iND80212 “Heimdall Slave”

indie’s highly integrated solution for Pyro InfraRed (PIR) and remote switch home security sensors.

11/16/15 **Preliminary Data sheet**

1.0 REVISION HISTORY

Table 1 Revision History			
Rev #	**Date**	**Action**	**By**
0.1	20 Jan 2011	Initial Draft	CG
0.2	27 Jan 2011	Second Draft	CG
0.3	04 Mar 2011	Third Draft	CG
0.4	24 Jun 2011	Correction (temporary, until next silicon, of the addresses and access mode of registers GPIOCTRL1,2 and 3	CG
0.5	01 Jun 2011	Correction in the R/W access of registers PLLCTRL0 and PLLCTRL1, which are R/W and not reserved.	CG
0.6	27 Jun 2011	General Update	DKM
0.7	30 Nov 2011	Update for C1 silicon	DDK
0.8	19 Nov 2013	Update measurement data and descriptions	JEA
0.9	19 Dec 2014	Updated Table 2, Added Active Attenuator, Changed Template to indie Semiconductor, Modified Memory Map Tables, Modified ADC and Temp Sections, Updated BoR, Wakeup Clock Speed, Keep Awake Feature, Code Protection NOTE: Pin6 in Pinout diagram needs to be updated to TMS	JEA, DDK
0.91	9 Mar 2015	Renamed Pin 6 from GND to Keep Awake to reflect C4 Change Update Pinout diagram	JEA
1.0	16 Nov 2015	Standard product release, table and figure format fix	CR

2.0 TABLE OF CONTENT

3.0 LIST OF TABLES

4.0 LIST OF FIGURES

5.0 REGISTER CONVENTION AND MEMORY MAP

5.1 REGISTER CONVENTION

Several registers will be defined and explained throughout this document. The general format of the description of the registers is as follows:

Name of the Register		Starting Address (Hex)			Reset or Default Value (Hex)		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit_Name	Bit_Name	Bit_Name	Bit_Name	Bit_Name	Bit_Name	Bit_Name	Bit_Name
MSB							LSB

Where R/W is the read and write permissions of the specific bit. An example:

TMR0REG		0x50020000			0x00000000		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
T7	T6	T5	T4	T3	T2	T1	T0
T15	T14	T13	T12	T11	T10	T9	T8
T23	T22	T21	T20	T19	T18	T17	T16
T31	T30	T29	T28	T27	T26	T25	T24
MSB							LSB
Bit31-0 **T[31:0]**: Timer Register initial value register.							

The name of this register is TMR0REG (Timer 0 Register). It is a 32-bit register, located at address 0x50020000, 0x50020001, 0x5002002 and 0x50020003. The first row of the data (T[7:0]) corresponds to address 0x50020000 and the fourth row of the data (T[31:24]) corresponds to address 0x50020003.

5.2 MEMORY MAP

Table 2 : System Memory Map			
Address	**Memory**	**Description**	**Reference**
0x00000000 - 0x00027FFF	Flash	160k-byte Flash	N/A
0x20000000 - 0x20001FFF	SRAM	8k-byte SRAM	N/A
0x50000000 – 0x5000007F	Peripheral	128-byte peripheral fast access	Table 3
0x50000080 - 0x50000085	Peripheral	6-byte Block Transfer control	N/A
0x50010000 - 0x5001FFFF	Peripheral	64k-byte peripheral slow access	Table 4
0x50020000 - 0x5002001F	Peripheral	32-byte timer control	N/A
0x50020020 - 0x50020047	Flash	40-byte Flash program/erase control	N/A
0xE0000000 - 0xE00FFFFF	Private peripheral bus	ARM peripherals	N/A
0xF0000000 - 0xF0001FFF	System ROM tables	ARM core IDs	N/A

Table 3 : Peripheral Fast Access Memory Map			
Address	**Peripheral**	**Description**	**Reference**
0x50000000 - 0x50000001	PIR	PIR configuration register	
0x50000002	PMU	Clock control register	
0x50000004 - 0x5000000F	RF	RF control register	
0x50000010	PMU	RC Oscillator calibration	
0x50000011	PMU	Wakeup Time register	
0x50000012 - 0x50000014	ADC	ADC control	
0x50000015	PMU	System Enable	
0x50000016 - 0x50000019	GPIO	GPIO control	
0x5001100A - 0x5001100C	GPIO	GPIO control	

Table 4 : Peripheral Slow Access Memory Map			
Address	**Memory**	**Description**	**Reference**
0x50010000 - 0x50010001	XTAL/BG	XTAL and Bandgap configuration register	
0x50010002	GPIO	LED current trim register	
0x50010003 - 0x50010007	RF	RF configuration register	
0x50010008 - 0x5001000A	ADC	ADC configuration register	
0x5001000B	PMU	1.8V Regulator trim register	
0x5001000C	RF	RF trim register	
0x5001000D	GPIO	GPIO configuration register	
0x50010010 - 0x50010013	GPIO	GPIO configuration register	

6.0 GENERAL DESCRIPTION

HeimdallSlave integrates an ARM Cortex-M0 low cost 32-bit microcontroller containing 160kB of flash program memory and 8kB of SRAM. It implements several general-purpose peripherals. Its main features are:

CPU Architecture:

- ARM Cortex-M0 processor running at 30 MHz (crystal) or 10 kHz (Internal auxiliary RC)
- System Tick Timer (SysTick – 24 bits, interruptible)
- Serial Wire Debugger
- Built-in Nested Vectored Interrupt Controller (NVIC)
- Programmable Watch-Dog Timer

Memory:

- 160kByte of Flash Program Memory
- 8kByte of SRAM
- Self-Programming
- Code Protection

Peripherals:

