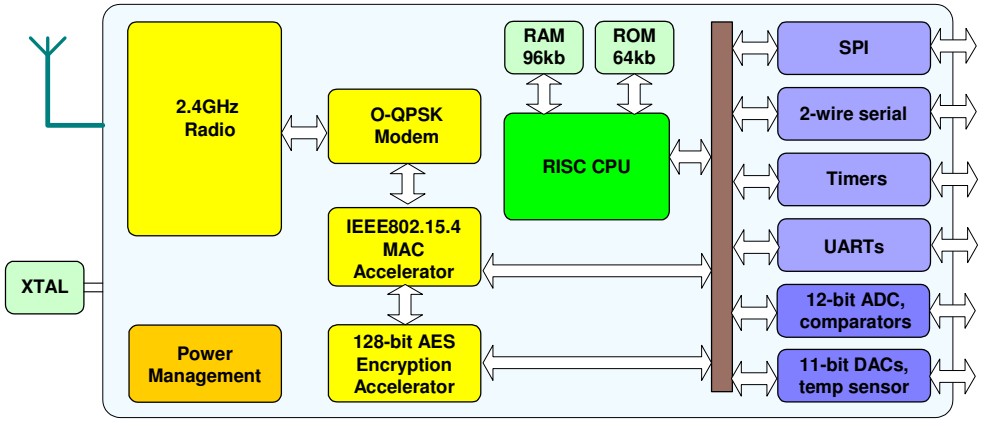
Data Sheet – JN5121

IEEE802.15.4 Wireless Microcontroller

Overview

The JN5121 is the first in a series of low power, low cost IEEE802.15.4 compliant wireless microcontrollers. Combining an on chip 32-bit RISC core, a fully compliant 2.4GHz IEEE802.15.4 transceiver, 64Kb of ROM and 96Kb of RAM, provides a versatile low cost solution for wireless sensor networking applications. The high level of integration helps to reduce the overall system cost. In particular, the ROM enables integration of point-to-point and mesh network stack protocols, and the RAM allows support of router and controller functions without the need for additional external memory. The JN5121 uses hardware MAC and highly secure AES encryption accelerators for low power and minimum processor overhead. Integrated sleep oscillator and power saving facilities are provided, giving low system power consumption. The device also incorporates a wide range of digital and analogue peripherals for the user to connect to their application.

Block Diagram



Benefits

- Single chip solution with integrated transceiver and microcontroller for wireless sensor networks
- Capacity and power efficient microcontroller for both controllers and sensor units
- Low application BOM cost and size
- Hardware MAC ensures low power consumption and low processor overhead
- Extensive user peripherals

Applications

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

Features: Transceiver

- 2.4GHz IEEE802.15.4 compliant
- Security processor (128-bit AES)
- 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
- Sleep current (with active beacon timer) < 5µA
- Minimum of external components at < US$1 cost
- Rx current < 50mA
- Tx current < 40mA
- Receiver sensitivity -93dBm
- Transmit power +1dBm

Features: Microcontroller

- 16MHz 32-bit RISC optimised for low power (3MIPS/mA) and efficient code density
- 96k RAM for shared program, data and routing tables
- 64k ROM for program code
- 4-input 12-bit ADC, 2 11-bit DACs, 2 comparators, temperature sensor
- 2 Application timer/counters, 3 system timers
- 2 UARTs (one for in-system debug)
- SPI port with 5 selects
- 2 wire serial interface
- 21 GPIO

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

8x8mm 56 lead QFN package

Lead-free and RoHS compliant

Contents

1 Introduction

The JN5121 IEEE802.15.4 wireless microcontroller provides a fully integrated solution for applications using the IEEE802.15.4 standard in the 2.4 - 2.5GHz ISM frequency band, including ZigBee™. It includes all 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 JN5121. 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 JN5121 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 JN5121 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 Hardware Peripheral Library Reference Manual [2].

An IEEE802.15.4 compliant wireless network can be developed using the IEEE802.15.4 MAC library described in JN-RM-2002 Stack Software Reference Manual [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 which 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 which require increased transmit power to be realised very easily.

The Security coprocessor provides hardware-based 128-bit AES-CCM, CBC, CTR and CCM* processing as specified by the 802.15.4 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].

1.3 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 memory space of the JN5121 is configured as 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 64K bytes of ROM, 96K bytes of RAM and 128 bytes of Low Power RAM (LPRAM).

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 I^2C)
- Slave SPI port
- Twenty-one digital IO 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 Hardware Peripheral Library Reference Manual [2] describes this interface in more detail.

1.5 Block Diagram

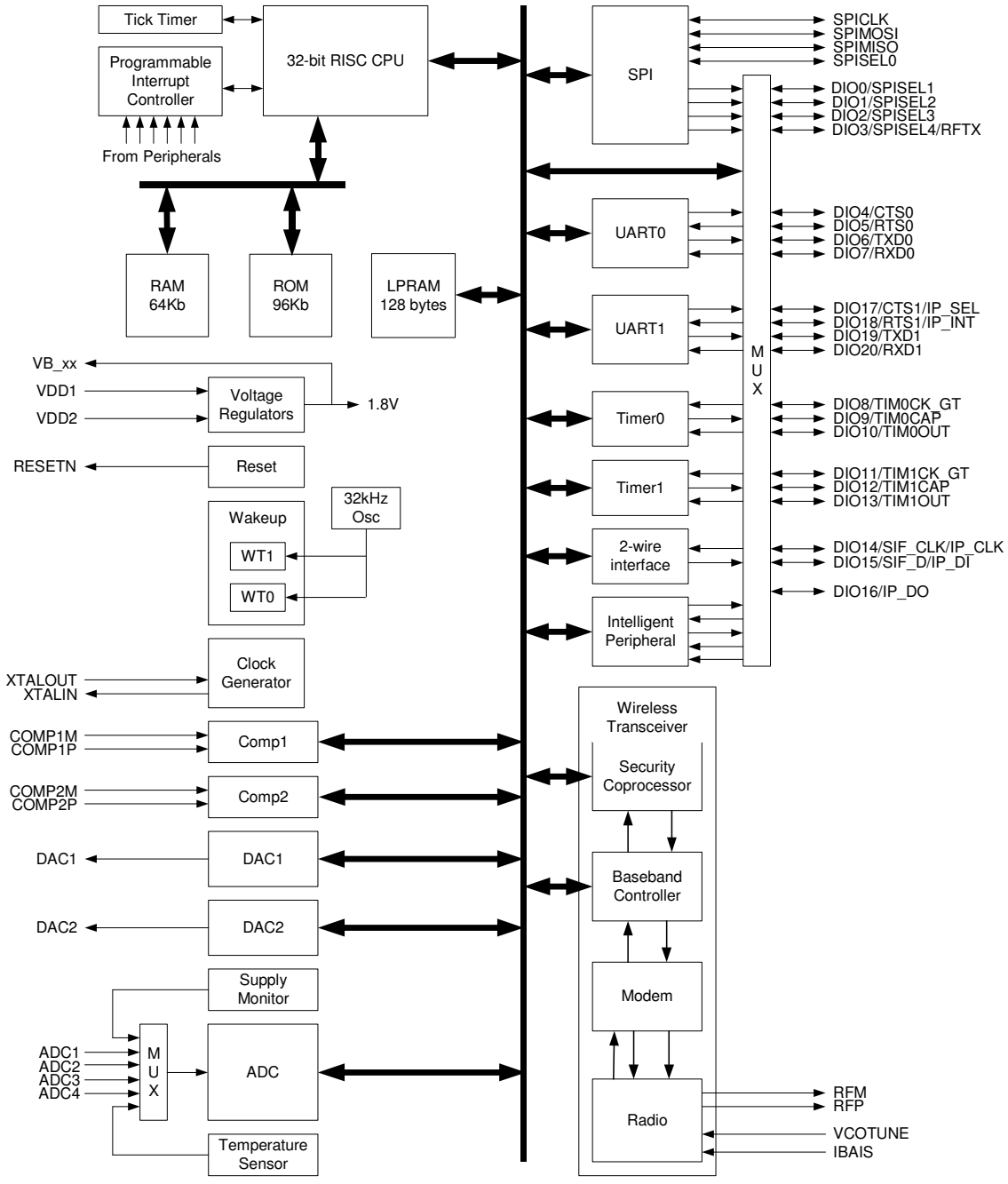


Figure 1: JN5121 Block Diagram

2 Pin Configurations

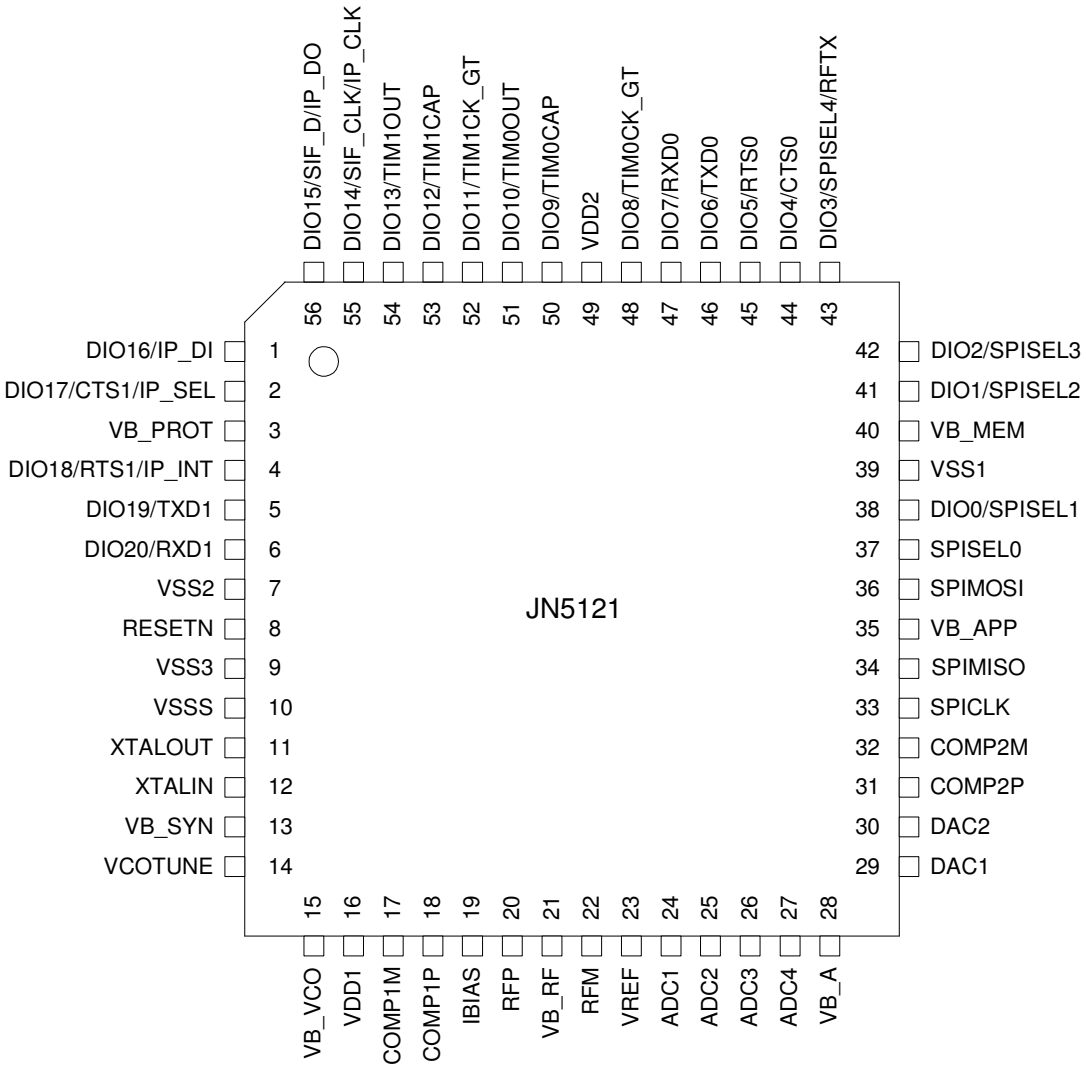


Figure 2: Pin Configuration (top view)

2.1 Pin Assignment

Pin No	Power supplies		Description
3, 13, 15, 21 28, 35, 40	VB_PROT, VB_SYN, VB_VCO, VB_RF, VB_A, VB_APP, 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	DIO2 or SPI Slave Select Output 3
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	DIO9 or Timer0 Capture Input
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
54	DIO13	TIM1OUT	DIO13 or Timer1 PWM Output
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 and should be decoupled to analogue ground. VDD2 is the power supply for the digital circuitry and 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_APP and VB_PROT 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 bidirectional 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 JN5121 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

The system clock is driven by a crystal connected between XTALIN and XTALOUT. 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, four comparator inputs and two DAC outputs. The analogue IO pins on the JN5121 can have signals applied up to 0.3v higher than VDD1. A schematic view of the analogue IO cell is shown in Figure 3 Analogue IO 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 comparators may optionally be used as a wakeup.

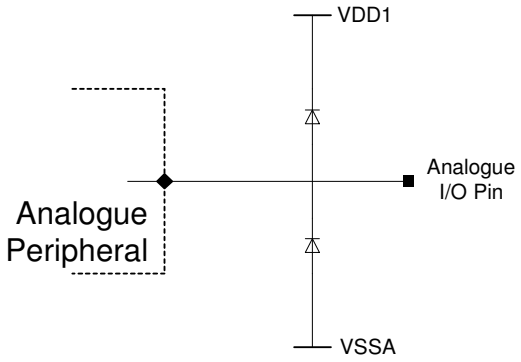


Figure 3 Analogue IO Cell

2.2.6 Digital Input/Output

Digital IO pins on the JN5121 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 IO cell is in Figure 4: DIO Pin Equivalent Schematic.

