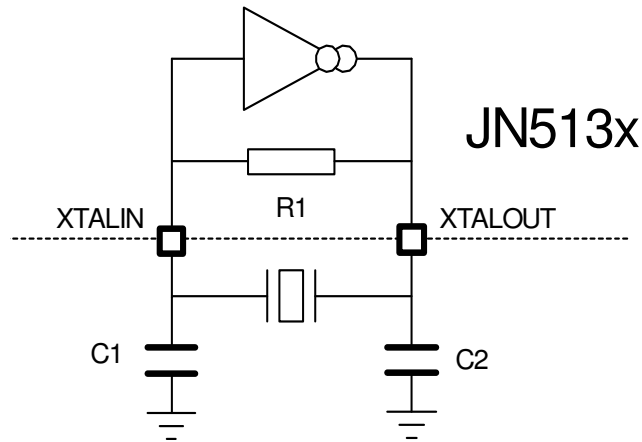


Figure 48: Crystal oscillator connections

The clock generated by this oscillator provides the reference for most of the JN513x subsystems, including the transceiver, processor, memory and digital and analogue peripherals.

B.3 Applications Schematic

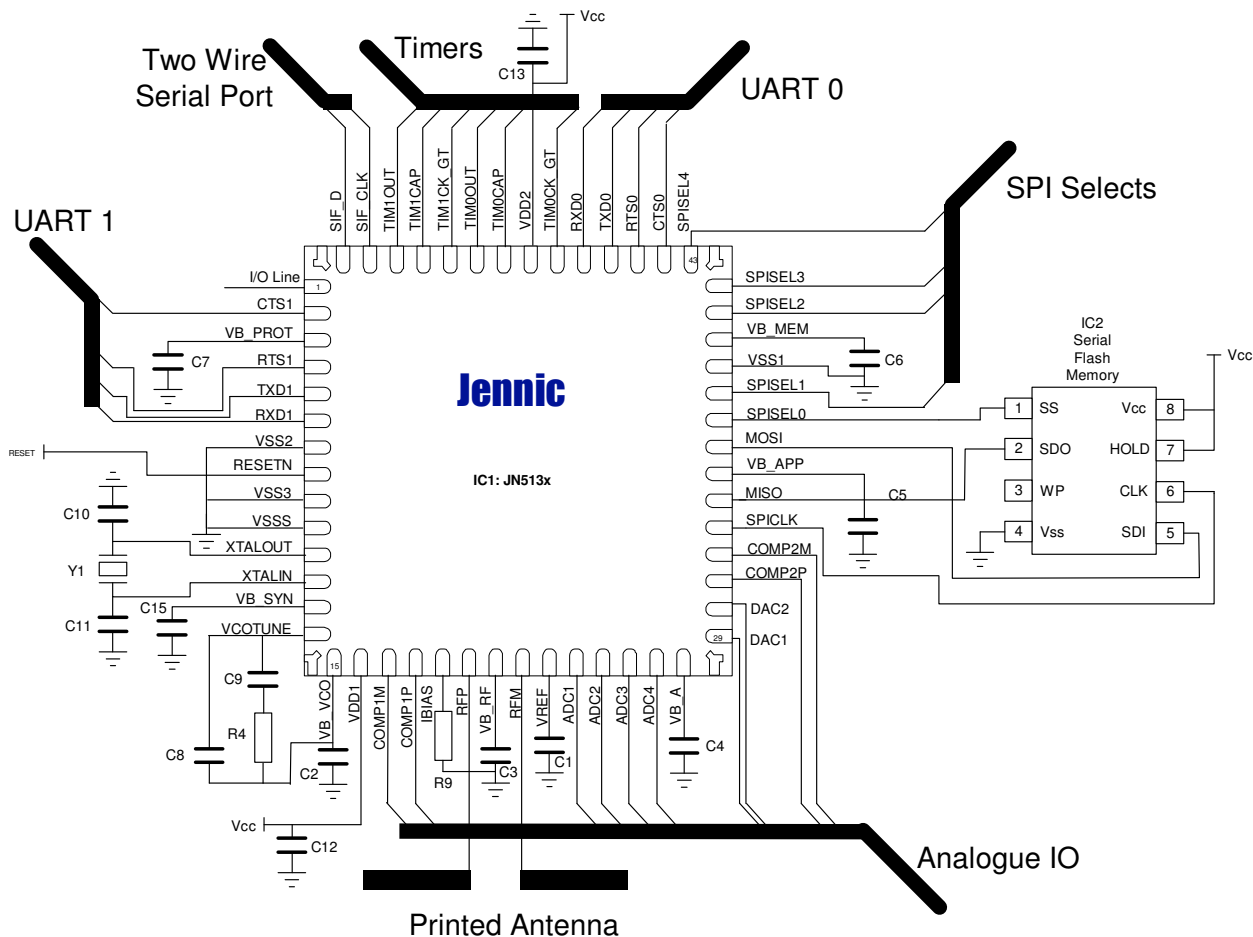


Figure 49: Application Schematic

Components	Values
C1, C2, C3, C4, C5, C6, C7, C12, C13, C15	100nF
C10, C11	15pF (COG)
C9	3n3F
C8	330pF (COG)
R4	4k7Ω
R9	43kΩ
Y1	16MHz Xtal
IC1	JN513x
IC2	128kB Serial Flash

Table 3: Bill of Materials

Appendix C

Related Documents

- [1] IEEE Std 802.15.4-2003 IEEE Standard for Information technology – Part 15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)
- [2] JN-RM-2001 Integrated Peripherals API
- [3] JN-RM-2002 802.15.4 Stack API
- [4] JN-AN-1003 Boot Loader Operation

RoHS Compliance

JN513x devices meet the requirements of Directive 2002/95/EC of the European Parliament and of the Council on the Restriction of Hazardous Substance (RoHS).

Status Information

The status of this Data Sheet is **Advanced**.

Jennic products progress according to the following format:

Advanced

The Data Sheet shows the specification of a product in planning or in development.

The functionality and electrical performance specifications are target values and may be used as a guide to the final specification. Integrated circuits are identified with an R suffix, for example JN5131-Z01R.

Jennic reserves the right to make changes to the product specification at anytime without notice.

Preliminary

The Data Sheet shows the specification of a product that is in production, but is not yet fully qualified.

The functionality of the product is final. The electrical performance specifications are target values and may be used as a guide to the final specification. Integrated circuits are identified with an R suffix, for example JN5131-Z01R.

Jennic reserves the right to make changes to the product specification at anytime without notice.

Production

This is the final Data Sheet for the product.

All functional and electrical performance specifications, including minimum and maximum values are final.