- 433 MHz ISM Transmitter supporting selectable ASK/OOK, FSK & DPSK modulation modes
- 8-bit ADC with 10 input channels, with selectable input references and input gain block
- 10 General purpose I/O ports, 2 of which has LED current sink capability
- Integrated fractional-N phase locked loop referenced to crystal oscillator
- Selectable 10kHz / ~250 kHz RC relaxation oscillator
- Red and Blue LED Drivers, with Charge Pump available for Blue LED
- Low Battery Detection
- Power Management
- Brown out Reset
- Temperature Sensor
- Wake-up Timer
- PIR Sensor Interface

Package:

- 4 x 4, 20 pin QFN package

7.0 PINOUT AND PACKAGE

7.1 PACKAGE OVERVIEW

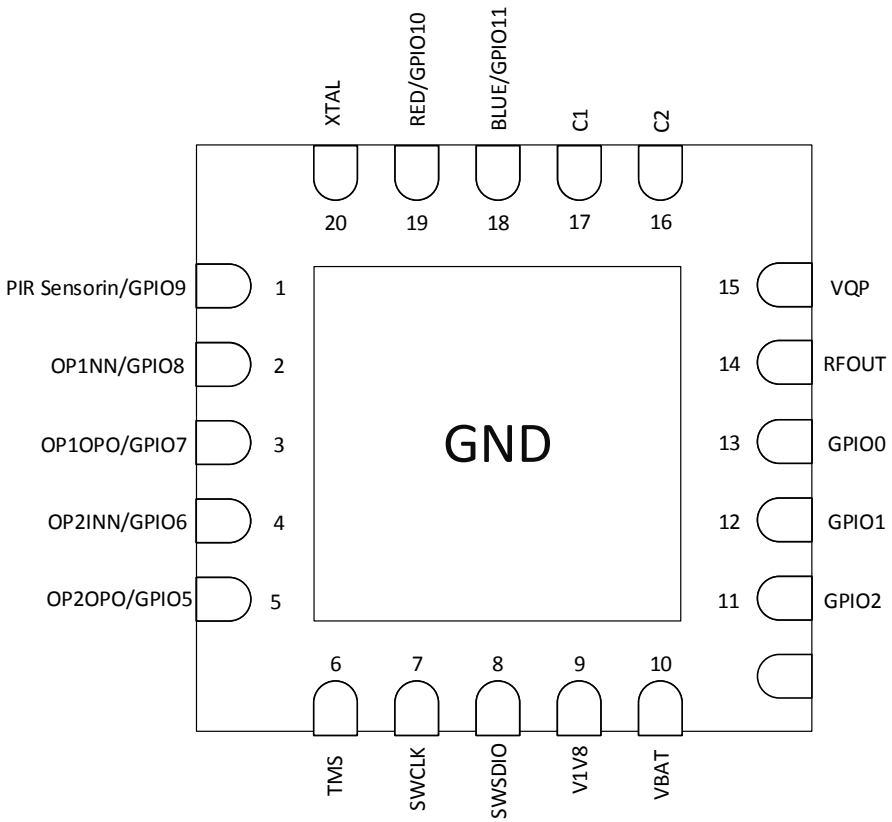
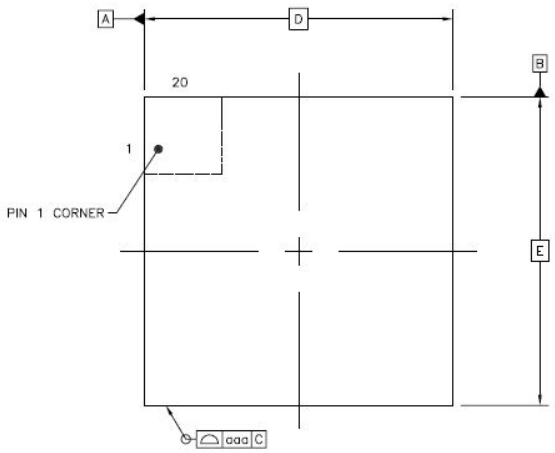


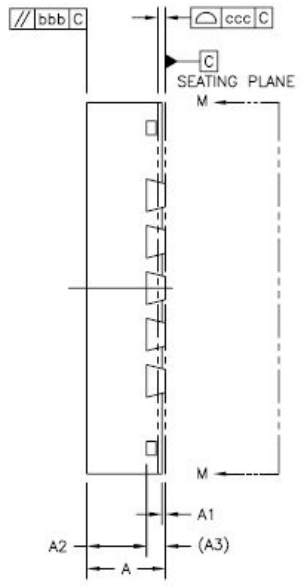
Figure 1: QFN 4mm X 4mm Pinout

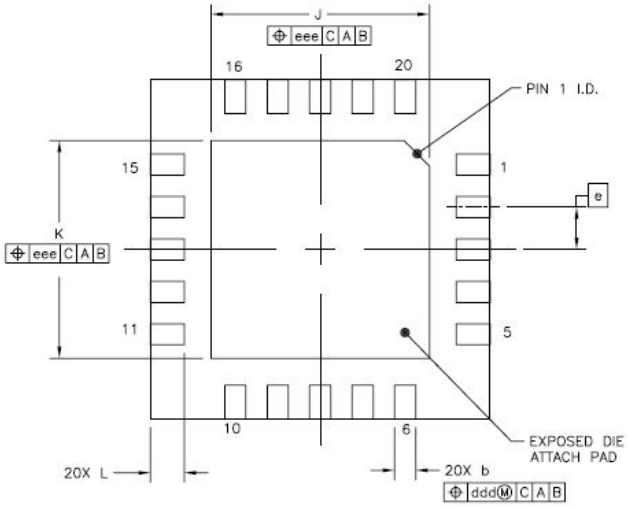
7.2 PACKAGE DIMENSIONS

DESCRIPTION		SYMBOL	MILLIMETER		
			MIN	NOM	MAX
TOTAL THICKNESS		A	0.8	0.85	0.9
STAND OFF		A1	0	0.035	0.05
MOID THICKNESS		A2	---	0.65	0.67
L/F THICHNESS		A3		0.203 REF	
LEAD WIDTH		b	0.2	0.25	0.3
BODY SIZE	X	D		4 BSC	
	Y	E		4 BSC	
LEAD PITCH		e		0.5 BSC	
EP SIZE	X	J	2.47	2.57	2.67
	Y	K	2.47	2.57	2.67
LEAD LENGTH		L	0.35	0.4	0.45
PACKAGE EDGE TOLERANCE		aaa		0.1	
MOLD FLATNESS		bbb		0.1	
COPLANARITY		ccc		0.08	
LEAD OFFSET		ddd		0.1	
EXPOSED PAD OFFSET		eee		0.1	