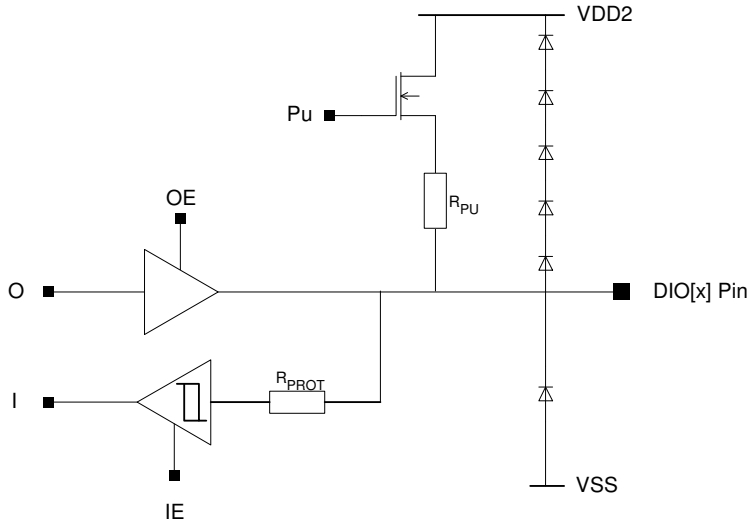


Figure 4: DIO Pin Equivalent Schematic

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 IO. The use of the pull-up resistor Rpu for each pin is controlled through the `vAHI_DioPullupControl()` 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 JN5121 from sleep.

3 CPU

The CPU of the JN5121 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 (16MHz) 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 a number of hardware Multiply/Accumulate instructions which 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 JN5121 is by using C, which is supported by a software developer kit comprising a C compiler, linker and debugger. For more detail on the CPU instruction set, refer to the JN-RM-2010 CPU Reference Manual [5].

The CPU architecture also contains features which make the processor suitable for embedded, real-time applications. In more complex 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 which has 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 JN5121, and are described more fully in section 16, Power Management and Sleep Modes. One of these modes is the doze mode which affects only the CPU, which shuts down the processor under software control when it is known to be idle and wakes up on interrupt.

4 Memory Organisation

This section describes the different memories found within the JN5121. The device contains ROM, RAM, the wireless transceiver and peripherals all within the same linear address space.

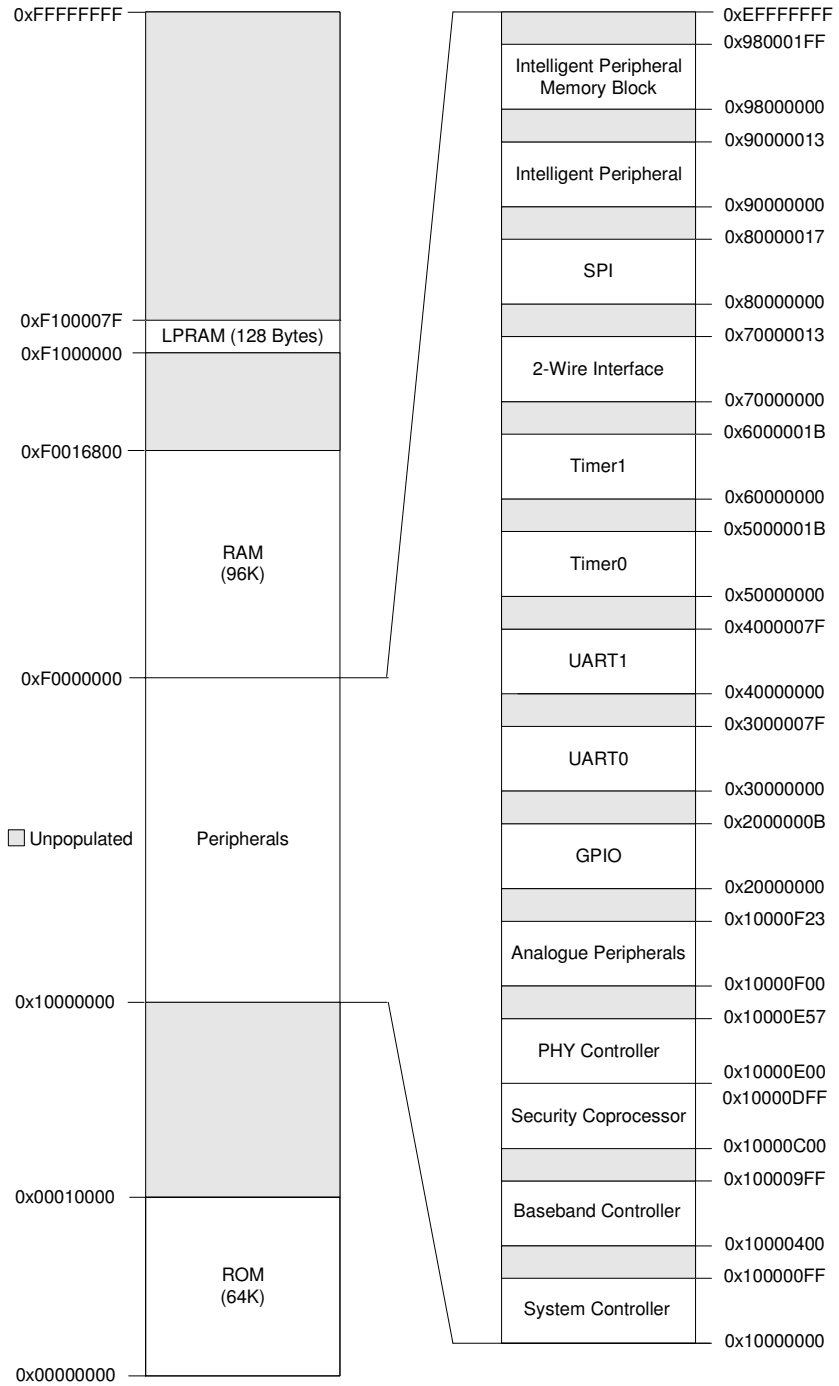


Figure 5: JN5121 Memory Map

4.1 ROM

The ROM is 64K bytes in size, organized as 16k 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 which may contain different protocol stacks or applications, although all versions will carry a default interrupt vector table and interrupt manager. Variants which can be used for application or protocol development will carry support for debugging and a boot loader, to allow the non-ROMed code to be loaded from external Flash memory. The operation of the boot loader is described in detail in Application Note JN-AN-1003 Boot Loader Operation [4]. Further information regarding the debug support can be found in the JN-UG-3001 SDK Installation User Guide [6]. 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 protocol stack is shown in Figure 6.

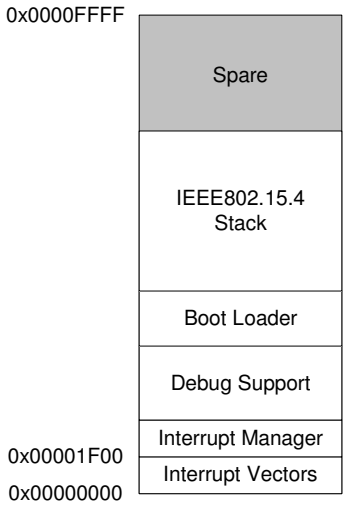


Figure 6: Typical ROM contents

4.2 RAM

The JN5121 contains 96K bytes of high speed RAM organized as 24k x 32-bit words. It can be used for both code and data storage and is accessed by the CPU in a single clock cycle. At reset, segments of code and data of the software not in ROM may be loaded from an external memory connected to the SPI port with a dedicated select line, under control of a boot loader utility. 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 unpowered.

4.3 Low-Power RAM

The 128 bytes of low power RAM are organized as 32 x 32-bit words and are available for the storage of application data. The power to this memory is maintained as long as an external supply is present and therefore the contents are held during all sleep modes. This allows key parameters, for example a MAC address to be stored at lowest power in an application. Unlike the other areas of memory the CPU requires 3 clock cycles to access the LPRAM.

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.

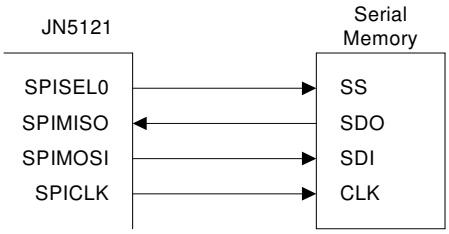


Figure 7: Connecting External Serial Memory

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 JN5121 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 JN-AN-1003 Boot Loader Operation for details on the format of the read command and other details of the boot loader.

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.

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 JN5121 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 12pF ±5% and use a COG dielectric. For a detailed specification of the crystal required see Appendix B.1.

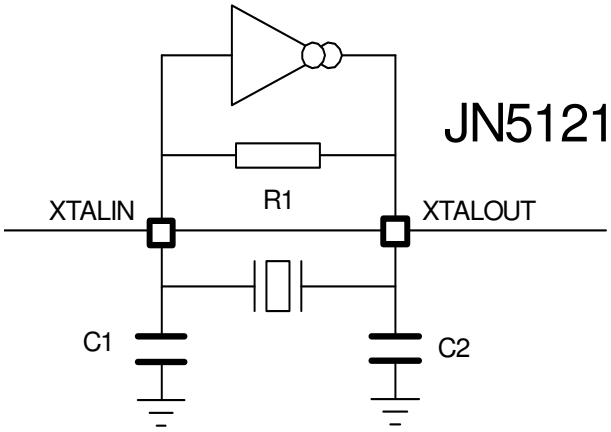


Figure 8: Crystal oscillator connections

The clock generated by this oscillator provides the reference for most of the JN5121 subsystems, including the transceiver, processor, memory and digital and analogue peripherals.

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 timing components within the device have a wide tolerance due to process variations, meaning that the oscillator is nominally 32kHz ±30%. In order to calibrate the actual frequency of this oscillator, a reference timer is provided which is clocked from the more accurate crystal oscillator. The RC oscillator can be set in a mode where the number of 16MHz clocks per 32kHz period is measured, and then used by software as a correction factor when programming the number of (nominally) 32kHz clock periods to be used in a sleep period. See section 12.3.1 for more details.

6 Reset

A system reset initializes the device to a predefined state and forces the CPU to start program execution from the reset vector. The reset process that the JN5121 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 crystal oscillator has begun oscillating at 16MHz. The crystal oscillator is running at full speed when at least 300 16MHz positive clock edges are seen in one 32kHz clock period. This figure takes into account the situation where the 32kHz oscillator runs at its fastest allowed tolerance.

The typical start-up time for the 16MHz oscillator is 2.5msec.

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 provided detailed electrical data and timing. Appendix B describes the JN5121 pin states during and after reset.

The JN5121 has three sources of reset:

- Power-on Reset
- External Reset
- Software Reset

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 JN5121 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.

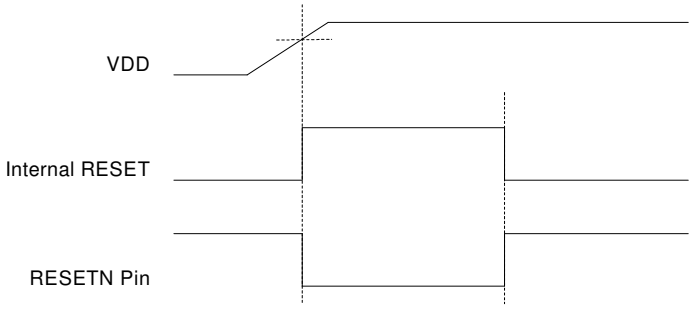


Figure 9: Power-on Reset

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 JN5121 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.

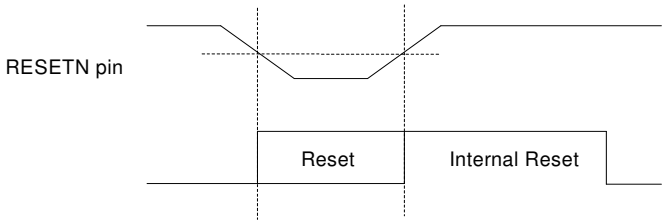


Figure 10: External Reset

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 JN5121 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 RESETN Pin

Multiple devices may connect to the RESETN pin in an open-collector mode. The JN5121 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.

7 Interrupt System

The interrupt system on JN5121 is a hardware-vectored interrupt system. The JN5121 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 are always performed in supervisor mode. For more detail, refer to the JN-RM-2010 CPU Reference Manual [5]. 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 which 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 which 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 JN5121 from sleep. The peripherals, baseband controller, security coprocessor and PIC are powered down during sleep but the wake-up timers, DIO interrupts and analogue comparator interrupts remain powered to bring the JN5121 out of sleep.