This Data Sheet supersedes all previous document versions.

Jennic reserves the right to make changes to the product specification at anytime to improve its performance.

Disclaimers

The contents of this document are subject to change without notice. Jennic reserves the right to make changes, without notice, in the products, including circuits and/or software, described or contained herein in order to improve design and/or performance. Information contained in this document regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications

Jennic assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work infringement, unless otherwise specified.

Jennic products are not intended for use in life support systems, appliances or systems where malfunction of these products can reasonably be expected to result in personal injury, death or severe property or environmental damage. Jennic customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Jennic for any damages resulting from such use.

All trademarks are the property of their respective owners.

Version Control

Version	Notes
1.0	22 rd December 2006 - First Release
1.1	9th February 2007 – Added solder reflow profile

Contact Details

Corporate Headquarters
Jennic Ltd, Furnival Street
Sheffield S1 4QT, UK
Tel: +44 (0)114 281 2655
Fax: +44 (0) 114 281 2951
info@jennic.com
www.jennic.com

Jennic Ltd Japan
Osakaya building 4F
1-11-8 Higashigotanda Shinagawa-ku
Tokyo 141-0022, Japan
Tel: +81 3 5449 7501
Fax: +81 3 5449 0741
info@jp.jennic.com
www.jennic.com

Jennic Ltd Taiwan
19F-1, 182, Sec.2 Tun Hwa S. Road.
Taipei 106, Taiwan
Tel: +886 2 2735 7357
Fax: +886 2 2739 5687
info@tw.jennic.com
www.jennic.com

Jennic America Inc - West Coast Office
1322 Scott Street, Suite 203
Point Loma, CA 92106, USA
Tel: +619 223 2215
Fax: +619 223 2081
info@us.jennic.com
www.jennic.com

Jennic America Inc - East Coast Office
1060 First Avenue, Suite 400
King of Prussia, PA 19406, USA
Tel: +1 484 868 0222
Fax: +1 484 971 5015
info@us.jennic.com
www.jennic.com

Jennic Ltd Korea
701, 7th Floor, Kunam Bldg.,
831-37, Yeoksam-Dong, Kangnam-ku
Seoul
135-080
Korea
Tel: +82 2 552 5325
Fax: +82 2 3453 8802
info@kr.jennic.com
www.jennic.com