Top View





Bottom View

Figure 2: QFN 4mmX4mm 20-pin Package Dimensions

7.3 PINS DESCRIPTION

Table 5 Pin List

#	Name	Type	Description
1	PIRSensorin/ GPIO9	GPIO	Input to first gain/filter stage Op. Amp. of PIR sensor interface/GPIO
2	OP1INN/ GPIO8	GPIO	Negative input of first gain/filter stage Op. Amp./GPIO
3	OP1OPO/ GPIO7	GPIO	Output of first gain/filter stage Op. Amp./GPIO
4	OP2INN/ GPIO6	GPIO	Negative input of second gain/filter stage Op. Amp./GPIO
5	OP2OPO/ GPIO5	GPIO	Output of second gain/filter stage Op. Amp./GPIO
6	TMS	Test	TMS shall be connected to a Test Point. Holding this pin high upon PoR keeps Cortex in wake state See Keep Awake Feature
7	SWCLK	Digital Input	Serial Clock Input (Debugger) with pull-down resistor
8	SWSDIO	Digital IO	Serial Data (Debugger) with pull-down resistor
9	V1V8	Power Supply	1.8V regulator output to the Microcontroller
10	VBAT	Power Supply	Battery voltage
11	GPIO2	GPIO	General purpose Digital Input/Output pin
12	GPIO1	GPIO	General purpose Digital Input/Output pin
13	GPIO0	GPIO	General purpose Digital Input/Output pin
14	RFOUT	RF Output	TX RF output pin
15	VQP	Analog Output	Charge Pump Voltage output
16	C2	Analog Input	Charge Pump Capacitor pin 1
17	C1	Analog Input	Charge Pump Capacitor pin 2

Table 5 Pin List			
#	**Name**	**Type**	**Description**
18	BLUE/ GPIO11	GPIO	GPIO with PMOS current source mode to supply a BLUE LED from an internally generated supply
19	RED/ GPIO10	GPIO	GPIO with PMOS current source mode to supply a RED LED from VBAT
20	XTAL	Analog Input	crystal oscillator pin or external clock input pin
GND	GND	Supply	ground slug

8.0 ELECTRICAL CHARACTERISTICS

8.1 MAXIMUM RATINGS

Absolute maximum ratings are defined in the following table. The operation of the device above these conditions may have unknown consequences and it is not implied nor recommended.

Table 6 Maximum Ratings

Name	Min.	Typ.	Max.	Unit
Vbat voltage	-0.3		3.3	V
Analog Input voltage	-0.3		Vbat+0.3	V
Analog Output voltage	-0.3		Vbat+0.3	V
Digital Input voltage	-0.3		Vbat+0.3	V
GIO, GIO+	-0.3		Vbat+0.3	V
RF Output	-0.3		Vbat+0.3	V
1.8V Digital Output Voltage	-0.3		V1V8+0.3	V
Operating Temp.	-40		+85	°C
HBM (all pins)	-2		2	kV
CDM (all pins)	-200		200	V
MM (all pins)	-100		100	V

8.2 TYPICAL OPERATING CONDITIONS

Table 7 Recommended Operating conditions – MCU and all other Peripherals including RF					
Name	**Conditions**	**Min.**	**Typ.**	**Max.**	**Unit**
Vbat voltage		2.2	3.0	3.2	V
Operating Temp		-40	25	85	°C

Table 8 Recommended PIR Operating conditions					
Name	**Conditions**	**Min.**	**Typ.**	**Max.**	**Unit**
Vbat voltage		2.5	3.0	3.2	V
Operating Temp		-10	25	60	°C

8.3 CURRENT CONSUMPTION

Table 9 Typical Current Consumption, typical conditions					
Name	**Conditions**	**Min.**	**Typ.**	**Max.**	**Unit**
Sleep Mode	All circuits disabled			1	μA
Low Power operation	10 kHz oscillator running, micro active			10	μA
Micro Active	30 MHz oscillator running, uDIV=8			1.7	mA
Radio TX	Radio Active, Pout = +13 dBm		18		mA