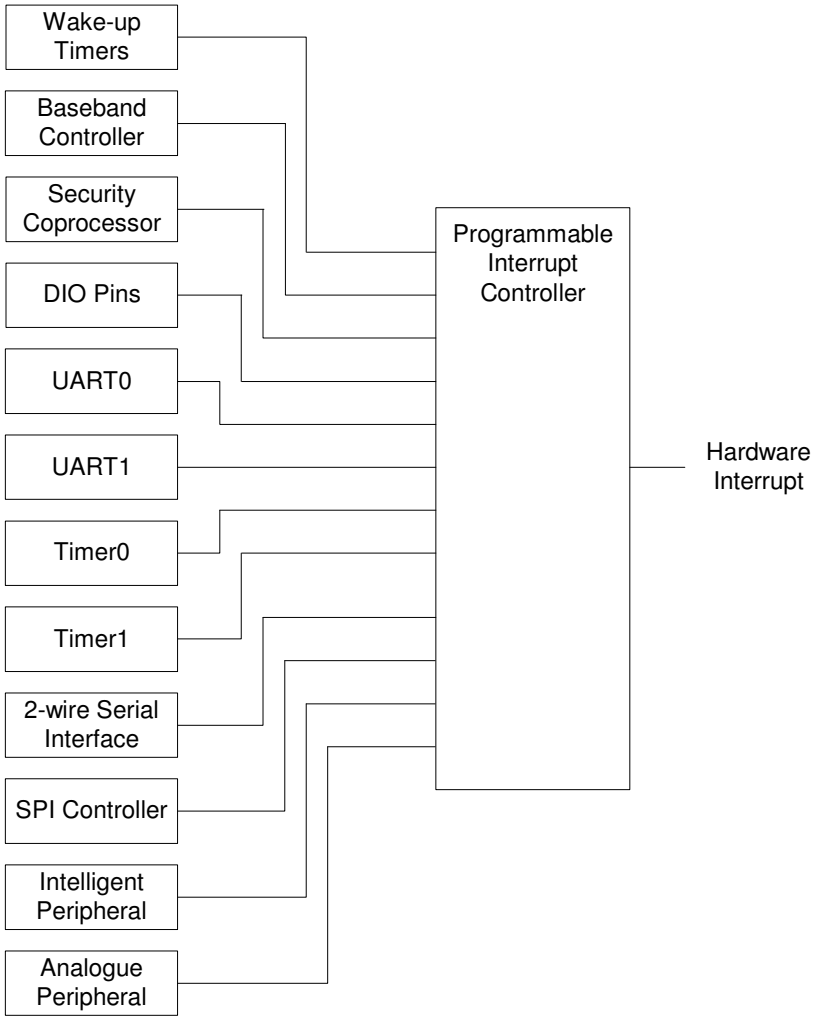


Figure 11: Programmable Interrupt Controller

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 Stack Software Reference Manual [3]. Applications interface to the protocol software via an API interface which corresponds to the SAP interfaces defined in IEEE Std 802.15.4-2003 [1]

8.1 Radio

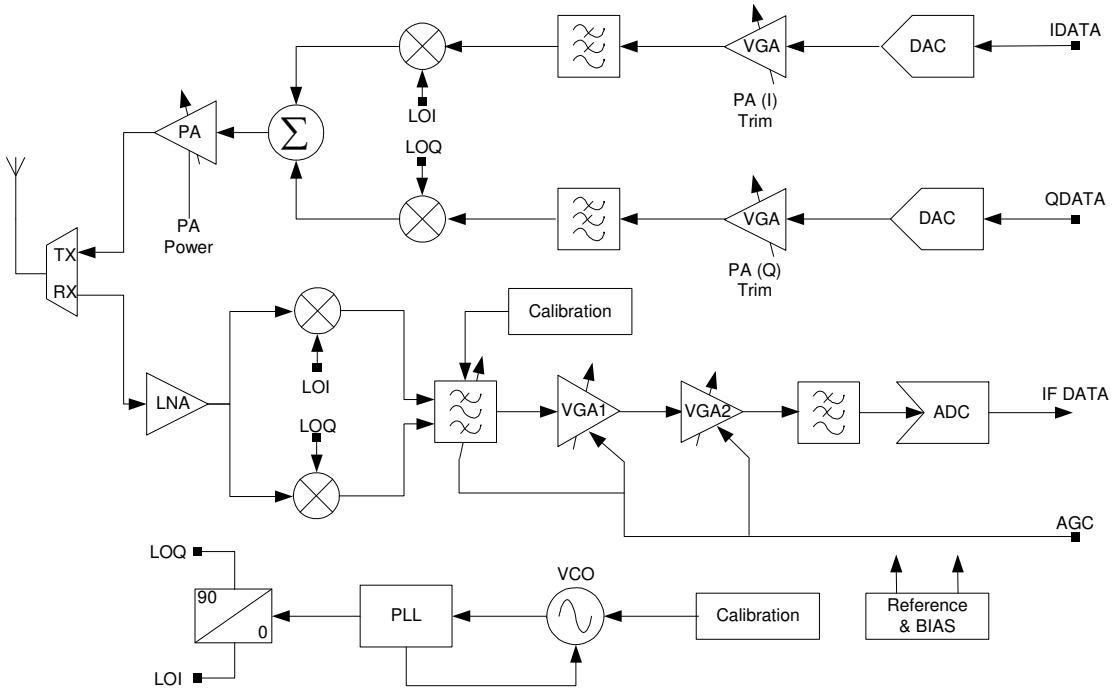


Figure 12: Radio Architecture

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 frequency synthesiser to provide a reference frequency to the 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) which 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 to be placed close to the JN5121 pins and analogue ground.

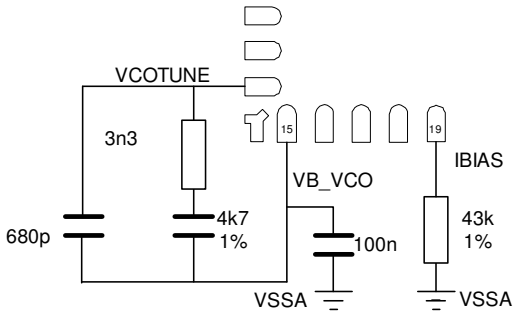


Figure 13: VCO Loop Filter and IBIAS

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 JN5121 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.

8.2 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.

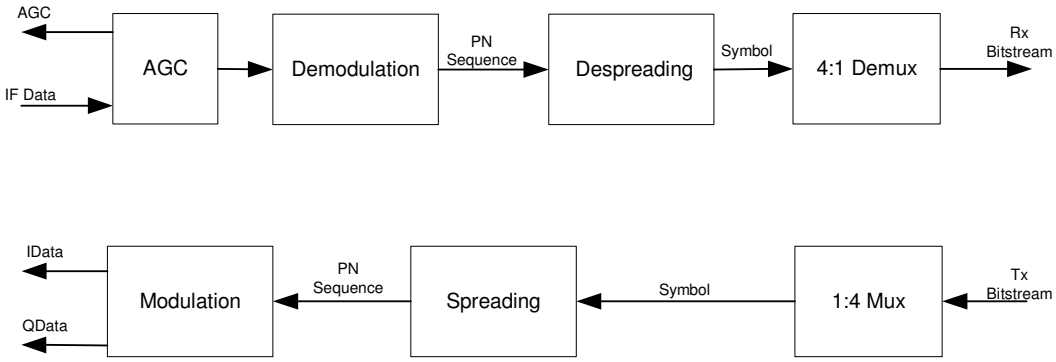


Figure 14: Modem Architecture

The transmitter assembles serial bitstream data from the baseband processor into 4-bit symbols. These are passed to the spreading function which maps 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. These baseband signals are subsequently converted to an analogue voltage 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, matched filtering, synchronization and data slicing. The synchronization scheme is tolerant to both carrier and symbol frequency offsets in excess of ±80ppm, thereby satisfying the maximum centre frequency tolerance specified in the IEEE802.15.4 standard.

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 has an integrated 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 minimum and maximum LQI values are associated with the lowest and highest quality IEEE 802.15.4 signals detectable by the receiver and the modem produces a LQI based upon the ED measurement.

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 partition provides for software to implement the sequencing of events required by the protocol and to schedule events at times with millisecond resolution, whereas the hardware implements 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.

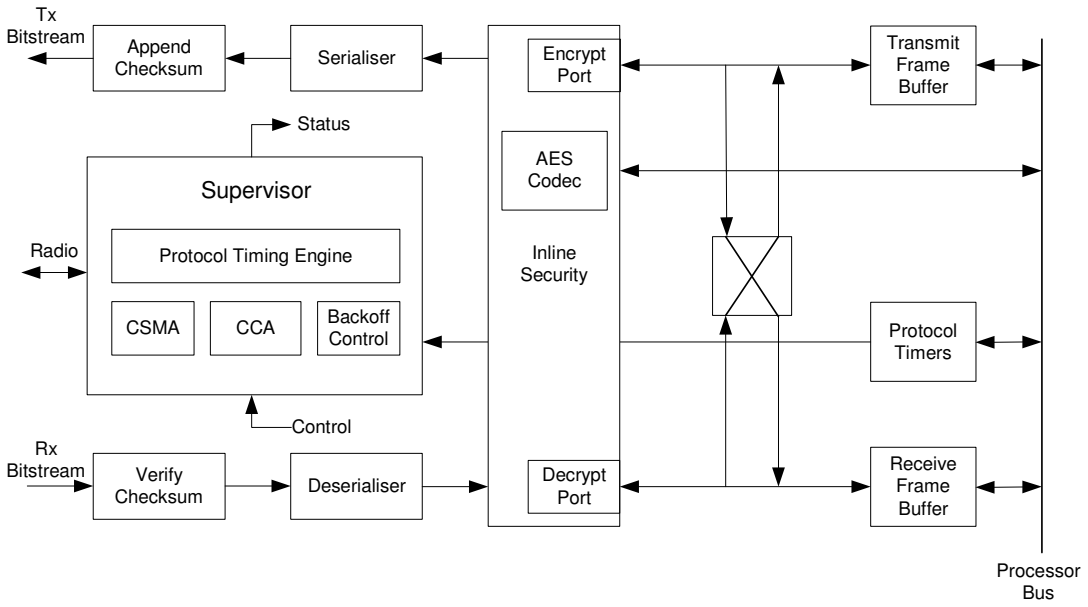


Figure 15: Baseband Processor

8.3.1 Transmit

A transmission is performed by software writing the data to be transferred into the transmit 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 which calculates the checksum on-the-fly, and appends it to the end of the frame.

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 Receive 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 is monitoring the Link Quality determined by the number of bit errors in the packet. The Link Quality Indication (LQI) value 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 acknowlege packet within a very short window after the transmitted frame has been received. The packet contains details of whether the transmitted packet was received error free. The JN5121 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.

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 all incoming and outgoing data through an in-line security engine which is able to perform encryption and decryption operations on-the-fly, with the result that minimal processor intervention is required. 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 and decryption operations over a streaming interface, it is also possible to use the AES core as a coprocessor to be used by the CPU of the JN5121. 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 the Zigbee™ standard) 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 peripheral library for the JN5121 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.

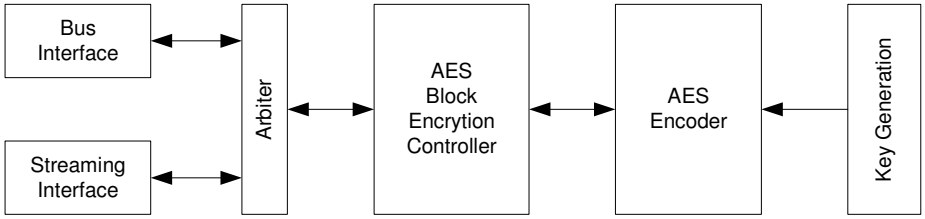


Figure 16: Security Coprocessor Architecture

9 Digital Input/Output

There are 21 Digital IO (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 functions for the peripheral features on the device. The alternate function is selected by enabling the relevant peripheral. Refer to the individual module sections for a full description of the alternate functions. From reset all peripherals are off and the DIO pins are configured as inputs with the internals pull-ups turned on.

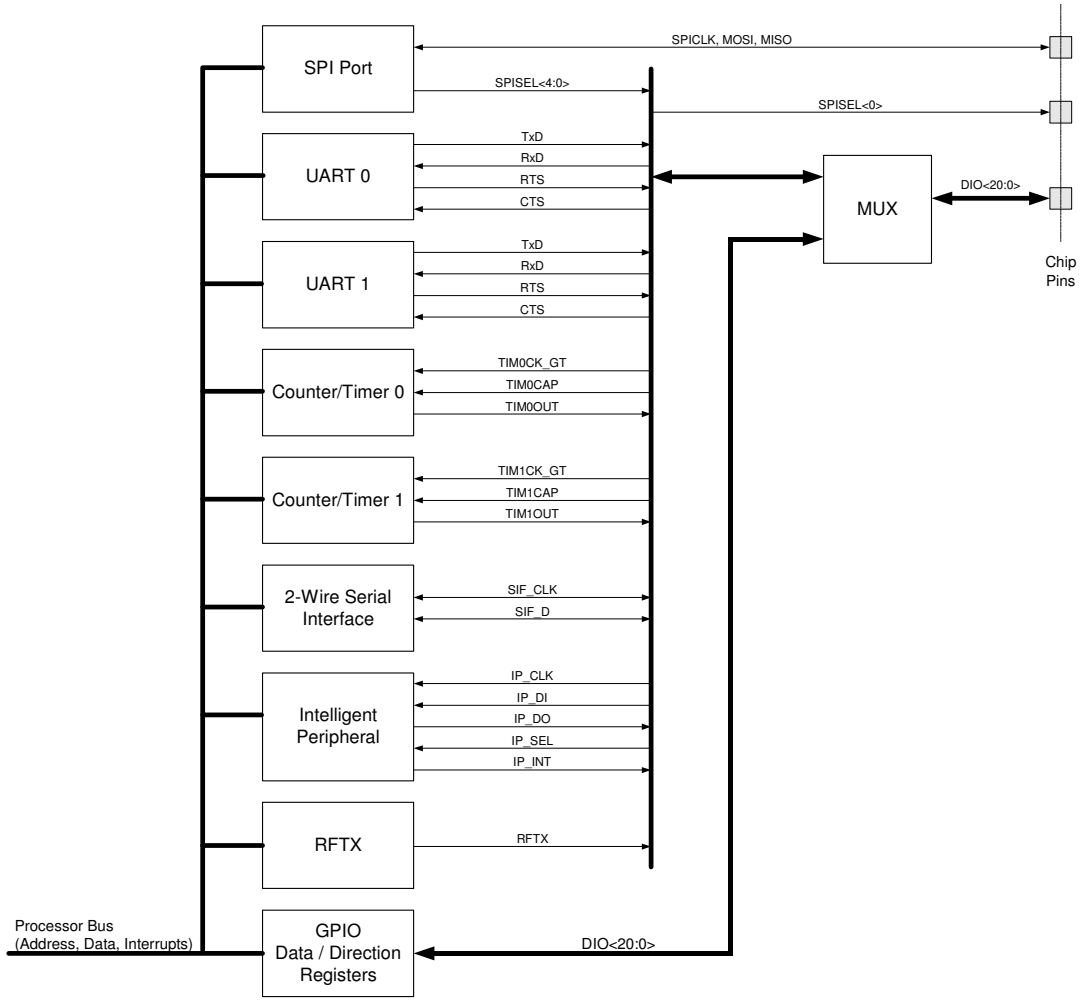


Figure 17: DIO Block Diagram

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 which can be used to wake the device up. Selection of the interrupt transition is done using `vAHI_DioIntEdge()`. Enabling and masking of DIO interrupts is done using `vAHI_DioIntEnable()` while the status of a DIO interrupt is given by `u32AHI_DioIntStatus()`. See section 16.3 for further details on sleep and wakeup.