9.0 DEVICE OVERVIEW

The overall structure of HeimdallSlave is depicted in Figure **3**

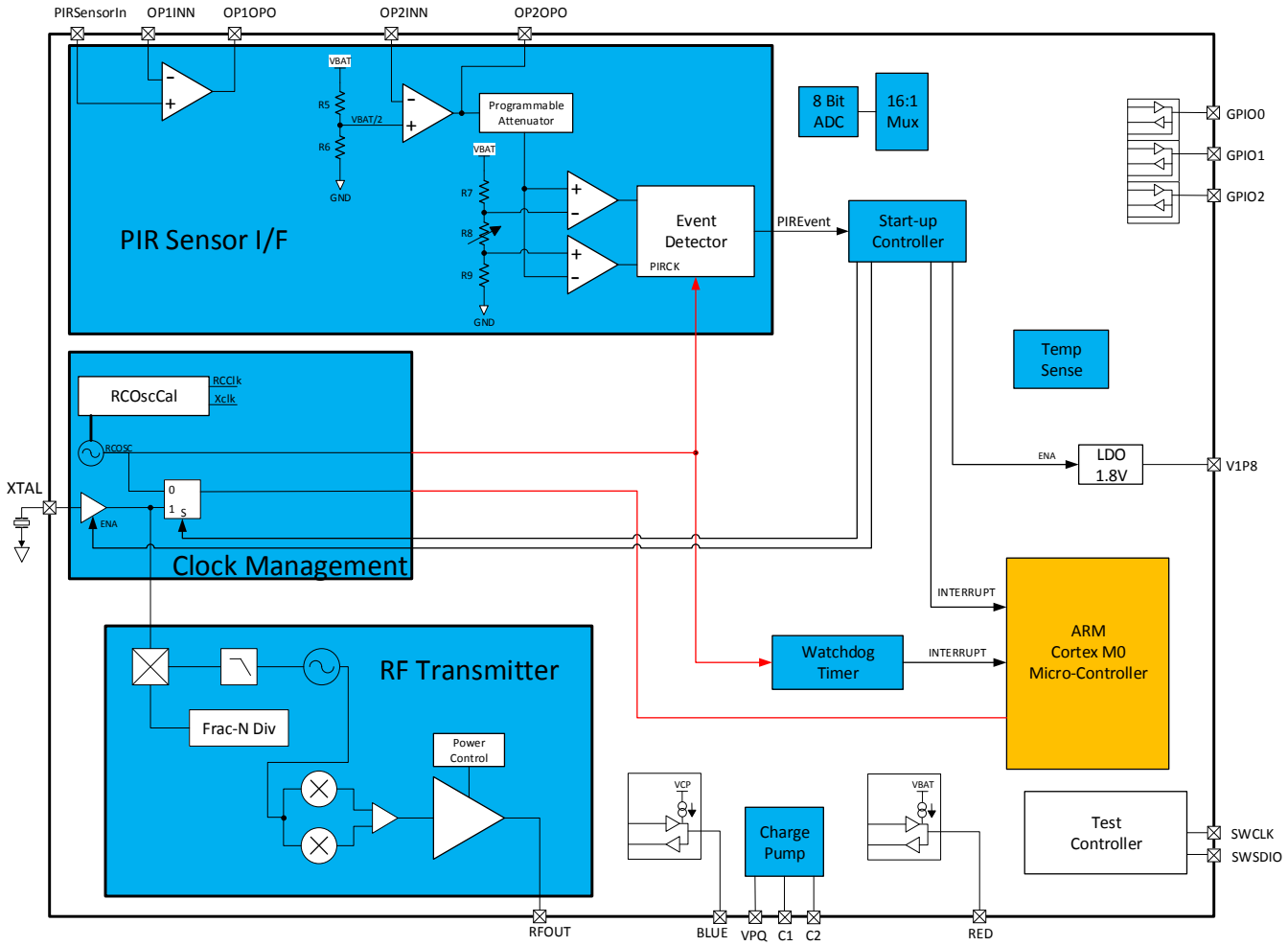


Figure 3**: HeimdallSlave Block Diagram**

9.1 HEIMDALLSLAVE

The HeimdallSlave ASIC contains the necessary hardware to realize a remote sensor interface with integrated radio transmitter. Figure 3 depicts a high-level block diagram of the device.

The subsystems are as follows, microcontroller, 300mHz – 450mHz transmitter, 8-bit general-purpose ADC, GPIOs, fractional-N LL, 10 kHz relaxation oscillator, LED drivers, Low Battery Detection Circuit, Power Management, Temperature Sensor, Wake up Timer, PIR sensor interface, including a charge pump to drive a blue LED when the battery is near end of life, and clock generation.

9.1.1 Microcontroller Subsystem

HeimdallSlave includes an embedded microcontroller subsystem, which is based on the ARM Cortex M0 core.

It includes a program flash memory of 160kBytes, and an SRAM of 8kBytes. It includes three 32-bit timers, plus a dedicated watchdog timer. Additionally, it includes a **N**ested **V**ector **I**nterrupt **C**ontroller (NVIC) to scheduled hardware interrupts, and a **W**akeup **I**nterrupt **C**ontroller (WIC), which enable the control of the various power modes.

9.1.2 Timers (0,1, and 2)

HeimdallSlave implements three identical timers: Timer0, Timer1 and Timer2. These timers use the system clock as clock source and once activated count up continuously. They start from the value initially loaded into the counting register (32-bit) and if enabled generate an interrupt upon rolling over (0xFFFFFFFF → 0x00000000).

9.1.3 Timers Registers

There are two basic registers associated with each of three timers:

Register 1 32-bit Timer0 initial value register							
TMR0REG		0x50020000			0x00000000		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
T7	T6	T5	T4	T3	T2	T1	T0
T15	T14	T13	T12	T11	T10	T9	T8
T23	T22	T21	T20	T19	T18	T17	T16
T31	T30	T29	T28	T27	T26	T25	T24
MSB							LSB
Bit31-0 **T[31:0]**: Timer Register initial value register.							

Register 2 Timer0 Control register							
TMR0CTRL		0x50020004			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	TSTART
MSB							LSB
Bit0 **TSTART**: Timer enable bit. 0 = Timer not running 1 = Timer running							

Register 3 32-bit Timer1 initial value register							
TMR1REG		0x50020008			0x00000000		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
T7	T6	T5	T4	T3	T2	T1	T0
T15	T14	T13	T12	T11	T10	T9	T8
T23	T22	T21	T20	T19	T18	T17	T16
T31	T30	T29	T28	T27	T26	T25	T24
MSB							LSB
Bit31-0 **T[31:0]**: Timer Register initial value register.							

Register 4	Timer1 Control register						
TMR1CTRL		0x5002000C			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	TSTART
MSB							LSB

Bit0 **TSTART**: Timer enable bit.
0 = Timer not running
1 = Timer running

Register 5	32-bit Timer2 initial value register						
TMR2REG		0x50020010			0x00000000		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
T7	T6	T5	T4	T3	T2	T1	T0
T15	T14	T13	T12	T11	T10	T9	T8
T23	T22	T21	T20	T19	T18	T17	T16
T31	T30	T29	T28	T27	T26	T25	T24
MSB							LSB

Bit31-0 **T[31:0]**: Timer Register initial value register.

Register 6	Timer2 Control register						
TMR2CTRL		0x50020014			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	TSTART
MSB							LSB

Bit0 **TSTART**: Timer enable bit.
0 = Timer not running
1 = Timer running

9.1.4 Timer Operation

The operation of the timers is quite straightforward. Load the initial counter register, enable the timer and either check (polling mode) the current value of the counter register or enable the interrupt and process it inside the interrupt service routine.

NOTE: Inside the interrupt the application code must reload the timer counting register.

Code Example1: Enable Timer1 to count from 0xFFFF0000 and to generate interrupt:

```
TMR_Config( 1, TIMERON, 0xFFFF0000);    //Enable timer1 to count up from
0xFFFF0000
NVIC_EnableIRQ( TIMER1_IRQn );        //Enable Timer1 interrupt


void Timer1_Handler( void )
{
  *TMR1REG = 0xFFFF0000;             //Reload Register
  //**** From this point application code inside ISR****
}
```

9.2 WATCH DOG TIMER

HeimdallSlave implements a WDT (**W**atch **D**og **T**imer) that can operate in one of two basic ways:

Interrupt Mode: In the event of a WDT rollover an interrupt will be executed.

Reset Mode: In the event of a WDT rollover the microcontroller will reset.

9.2.1 WDT Registers

The Watch Dog Timer implements two 32-bit registers:

Register 7 Watch Dog Timer control register (32-bit)							
WDTCTRL		0x50020018			0x0000000x		
Reserved	Reserved	Reserved	R/W	R/W	R/W	R/W	R/W
-	-	-	WDTPRES1	WDTPRES0	RSTFLAG	RESETEN	WDTEN
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
MSB							LSB

Bit4-3 WDTPRES1: WDTPRES0: WDT Prescaler:

= 213/SystemClock

= 219/SystemClock

10 = 222/SystemClock

11 = 232/SystemClock

Bit2 RSTFLAG: Reset Flag. This flag is set by the system at the initialization if the initialization was caused by a reset triggered by the WDT. The bit can be de-asserted by the application.

Bit1 RESETEN: Reset enable. If enabled a WDT time-out will force the microcontroller to reset. This bit can be asserted but it cannot be de-asserted.

Bit0 WDTEN: WDT enable. This bit can be asserted but it cannot be de-asserted. It means that once the WDT is enabled it cannot be turned off until a Reset or Power-On Reset occurs.

For instance, a system running from a 30 MHz Crystal with WDTPRES[1…0] = 10 will trigger the WDT after approximately 0.14seconds if not cleared properly and in time by the application.

Register 8 WDT Clear register (32-bit)							
WDTCLR		0x5002001C			0x0000000x		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
WCLR7	WCLR6	WCLR5	WCLR4	WCLR3	WCLR2	WCLR1	WCLR0
WCLR15	WCLR14	WCLR13	WCLR12	WCLR11	WCLR10	WCLR9	WCLR8
WCLR23	WCLR22	WCLR21	WCLR20	WCLR19	WCLR18	WCLR17	WCLR16
WCLR31	WCLR30	WCLR29	WCLR28	WCLR27	WCLR26	WCLR25	WCLR24
MSB							LSB

Bit31-0 **WCLR[31:0]**: Clear Register. To clear the WDT counting the following words must be written in this order and without any other instruction between then:

0x3C570001

0x007F4AD6

Warning: Programming WDTCLR with other values or in the wrong order will cause the watchdog to throw an interrupt or reset the system.

Example Code: Setting and cleaning the WDT. (Interrupt mode with a time of 2^22)

```
WDT_Config(WDT_INT, WDT22);          //Enable WDT in interrupt mode (2^22 system
clock cycles)
WDT_Clear();                         //Clear WDT
```

9.3 INTERRUPT VECTORS

HeimdallSlave implements an interrupt vector defined in Table 10:

Table 10 Interrupt Vectors			
Cortex M0 Specific Exceptions			
Name	**Number**	**Comments**	**Required Interrupt Handler (Function)**
HardFault_IRQn	-13	HardFault handler*	HardFault_Handler (void)
SVCall_IRQn	-5	Supervisory call*	
PendSV_IRQn	-2	Interrupt-driven request for system level service*	
SysTick_IRQn	-1	SysTick Timer interrupt	void SysTick_Handler(void)
HeimdallSlave Specific Exceptions			
Name	**Number**	**Comments**	**Required Interrupt Handler (Function)**
TX_RELOAD_IRQn	0	TX Reload Handler	void BrownOut_Handler (void)
TX_DONE_IRQn	1	TX Done Handler	void ClkMon_Handler (void)
IRQ2-8	2-8	Reserved	
IRQ9_IRQn to IRQ15_IRQn	9-15	Reserved	void Default_IRQ_Handler(void)
TIMER0_IRQn	16	Timer0 interrupt	void Timer0_Handler (void)
TIMER1_IRQn	17	Timer1 interrupt	void Timer1_Handler (void)
TIMER2_IRQn	18	Timer2 interrupt	void Timer2_Handler (void)
WATCHDOG_IRQn	19	Watchdog timer interrupt	void Watchdog_Handler (void)

*NOTE: For more information see Cortex-M0 Devices – Generic Users Guide (ARM DUI 0497A (ID112109)) at: http://infocenter.arm.com/help/topic/com.arm.doc.dui0497a/DUI0497A_cortex_m0_r0p0_generic_ug.pdf

9.4 CODE PROTECTION

Heimdall Slave implements a Code Protection Mechanism. Code Protection may be enabled and disabled with a specific sequence.

Please Contact indie semiconductor application team for details

9.5 WAKEUP

Heimdall Slave will wake up based on pin interrupt, an event from the PIR sensor or the Wakeup Timer.

During sleep Heimdall Slave uses the 10kHz RC Oscillator to optimize current consumption.