10 Serial Peripheral Interface

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the JN5121 and peripheral devices. The JN5121 operates as a master on the SPI bus and all other devices connected to the SPI are expected to be slaves devices under the control of the JN5121 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

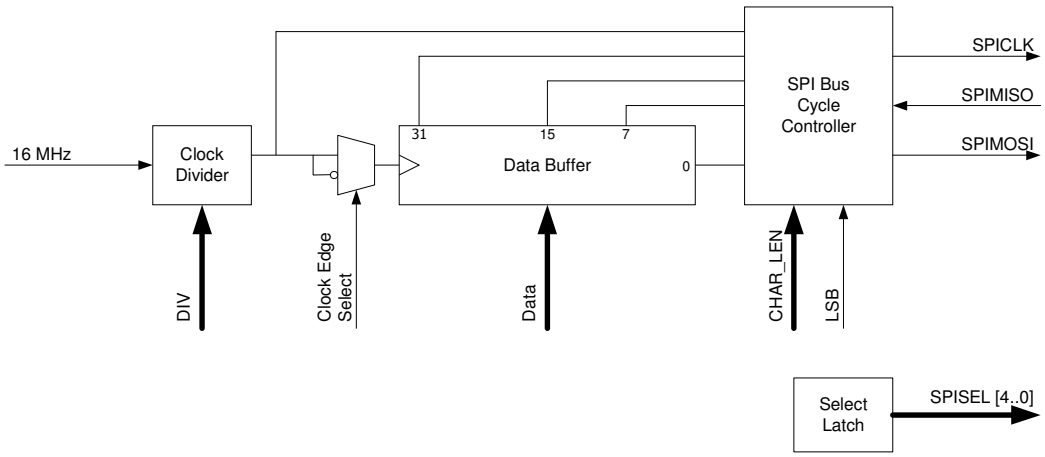


Figure 18: SPI Block Diagram

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 JN5121.

The JN5121 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.

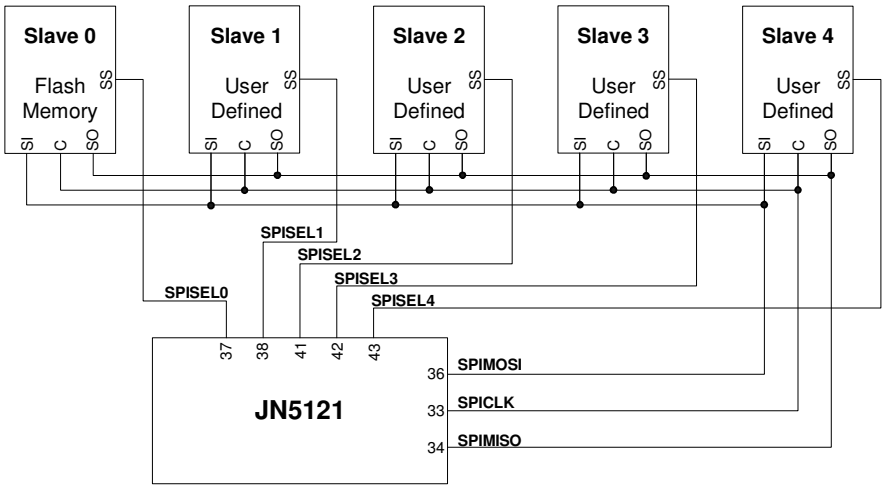


Figure 19: Typical JN5121 SPI Peripheral Connection

The data transfer rate on the SPI bus is determined by the SPICLK signal. The JN5121 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 high when the interface is not being used. The clock phase determines which edge of SPICLK is used by the JN5121 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, SPISELn, 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 JN5121 indicating 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 initialize 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

```
PRIVATE void vReadFromFlash(uint32 u32Addr, uint32 u32NumWords, uint8 *pau8Buffer )
{
    #define FLASHREADCMD 0x03

    uint32 u32Temp;
    uint32 i;

    vAHI_SpiConfigure(  1,          /* number of slave select lines in use*/
                        MSBFIRST,   /* data to be sent MSB first */
                        TXNEG,      /* TX data to change on negative edge */
                        RXNEG,      /* RX data to change on negative edge */
                        SPICLK16M,  /* Generate 16MHz SPI clock */
                        SPIINTOFF,  /* Disable SPI interrupt */
                        SPIASSOFF);   /* Disable auto slave select */

    /* combine read cmd & addr into single value to be sent over SPI */
    u32Temp = (u32Addr & x00FFFFFF) | (FLASHREADCMD << 24);

    vAHI_SpiSelect(SPISEL0);              /* select spi device */
    vAHI_SpiStartTransfer32(u32Temp);     /* send read cmd and address location */
    vAHI_SpiWaitBusy();

    for (i=0; i<=u32NumWords; i++)
    {
        vAHI_SpiStartTransfer32(u32Addr);   /* read data 4 bytes at a time */
        vAHI_SpiWaitBusy();
        u32Temp = u32AHI_SpiReadTransfer32(); /* copy into temp variable */
        memcpy( (pau8Buffer+i), &u32Temp, sizeof(u32Temp) ); /* copy to buffer */
    }
    vAHI_SpiSelect(SPISELNONE);              /* deselect select spi device */
}
```

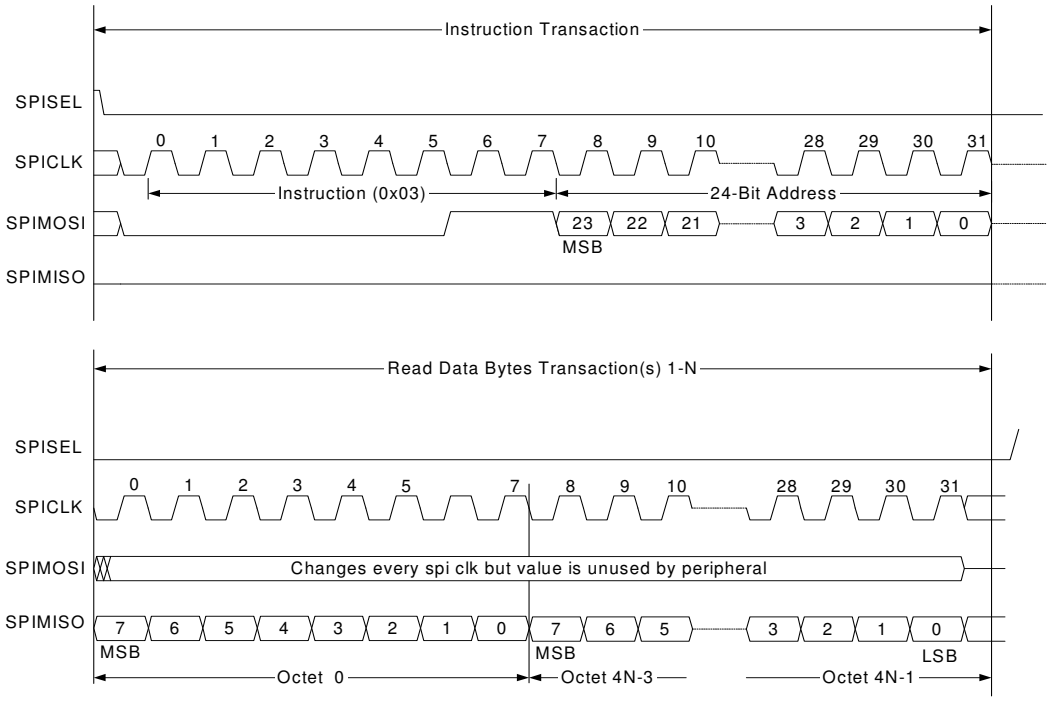


Figure 20: SPI Transaction Waveforms

11 Intelligent Peripheral Interface

The Intelligent Peripheral (IP) Interface is provided for use in more complex systems where there is a processor that requires a wireless peripheral. As an example, the JN5121 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 JN5121 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.

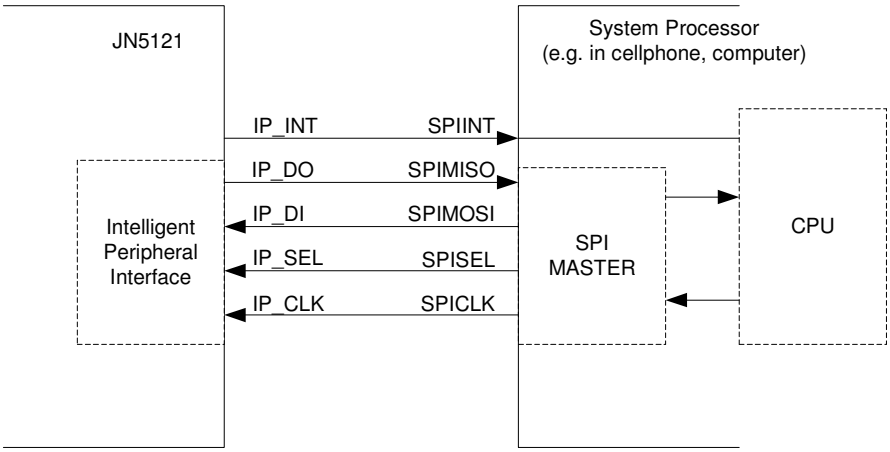


Figure 21: Intelligent Peripheral Connection

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_DOUT 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 JN5121 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 JN5121 or the remote processor

have no data to transfer 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 JN5121 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 JN5121 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 the transmit and receive data have 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.

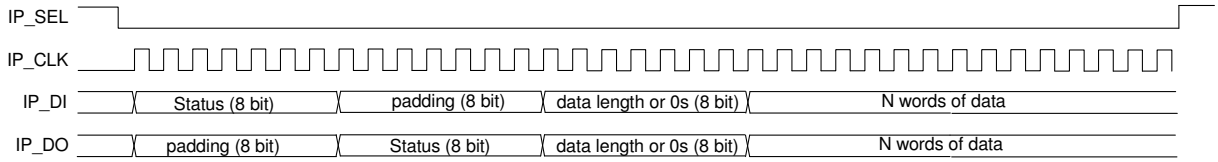


Figure 22: Intelligent Peripheral Data Transfer Waveforms

11.2 JN5121 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 JN5121. When a remote processor starts a transfer to the JN5121, 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 (ie 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 receive buffer needs to be configured in memory using `bAHI_IpSetRxBuffer()`. 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 JN5121. Successful data arrival can be indicated by an interrupt, or the interface can be polled using `bAHI_IpRxDataAvailable()`

To send and receive at the same time the transmit and 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 JN5121 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 JN5121, it should check that the JN5121 has a buffer ready by reading the RXRDY bit. If the RXRDY bit is 0 indicating that the JN5121 cannot accept data, it should terminate the transfer by deasserting IP_SEL unless it is receiving data from the JN5121. If the RXRDY bit is 1, indicating that the JN5121 can accept data, then the master should generate a further 6 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 JN5121. The master should then deassert IP_SEL to show the transfer is complete.