When the IC wakes up, the default RC Oscillator speed increases to ~250kHz to optimize startup time and current consumption.

During Wakeup the SW may instruct the system to switch to the Crystal Oscillator to complete RF transmissions or other tasks.

NOTE: The 10kHz RC Oscillator speed may be used upon wakeup. This is enabled in SW by setting bit TBD in Register TBD.

9.6 KEEP AWAKE FEATURE

In order to ensure the CortexM0 does not enter sleep, the TMS pin may be held to Vbat during PoR.

In order to implement this feature on Heimdall Slave, GPIO1 shall be left floating.

9.7 RF TRANSMITTER

HeimdallSlave implements a powerful RF transmitter operating in the ISM (**I**ndustrial, **S**cientific and **M**edical) band and capable of transmitting using either ASK/OOK, PSK or FSK modulations. Its main characteristics are:

- Precise fractional-N PLL referenced to a crystal oscillator, which drives into a modulation control circuit, and then into a high power output stage
- ASK/OOK, PSK or FSK modulation circuits
- High power output stage
- Integrated power control
- Autonomous state machine which controls transmit bursts

The following block diagram details the main structures used by the transmitter.

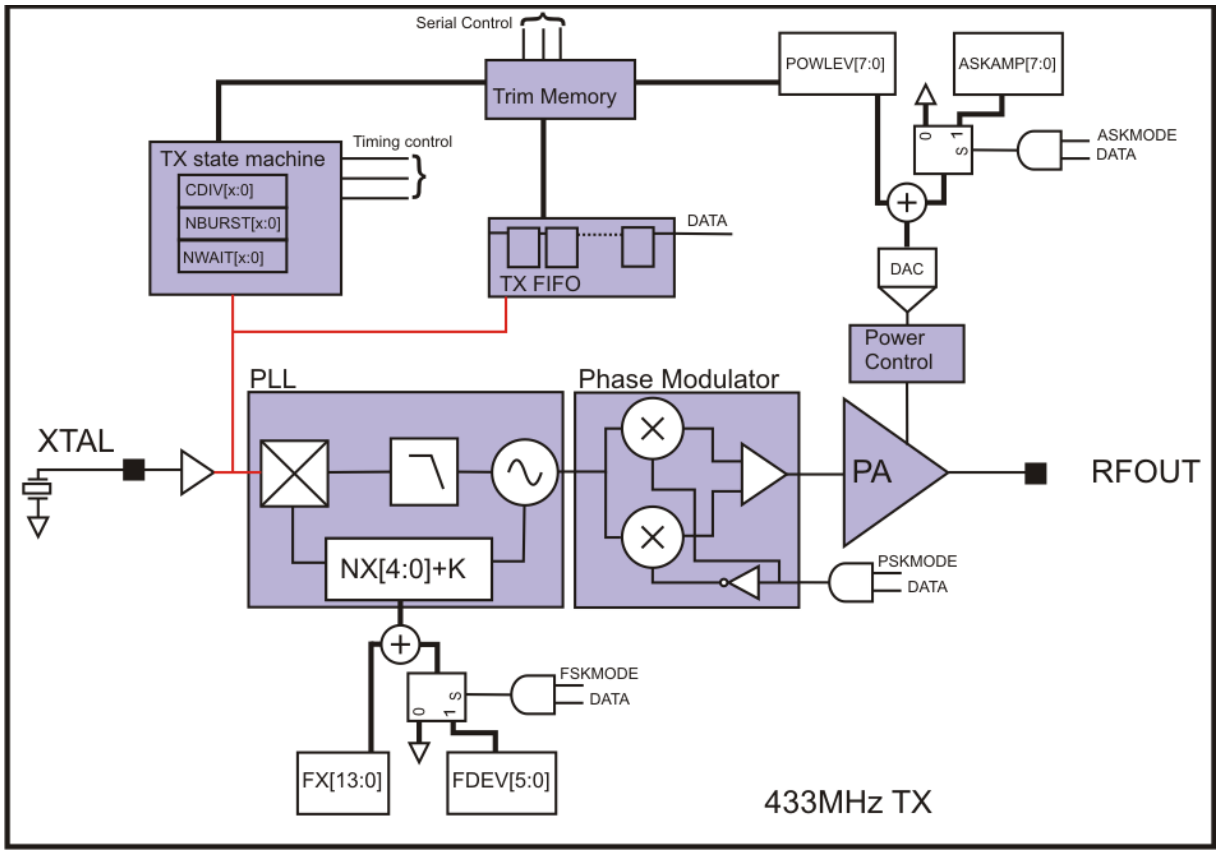


Figure 4: Transmitter Block Diagram

The output stage amplifies modulated data up to a maximum power level of +15dBm.The device has a single output pin, which requires external matching. The PA output power is regulated by an integrated power control stage. This power control circuit takes an internal measurement, which correlates with the output power of the high power output stage, and compares it to an internal reference, which is set by a digital control register.

The RF transmit frequency is generated via a fractional-N PLL. The frequency is generated by comparing a divided version of an RF VCO to a crystal reference frequency. The PLL may be operated in integer mode by setting bit F_EN to logic 0, or in fractional mode by setting bit F_EN to logic 1.
The frequency generated is calculated according to the following formula:-