The master may initiate a transfer to read data from the JN5121 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 JN5121, it should check that the JN5121 has a buffer ready by reading the TXRDY bit. If the TXRDY bit is 0, indicating that the JN5121 does not have data to send, it should terminate the transfer by deasserting IP_SEL unless it is transmitting data to the JN5121. If the TXRDY bit is 1, indicating that the JN5121 can send data, then the master should generate a further 6 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 JN5121. 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 x Prescale value as the clock to the timer
- Clock source selectable between internal system clock and external input
- 16-bit counter
- 16-bit High and Low (period) registers
- Timer: can generate interrupts off High and Low count. 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
- Sigma-Delta: NRZ and RZ modes

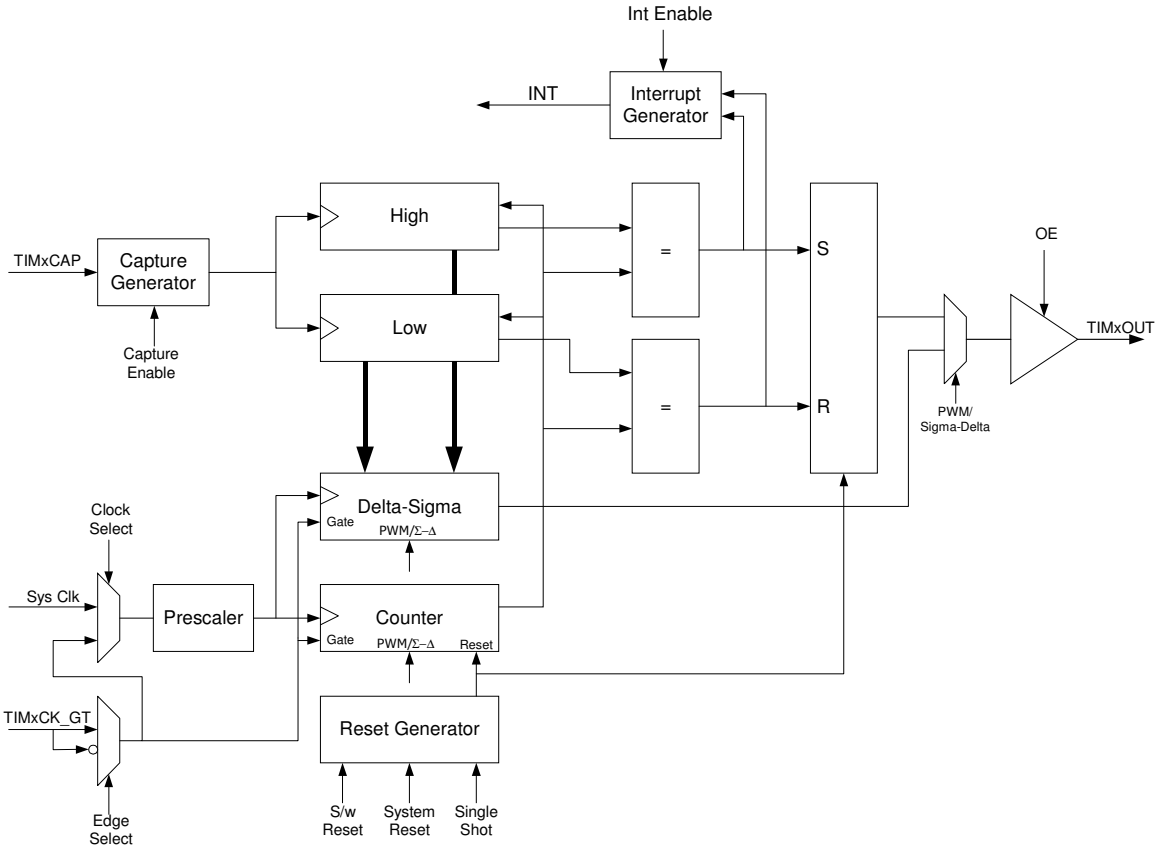


Figure 23: Timer Unit Block Diagram

The clock source is selectable using v`AHI_TimerClockSelect()` between the system clock or an external pin (TIMxCK_GT); the selected clock is fed to a divider or prescaler. The prescaler has a 16-bit divider value, a value of 0 leaving the clock unmodified, with other values dividing the clock by 2 x prescale value. Hence a prescale value of

2 applied to the 16MHz system clock results in a frequency 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 v`AHI_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.

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 (Low and High). The counter increments and its output is compared to the 16-bit High and Low registers. When the counter is equal to the High register, the PWM output is set to high; when the counter reaches the Low value, the output returns to low. In continuous mode, when the counter reaches the Low 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()`.

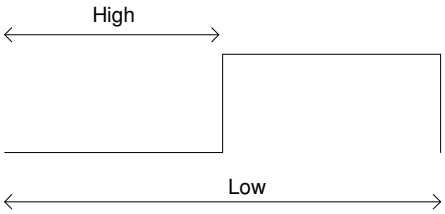


Figure 24: PWM Output Timings

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) with a resolution set by the selected clock and prescaler value. The mode is selected and the counter started by `vAHI_TimerStartCapture()`. On the next low-to-high transition of the signal on the capture input the count value is stored in the High register, and on the following high-to-low transition the counter value is stored in the Low register. The pulse width is the difference in counts in the two registers multiplied by the clock period driving the counter. 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.

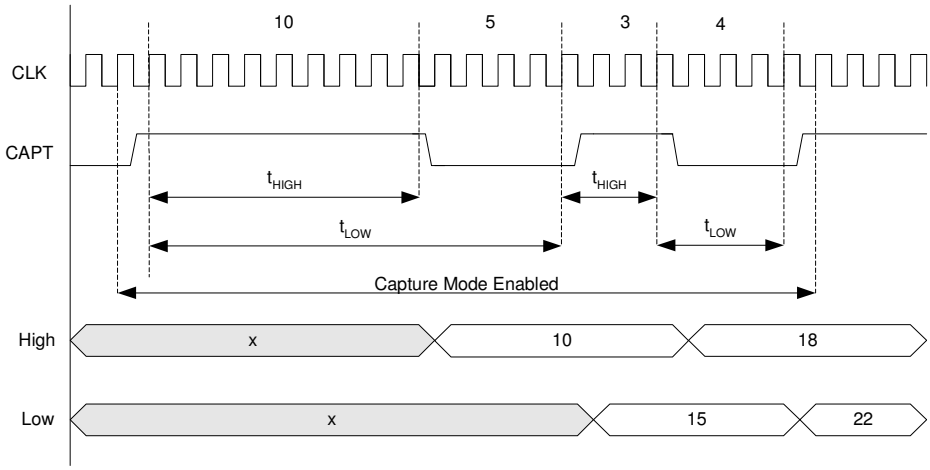


Figure 25: Capture Mode

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 Low register and the Low 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 Low 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 Low register. The low 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 which 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, which may cause the response of the RC network to be uneven, especially if the time constant chosen is comparable to the individual pulse width. 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 NRZ and RTZ for the same programmed cycle time value and number of pulses.

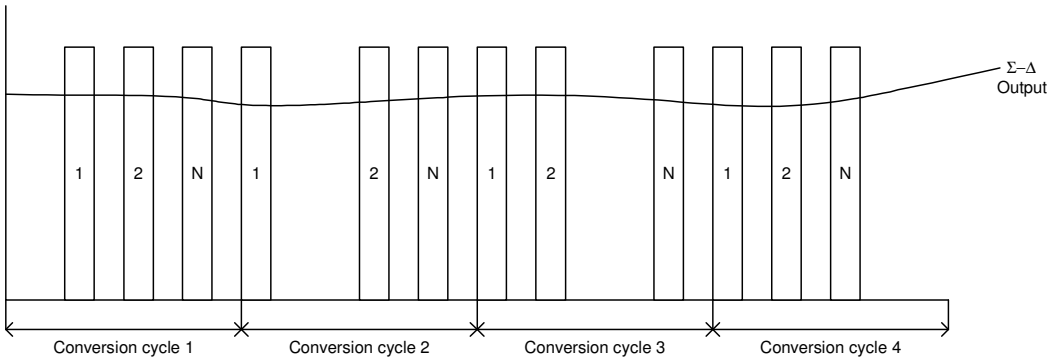


Figure 26: Return To Zero Mode in Operation

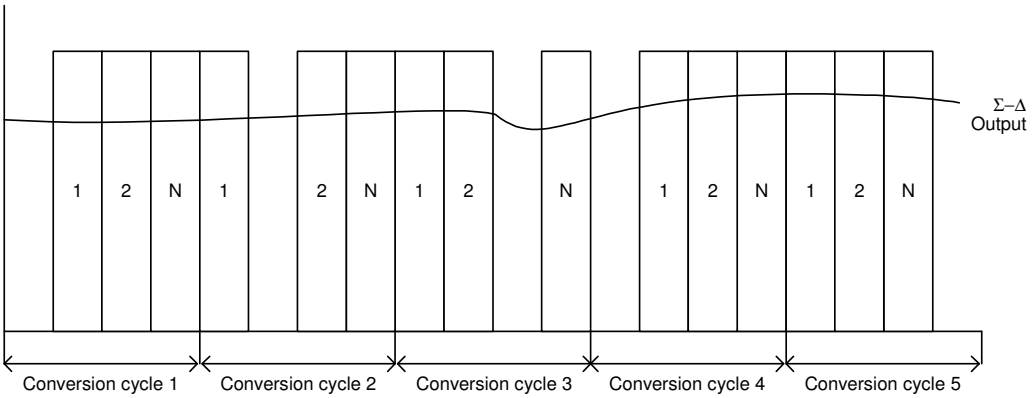


Figure 27: Non-Return to Zero Mode

12.1.5 Timer / Counter Application

Figure 28 shows an application of the JN5121 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.

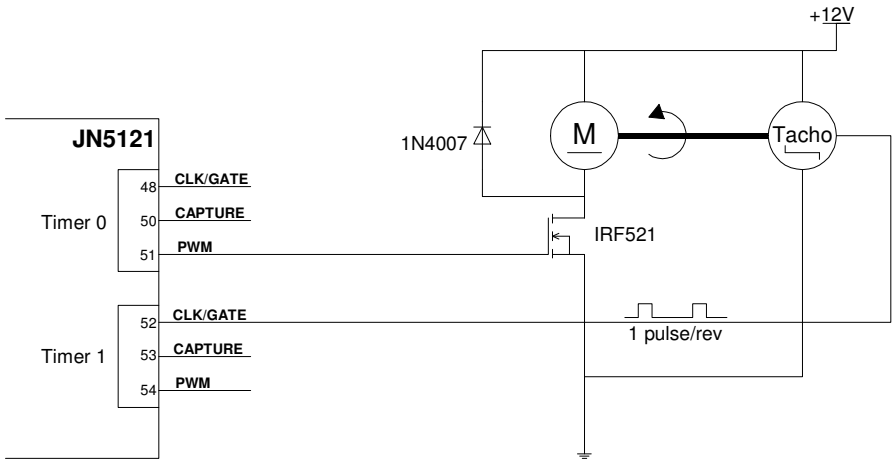


Figure 28: Closed Loop PWM Speed Control Using JN5121 Timers

12.2 Tick Timer

The JN5121 contains a hardware timer which 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

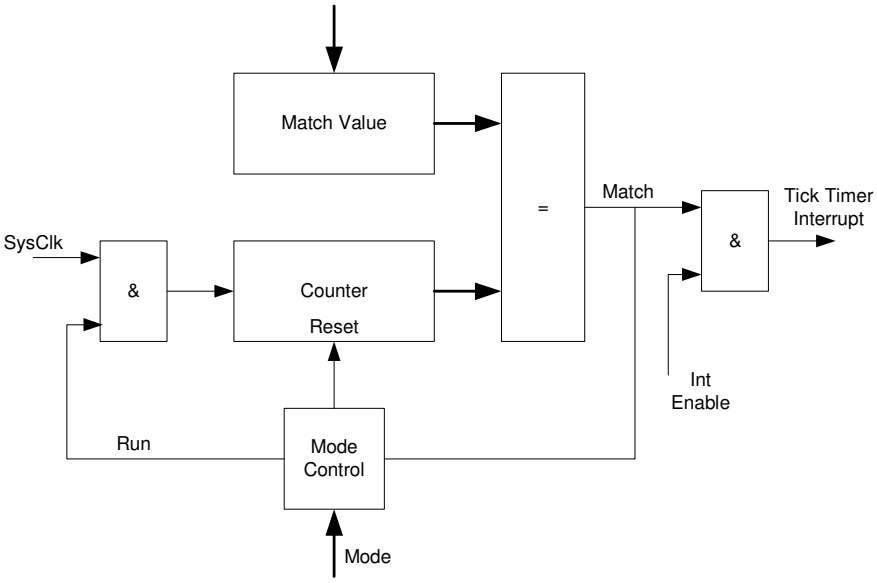


Figure 29: Tick Timer

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 is also 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_TickTimerIntClr()`.

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 JN5121 driven from the 32kHz internal clock. They continue to run during sleep periods when the majority of the rest of the device is powered down. The wakeup timers do not run during deep sleep. They are intended to be used in timing sleep periods or other long period timings that may be required by the application. When a wakeup timer expires it typically generates an interrupt which, if the device is asleep 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. Feature include:

- 32-bit down-counter
- 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 the fired status will be reset by the function. The value of the counter can be read using `u32AHI_WakeTimerRead()`.

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. As a consequence it has low accuracy and temperature stability. 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 still expect to see it. For less accurate sleep times 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 will consume more power.

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 specified 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 JN5121 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

Due to the temperature dependent behaviour of the RC Oscillator, the calibration process should be done performed regularly (for example once every wakeup period) to account for changes in ambient temperature.

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 10 seconds the timer should be loaded with 35,556 ((10000/9000) * (32000*10)) rather than 32000.

13 Serial Communications

The JN5121 has two independent Universal Asynchronous Transmit / Receive (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
- 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

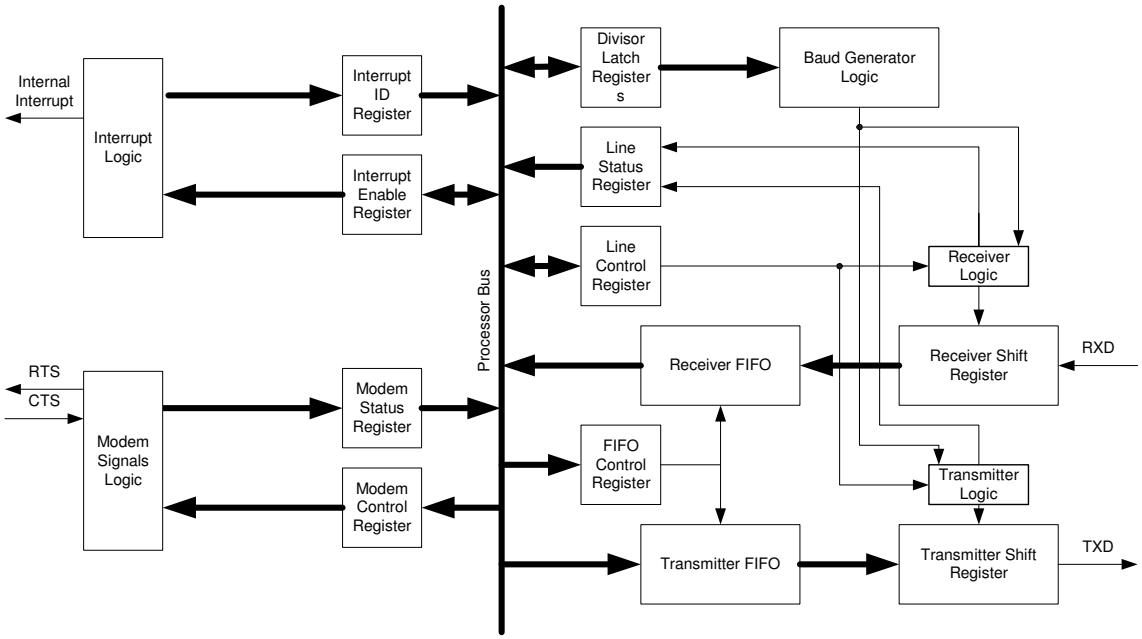


Figure 30 UART Block Diagram

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.

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()`. This 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 `u8AHI_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.

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 connect to a 9-pin connector compatible with a PC. The software developer kit uses such an interface as the debugger interface between the JN5121 and a PC. As the JN5121 device pins do not provide the RS232 line voltage a level shifter is used.

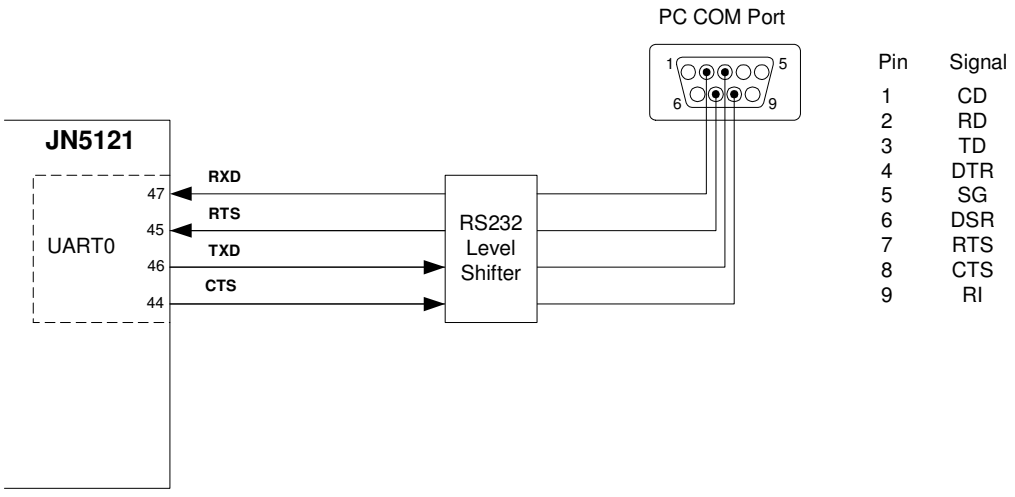


Figure 31 JN5121 Serial Communication Link

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, UART_TX_RESET, 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,
                    UART_EVEN_PARITY,
                    UART_NO_PARITY,
                    E_AHI_UART_WORD_LEN_8,
                    UART_1_STOP_BIT,
                    UART_RTS_INACTIVE);
/* clear reset */
vAHI_UartReset(E_AHI_UART_0, UART_TX_ENABLE, 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 JN5121 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 bidirectional data transfers and includes the following features:

- Compatible with both I^2C 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

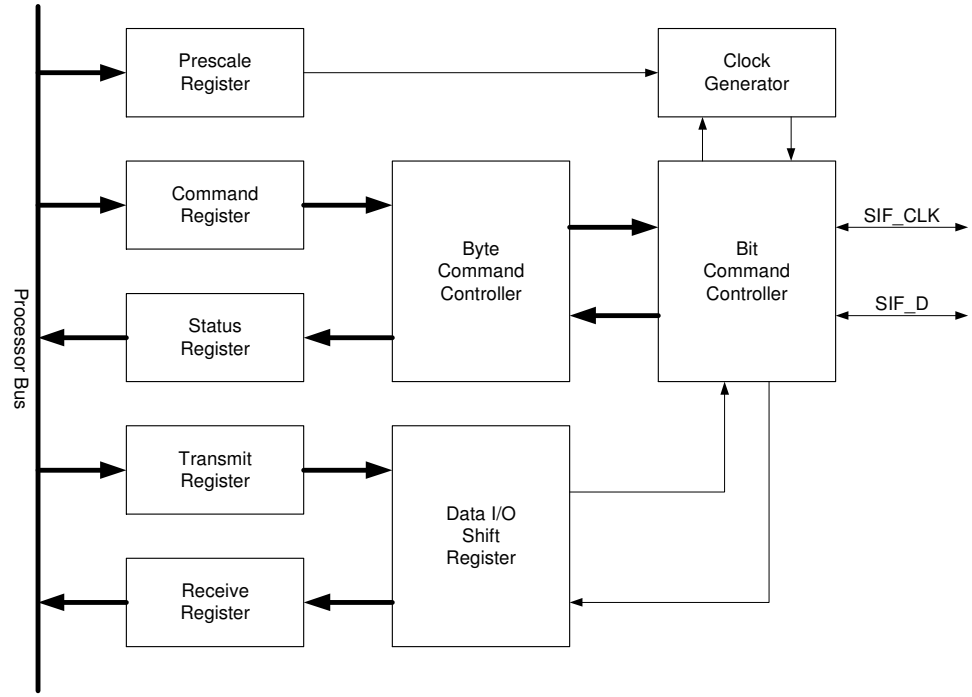


Figure 32: SIF Block Diagram

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 `vAHI_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 JN5121 supports both 7 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 bidirectional 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.

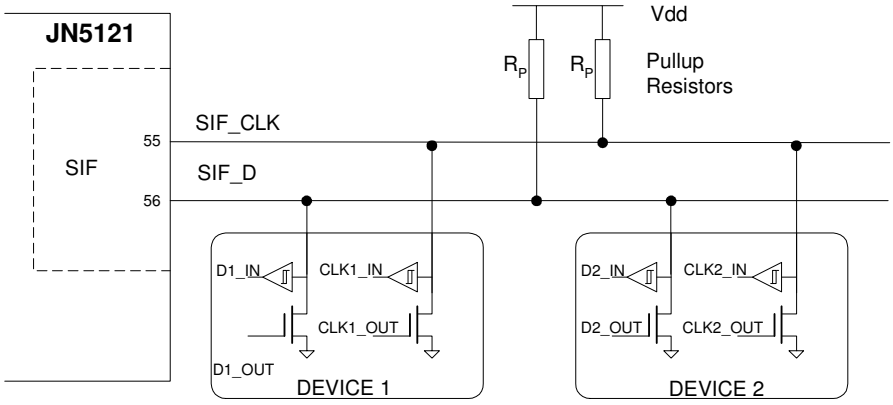


Figure 33: Connection Details

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. Therefore a high-to-low transition on the SF_CLK line causes all concerned devices to count off their low period. Once a devices clock input has gone low it will hold the SF_CLK line in that state until the clock high state is reached. Due to the wired-AND connection the SF_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.

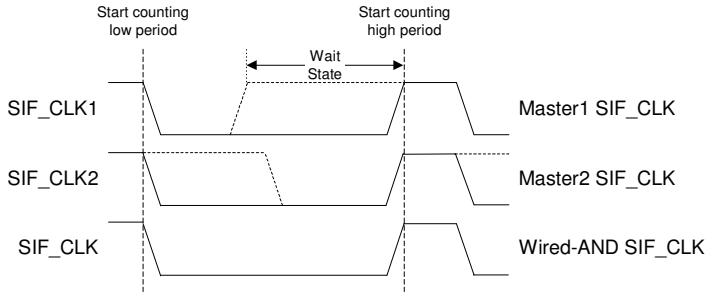


Figure 34: Multi-Master Clock Synchronization

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.

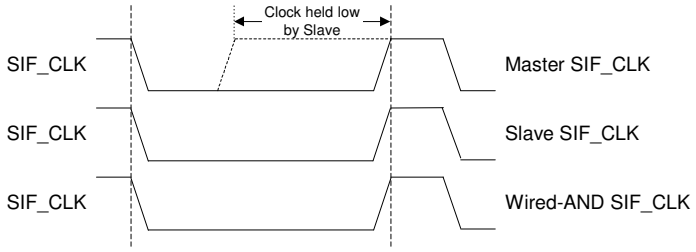


Figure 35: Clock Stretching

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 JN5121. 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.

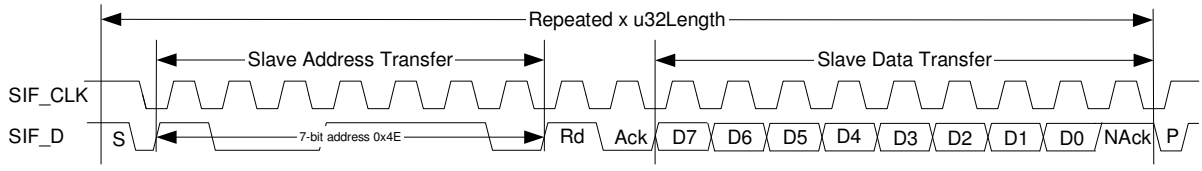


Figure 36: Read From Slave Device

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_SiCheckArbitrationLost()|bAHI_SiCheckRxNack())
        {
            /* 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_SiCheckArbitrationLost())
        {
            /* release bus & abort */
            vAHI_SiSetCmdReg(E_AHI_SI_NO_START_BIT_DISABLE, 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 JN5121 contains a number of analogue peripherals allowing the direct connection of a wide range of external sensors, switches and actuators.

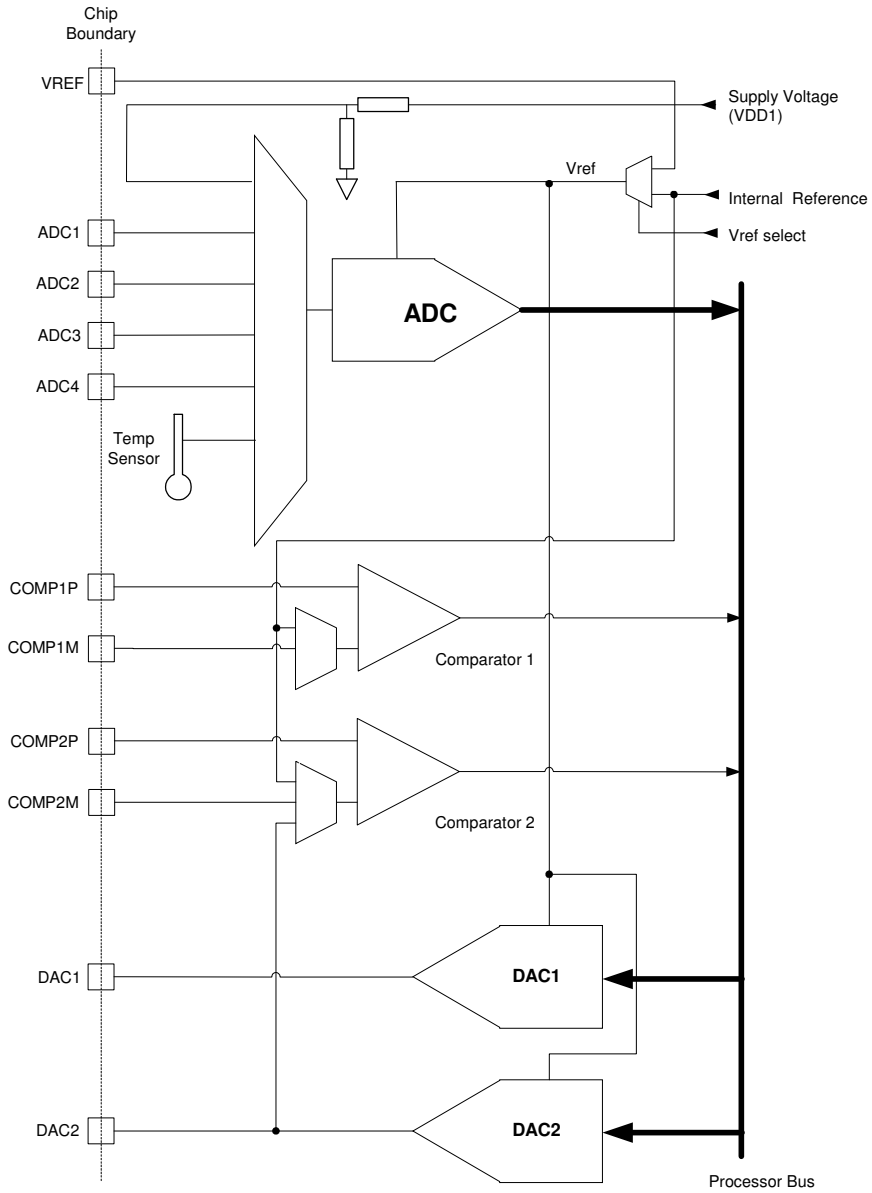


Figure 37: On-chip Analogue Peripherals

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 taken either from the internal voltage reference or from the external voltage applied to the VREF pin. For example an external reference of 0.8v supplied to VREF may be used to set the ADC range between 0V and 1.6V.

The device has programmable clock periods to allow a trade-off between conversion speed and resolution with the full 12-bit resolution being achieved with the 250kHz clock rate. See section 17.3.7 electrical characteristic for more details.

The input clock to the ADC is 16MHz and is divided down to 2MHz, 1MHz, 500kHz or 250kHz with a programmable divider. 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 (2 x sampling interval) + (14 x Clock periods), for example if the sampling period is set to 2 clock periods and the clock is set to 2MHz (the fastest clock rate), the conversion rate will be 2 x 2 + 14 = 18 clock periods or 111.1kHz.