In integer mode:

$$f_{VCO}=f_{XTAL}*(NX[4:0]+K), \text{ where } K=8$$

In fractional mode:

$$f_{VCO}=f_{XTAL}*(NX[4:0]+K+(NF[13:0]/2^{14})), \text{ where } K=9$$

Table 11 Transmitter Performance Specification					
name	**conditions**	**min**	**typ**	**max**	**unit**
Maximum Output Power	Vbat=2.2V			7.8	dBm
	Vbat=3.0V			14.1	dBm
	Vbat=3.2V			15.1	dBm
Output Power Range	Vbat=3.0V, T_A=27°C	-13.5		+14	dBm
Output Power Step	Valid from +11.5 to -9.5dBm output power		1.5		dB
Frequency Range	315 MHz, 350 MHz, 390 MHz, 433.92 MHz	315		450	MHz
Frequency deviation	FSK mode	1.83		115	kHz
Current Consumption	Pout=max (+13dBm)		18		mA
Sleep Current Consumption	All blocks disabled			100	nA
Data Rate	Output Power below 0dBm for ASK/OOK			50	kbps
Data Rate	Output Power higher than 0dBm for ASK/OOK or PSK mode			25	kbps
Output Noise	fc = 434.07 MHz, frequency offset=1.6 MHz compliant with EN 300 220-1 (2000.09)			-36	dBm
Harmonics, Conducted 2nd Harmonic 433 MHz 3rd Harmonic 433 MHz			 -45.1 -49.4		dBm
Frequency stability	Depends on crystal employed			13	ppm

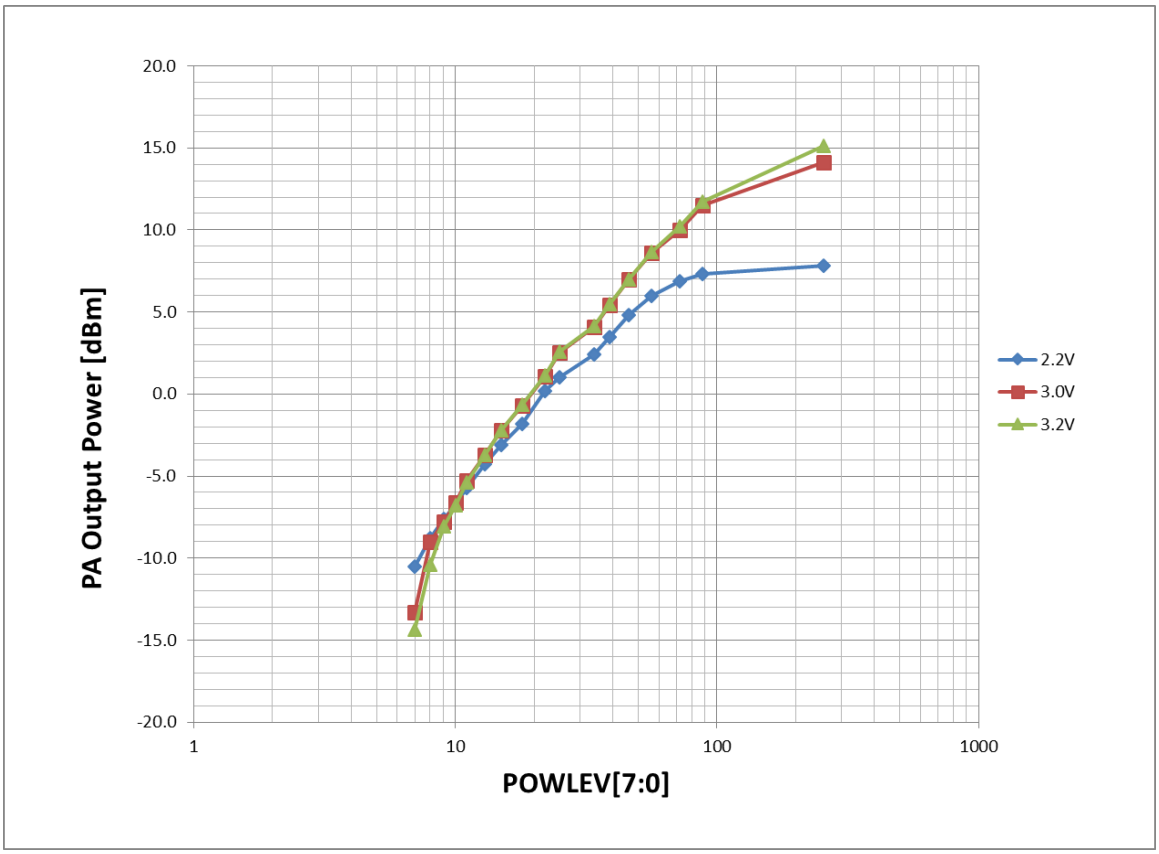


Figure 5: PA Output Power vs POWLEV[7:0]

Freq = 434.08 MHz

Temp = 27C, TX_DATA_BUF[15:0]