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 10kΩ 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, for example 7 time constants gives an error of 0.1%, but 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 clock and sampling periods are 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 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 9 to 120 µsecs. Polling over this period would be wasteful of processor bandwidth. The result of a conversion can be read using `vAHI_AdcRead()` function.

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 which 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 17.3.10.

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 uses demonstrates data logging using the ADC1 input channel.

Programming Example

```
PRIVATE void vAdcDataLogger(uint16 *pau16DataBuffer, uint32 u32Length)
{
    /* configure Analogue Peripheral timings, interrupt & ref voltage */
    vAHI_ApConfigure(E_AHI_AP_INT_DISABLE,
                     E_AHI_AP_SAMPLE_2,
                     E_AHI_AP_CLOCKDIV_2MHZ,
                     E_AHI_AP_INTREF);

    /* configure & enable DAC */
    vAHI_AdcEnable(E_AHI_ADC_CONVERT_ENABLE,
                   E_AHI_ADC_GAIN_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
            pau16DataBuffer[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 9µs.

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 RC load up to that specified in section 17.3.8.

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 9 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 conversionThe `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)
{
    /* configure Analogue Peripheral timings, interrupt & ref voltage */
    vAHI_ApConfigure( E_AHI_AP_INT_DISABLE,
                      E_AHI_AP_SAMPLE_2,
                      E_AHI_AP_CLOCKDIV_2MHZ,
                      E_AHI_AP_INTREF);
    vAHI_DacEnable(E_AHI_DAC_1, FALSE, 0); /* configure & enable DAC */
    while(TRUE)
    {
        for (i=0;i<2048;i++)
        {
            vAHI_DacOutput(E_AHI_DAC_1, i); /* value to output */
            while(bAHI_DacPoll());     /* busy wait until conversion
complete */
        }
    }
}
```

15.3 Comparators

The JN5121 contains two analogue comparators COMP1 and COMP2. They 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 (COMP2 only) 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 individually 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 17.3.9. This mode is useful to wake up the JN5121 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 Power Domains

There are six power domains present in the device allowing different parts to be turned off to save power under different operating conditions. These domains are as follows:

- The supply power domain is directly powered from VDD1, VDD2 that supplies the wake-up timers and controller, DIO blocks, Comparators, low-power RAM, 32kHz RC oscillator and bandgap reference. This domain remains powered as long as the external supply is maintained
- The Application Logic domain comprising the SPI interface, CPU, and ROM is powered from an on-chip regulator. It is powered off during sleep and powered on at wakeup
- The Analogue Peripheral domain comprises the ADC, DACs and the temperature sensor is powered from an on-chip regulator. It is powered off during sleep and optionally powered on under software control
- Memory (RAM) power domain provides the ability to power the CPU RAM during sleep periods in order to keep the memory contents. It is powered from an on-chip regulator that is on when the JN5121 is not in sleep and optionally, under software control powered off during sleep
- The Transceiver Logic domain, comprising the Baseband Controller, Modem and Encryption coprocessor, is powered from an onchip regulator. It is typically under software control and powered when wireless communications is required
- The Radio power domain supplies the radio interface. It is powered during transmit and receive and controlled by the baseband processor.

16.2 Sleep Modes

Sleep modes enable the application to shut down unused functions in the device, thereby saving power. The JN5121 provides three sleep modes allowing the user to tailor the power consumption to the application's requirements.

The state of the JN5121 pins during sleep are described in section 2.2. The DIO pins retain their input or output status and their output value for the sleep period.

16.2.1 CPU Doze

Whilst in doze mode CPU operation is stopped but it 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.

16.2.2 Sleep

The JN5121 enters sleep under CPU control using the `vAHI_PowerDown()` function. All power domains are turned off except the supply power domain and optionally the memory domain, determined by `vAHI_MemoryHold()`. A wakeup event, caused by an interrupt from the wakeup timers, DIO pins or analogue comparator inputs bring the device out of sleep. Wakeup from sleep is described in detail in section 16.3.

16.2.3 Deep Sleep

Deep sleep mode gives the lowest power consumption as all switchable power domains are off and functions in the supply power domain, including the 32kHz oscillator are stopped. It is entered by executing the `vAHI_PowerDown()` function. This mode can only be exited by a power down or hardware reset on the RESETN pin.

16.3 Wakeup Events

Wakeup events are interrupts that can be used to restart the JN5121 from sleep mode. These interrupts are powered from the supply power domain which remains powered during sleep mode. There are three sources of wakeup events; transitions on DIO inputs, expiry of wakeup timers and comparator events. Only one wakeup will occur even if multiple sources were triggered. Software should remove the pending wakeup events prior to requesting a power-down in order to avoid a wakeup event generated during the previous awake period persisting and re-awakening the device immediately it goes to sleep.

Wakeup has a similar sequence of events to the reset process described in section 6.1. The 16MHz oscillator is started up and, once stable, the application power domain is powered and the CPU reset removed. Software determines a reset from sleep and commences the wakeup process.

16.3.1 Wakeup Timer Event

The JN5121 contains two 32-bit wakeup timers, which 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 JN5121 Hardware Peripheral API Reference Manual [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_DioIntEnable()` function the type of transition can be specified (rising or falling edge) by using the `vAHI_DioIntEdge()` function. Even when groups of DIO lines are configured to be used for 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 comparators 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 COMP1N and COMP2N or the internal voltage reference. The ability to wakeup when continuously monitoring analogue signals is useful in ultra-low power applications. The JN5121 can remain in sleep mode until the voltage drops below a threshold and then be woken up to deal with the alarm condition.

17 Electrical Characteristics

17.1 Maximum ratings

Exceeding these conditions will 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, COMP1P, COMP1M, RFP, RFM,	-0.3V	VB_xxx + 0.3V
Voltage on analogue pins VREF, ADC1-4, DAC1-2, 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	-40ºC	150ºC

17.2 DC Electrical Characteristics

17.2.1 Operating Conditions

Supply	Min (V)	Max (V)
VDD1, VDD2	2.2	3.6
Ambient temperature range	-40ºC	85ºC

17.2.2 DC Current Consumption

VDD = 2.2-3.6V, -40 to +85 deg. C

Mode:	Min	Typ	Max	Notes
Deep sleep		<1uA		
Sleep		<5uA		
CPU running and peripherals enabled		9mA		CPU running @ 16MHz
CPU running, peripherals and baseband enabled		12mA		CPU running @ 16MHz
CPU doze, peripherals enabled, baseband processing packets, radio transmit		35mA		
CPU running, peripherals enabled, baseband processing packets, radio transmit		40mA		CPU running @ 16MHz
CPU running, peripherals enabled, baseband processing packets, radio receive.		50mA		CPU running @ 16MHz
Comparator		72uA		
Comparator (Low power mode)		1.3uA		
ADC		700uA		
DAC		300/410uA		One/both
Temperature sensor		10uA		

17.2.3 I/O Characteristics

VDD = 2.2 to 3.6V, -40 to +85 deg. C

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

17.3 AC Characteristics

17.3.1 Reset

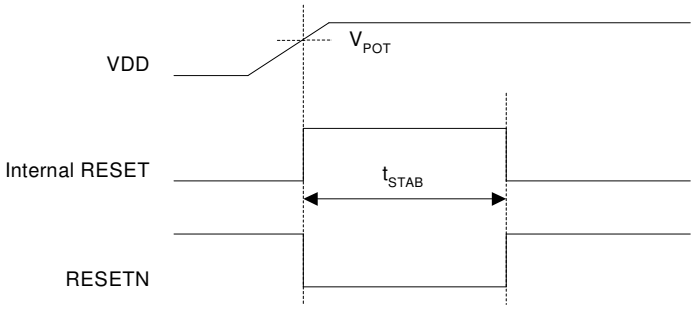


Figure 38: Power-on Reset

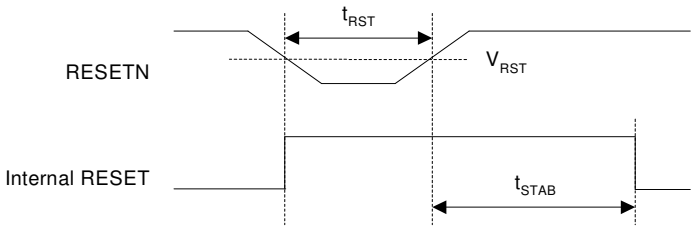


Figure 39: External Reset

Parameter	Min	Typ	Max	Notes
External Reset pulse width	1 µsec			Assumes internal pullup resistor value of 100K worst and ~5pF external capacitance.
External Reset threshold voltage	VDD2 x 0.7V			
Internal Power-on Reset threshold voltage (V_{POT})		2.1V 2.3V 2.45V		VDD2 = 2.2V VDD2 = 3.0V VDD2 = 3.6V Note 1

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

17.3.2 SPI Timing

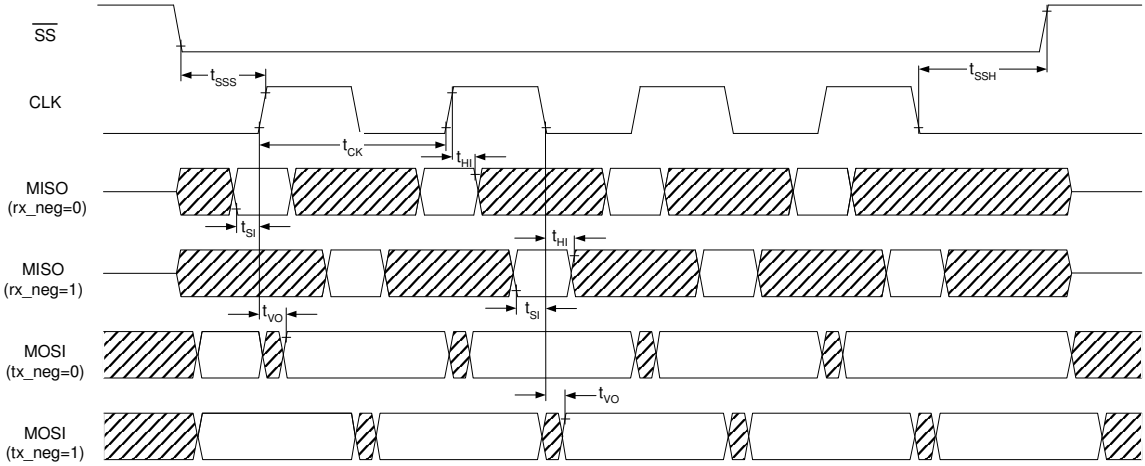


Figure 40: SPI Timing (Master)

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

Select set-up period	t_{SSS}	10	-	nsec
Select hold period	t_{SSH}	10	-	nsec

17.3.3 Two-wire serial interface

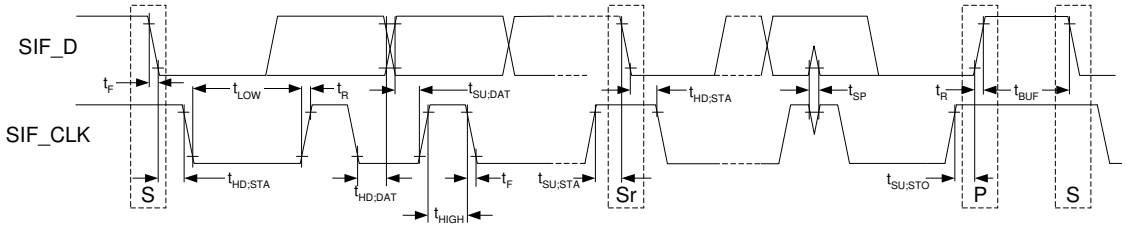


Figure 41: Two-wire serial Interface Timing

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	-	µsec
LOW period of the SIF_CLK clock	t_{LOW}	1.3	-	µsec
HIGH period of the SIF_CLK clock	t_{HIGH}	0.6		µsec
Set-up time for repeated START condition	$t_{SU:STA}$	0.6	-	µsec
Data hold time SIF_D	$t_{HD:DAT}$	0	0.9	µsec
Data setup time SIF_D	$t_{SU:DAT}$	100	0	µsec
Rise Time SIF_D and SIF_CLK	t_R	20+0.1Cb	300	nsec
Fall Time SIF_D and SIF_CLK	t_F	20+0.1Cb	300	nsec
Set-up time for STOP condition	$t_{SU:STO}$	0.6	-	µsec
Bus free time between a STOP and START condition	t_{BUF}	1.3	-	µsec
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	nsec

17.3.4 Power Down and Wake-Up timings

Parameter	Min	Typ	Max	Notes
Wake up from Deep Sleep		2.5 + 0.84* program size in kBytes ms		
Wake up from Sleep (memory not held)		2.5 + 0.84* program size in kBytes ms		
Wake up from Sleep (Memory held)		2.5ms		
Wake up from Processor Doze mode		0.5us		

17.3.5 32kHz Oscillator

VDD = 2.2 to 3.6V, -40 to +85 deg. C

Parameter	Min	Typ	Max	Notes
Current consumption of cell and counter logic		5uA 3.8uA 3.0uA		3.3V 2.5V 2.2v
32kHz clock native accuracy	-30%		+30%	
Calibrated 32kHz accuracy		+/-40ppm		Dependant on crystal accuracy
Un-calibrated variation with temperature		8Hz/ deg. C		
Un-calibrated variation with VDD2		150Hz/V		

17.3.6 16MHz Crystal Oscillator

VDD = 2.2 to 3.6V, -40 to +85 deg. C

Parameter	Min	Typ	Max	Notes
Current consumption		150uA		Excluding bandgap ref.
Start – up time		2.5mS		Assuming xtal with ESR of 40ohms and CL=9pF. External caps =12pF
Input capacitance		2pF		Bondpad and package
Transconductance		1.15mA/V		
DC voltages, XTALIN, XTALOUT		410mV		
External Capacitors (C1 & C2)		12pF		Total external capacitance needs to be 2*CL. , allowing for stray capacitance from chip, package and PCB

17.3.7 Analogue to Digital Converters

VDD = 3.0V, VREF = 1.2V, -40 to +85 deg. C

Parameter	Min	Typ	Max	Notes
Resolution			12 bits	250KHz Clock
Current consumption		700uA		

Parameter	Min	Typ	Max	Notes
Integral nonlinearity		+/-2 LSB		
Differential nonlinearity			+/-1 LSB	Guaranteed monotonic
Offset error		+/- 20 mV		
Gain error		TBA		
Internal clock		2MHz, 1MHz, 500kHz, 250kHz		16MHz input clock, programmable prescaler
No. internal clock periods to sample input		2,4,6 or 8		Programmable
Conversion time	9us			2MHz Clock with sample period of 2
Effective No. bits with Clock 250KHz 500KHz 1MHz 2MHz		 TBA TBA TBA TBA		
Input voltage range			0 to Vref or 0 to 2*Vref	Switchable
Vref (Internal)	1.15	1.2	1.25	Bandgap voltage
Vref (External)	0.8	1.2	1.6	Allowable range into VREF pin
Input capacitance		8pF		In series with 5K ohms

17.3.8 Digital to Analogue Converters

VDD = 3.0V, VREF = 1.2V, -40 to +85 deg. C

Parameter	Min	Typ	Max	Notes
Resolution		11bits		
Current consumption:		300uA (single) 410uA (both)		
Integral nonlinearity		+/-1 LSB		
Differential nonlinearity			+/-1 LSB	Guaranteed monotonic
Offset error		+/- 20 mV		
Gain error				
Internal clock		2MHz, 1MHz, 500kHz, 250kHz		16MHz input clock, programmable prescaler

Parameter	Min	Typ	Max	Notes
Output settling time to 0.5LSB	5uS			With and 10K ohms & 20pF load
Minimum Update time	9us			2MHz Clock with sample period of 2
Output voltage swing		0 to VREF or 0 to 2xVREF		Switchable
Vref (Internal)	1.15V	1.2V	1.25V	Bandgap voltage
VREF (External)	0.8V	1.2V	1.6V	Allowable range into VREF pin
Resistive load	10kΩ			To gnd
Capacitive load			20pF	

17.3.9 Comparators

VDD = 2.2 to 3.6V -40 to +85 deg. C

Parameter	Min	Typ	Max	Notes
Analog response time (normal)		60ns	73ns	+/- 250mV overdrive
Total response time (normal) including delay to Interrupt controller			73ns +125nS	Digital delay can be up to a max. of two 16MHz clock periods
Analog response time (low power)		2.5us	4.2us	+/- 250mV overdrive No digital delay
Hysteresis		10mV 20mV 40mV		Programmable in 3 steps and zero.
Vref (Internal)	1.15V	1.2V	1.25V	
Common Mode input range	0V		Vdd	
Current (normal mode)		72uA		
Current (low power mode)		1.3uA		
Analog response time (normal)		60ns	73ns	+/- 250mV overdrive

17.3.10 Temperature Sensor

Parameter	Min	Typ	Max	Notes
Operating Range	-40°C	-	85°C	
Sensor Gain	-1.55 mV/°C	-1.6 mV/°C	-1.63 mV/°C	
Accuracy	-	-	±10°C	
Non-linearity	-	-	2°C	
Output Voltage Range	620 mV	750 mV	845 mV	
Resolution	0.19°C/LSB	0.183°C/LSB	0.179°C/LSB	0 to Vref ADC I/P Range

17.3.11 Radio Transceiver

This device is fully compatible with the IEEE802.15.4 standard and offers the following improved RF characteristics:

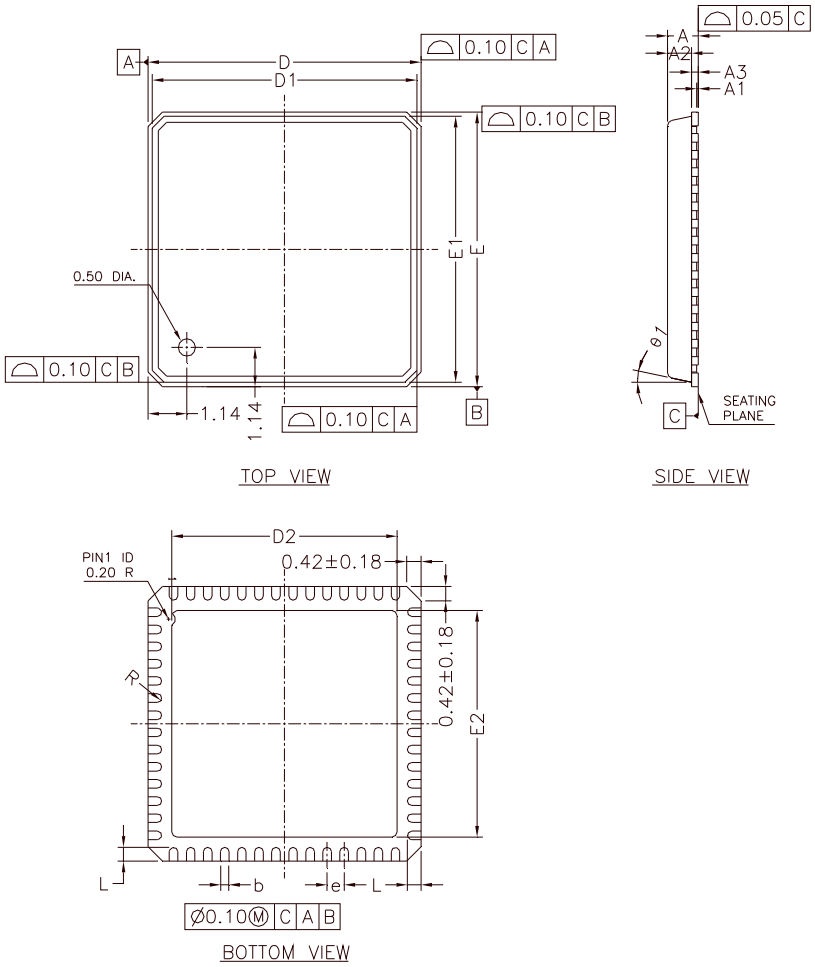
All RF characteristics are measured single ended and include the losses of a ceramic balun.

Parameter	Min	Typical	Max	Notes
Frequency range	2.4 GHz		2.4835GHz	
Receiver Characteristics				
Receive sensitivity		-93dBm		Nominal for 1% PER, as per 802.15.4 section 6.5.3.3
Maximum input signal		-10dBm		For 1% PER, measured as sensitivity
Adjacent channel rejection		20dB		For 1% PER with wanted signal 3dB above sensitivity, as per 802.15.4 section 6.5.3.4
Alternate channel rejection		40dB		For 1% PER with wanted signal 3dB above sensitivity, as per 802.15.4 section 6.5.3.4
Other in band rejection		40dB		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)		-57dBm -47dBm		30MHz - 1GHz 1 - 12GHz

Parameter	Min	Typical	Max	Notes
Intermodulation protection		35dB		For 1% PER at with wanted signal 3dB above sensitivity. Modulated Interferers at 2 & 4 channel separation
RSSI range		-95 to -10 dBm		Linear within +/-2dB. Available through Hardware API
Transmitter Characteristics				
Transmit power		1dBm		Nominal
Output power control range		-30dB		in 5 6dB steps
Spurious emissions (TX)		-36dBm -47dBm -43dBm		30MHz to 1GHz 1.8-1.9GHz & 5.15-5.3GHz 1GHz-12.5GHz
EVM			25%	At maximum output power
Transmit Power Spectral Density		-20dBc		At 3.5MHz offset, as per 802.15.4, section 6.5.3.1
RF Port Characteristics				
Type				Differential
Impedance		200ohm		2.4-2.5GHz

Appendix A Mechanical and Ordering information

A.1 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	
υ1	0°	------	12°
R	0.09	------	------
Tolerances of Form and Position			
aaa		0.10	
bbb		0.10	
ccc		0.05	

A.2 Ordering Information:

Part numbering:

JN5121(-XXX) - Y1Y2 - Y3Y4

XXX is an optional identifier for customer specific masked ROM versions of the chip.

Y1: Temperature Range:

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

Y2: Screening Options

G Standard Jennic Product Screening

Y3: Package Variant

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

Y4: Packing Options

R Trays

T Tape and reel

Ordering codes:

Shipping Format:	Ordering Code	Notes
Tape & Reel	JN5121-xxx-IG-AT	Multiples of 2,500 on a 13” reel
Tray	JN5121-xxx-IG-AR	Up to 348 devices per tray

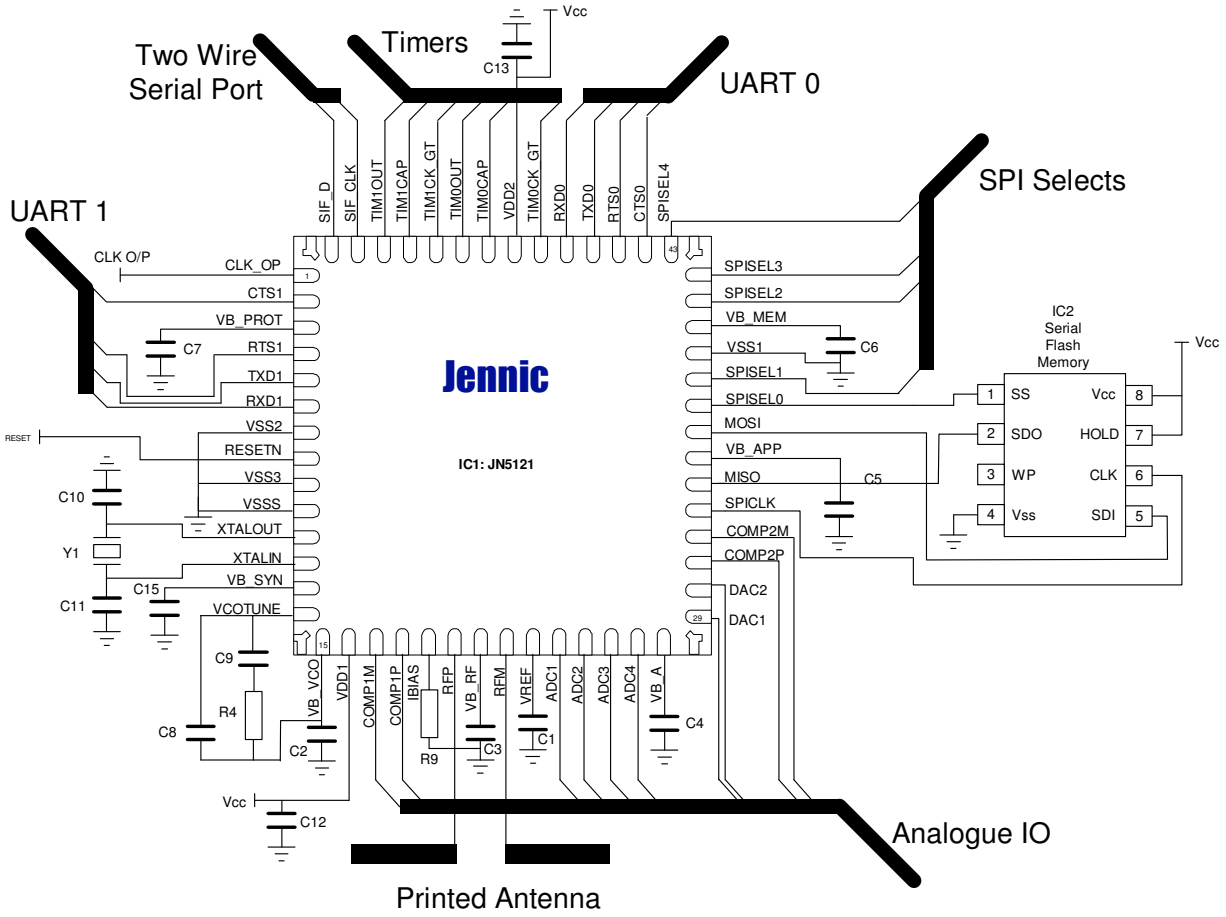
Appendix B Development Support

B.1 Crystal Requirements

16MHz Crystal Requirements

Parameter	Min	Typ	Max	Notes
Crystal Frequency		16MHz		
Crystal Tolerance		+/-40ppm		Including temperature and aging
Crystal ESR	20Ω		60Ω	
Crystal Load Capacitance CL		9pF		
External Capacitors (C1 & C2)		12pF		Total external capacitance needs to be 2*CL. , allowing for stray capacitance from chip, package and PCB

B.2 Applications Schematic



Components	Values
C1, C2, C3, C4, C5, C6, C7, C13, C12	100nF
C10, C11	12pF
C9	3n3F
C8	680pF
R4	4k7
R9	43k
Y1	16MHz Xtal
IC1	JN5121
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 Hardware Peripheral Library Reference Manual

[3] JN-RM-2002 Stack Software Reference Manual

[4] JN-AN-1003 Boot Loader Operation

[5] JN-RM-2010 CPU Reference Manual

[6] JN-UG-3001 SDK Installation User Guide

Version Control

Version	Notes
1.0	26th August 2005 Final Draft
1.1	Modifications to pin list and pin configurations
1.2	

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.

Contact Details

UK 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

Japan Sales Office
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

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

USA Sales Office
1322 Scott Street
Point Loma, CA 92106, USA
Tel: +619 223 2215
Fax: +619 223 2081
info@us.jennic.com
www.jennic.com