9.7.1 RF Transmitter Spurious – Matched

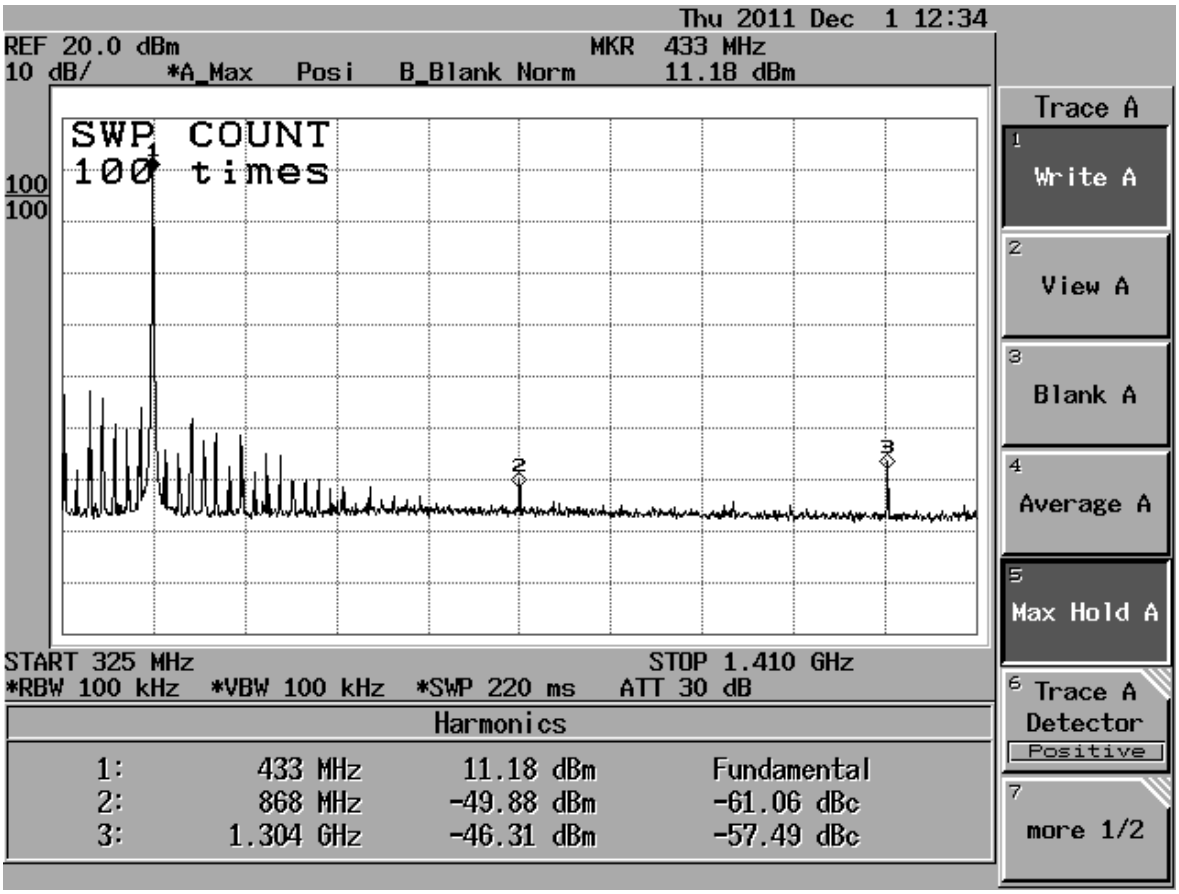


Figure 6: Vbat=3V, Temp=27C, fundamental=433.92MHz

TX_DATA_BUF[15:0] = all 1's

9.7.2 RF Transmitter Phase Noise

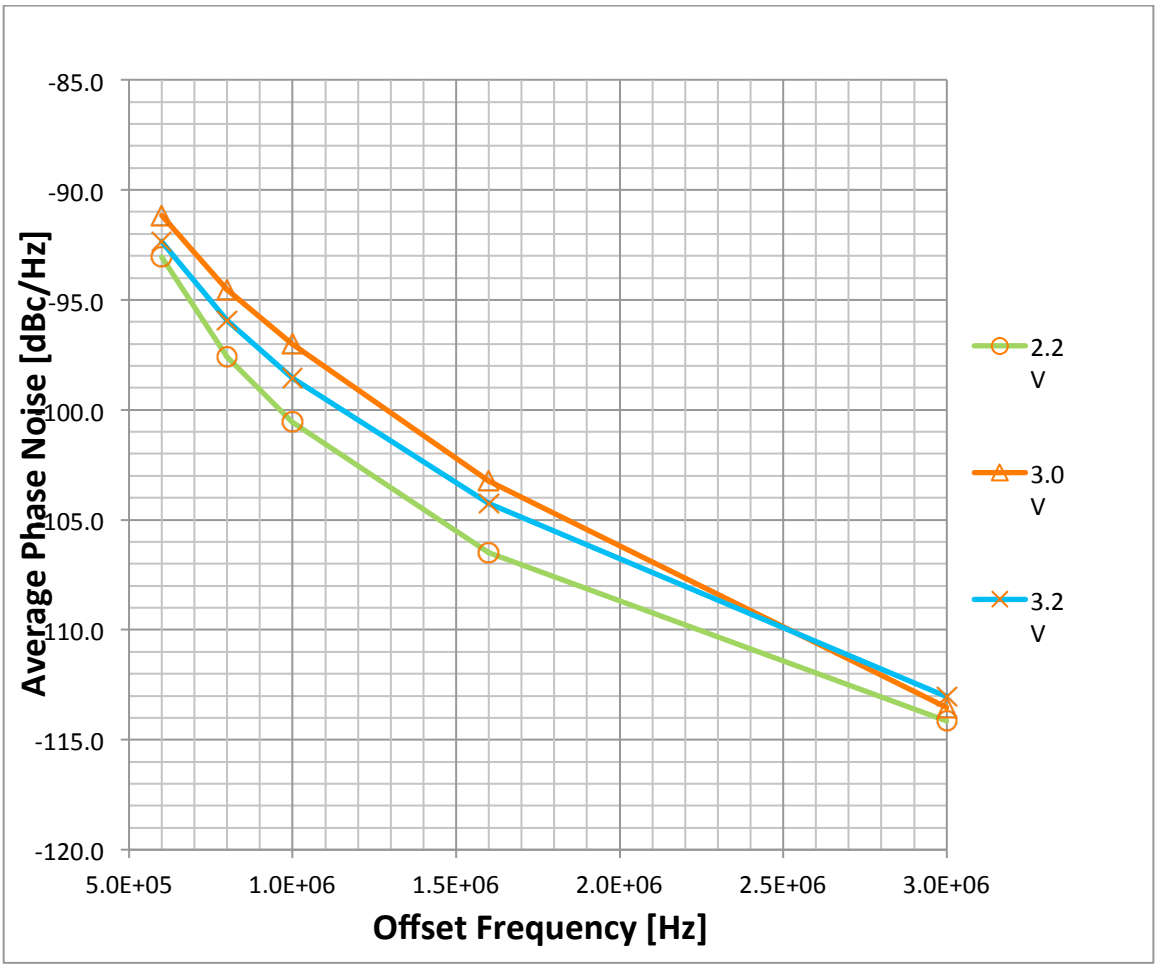


Figure 7: Average Phase Noise vs Offset Frequency

Span 5M, RBW 100 kHz, VBW 10 kHz, SWP 20ms

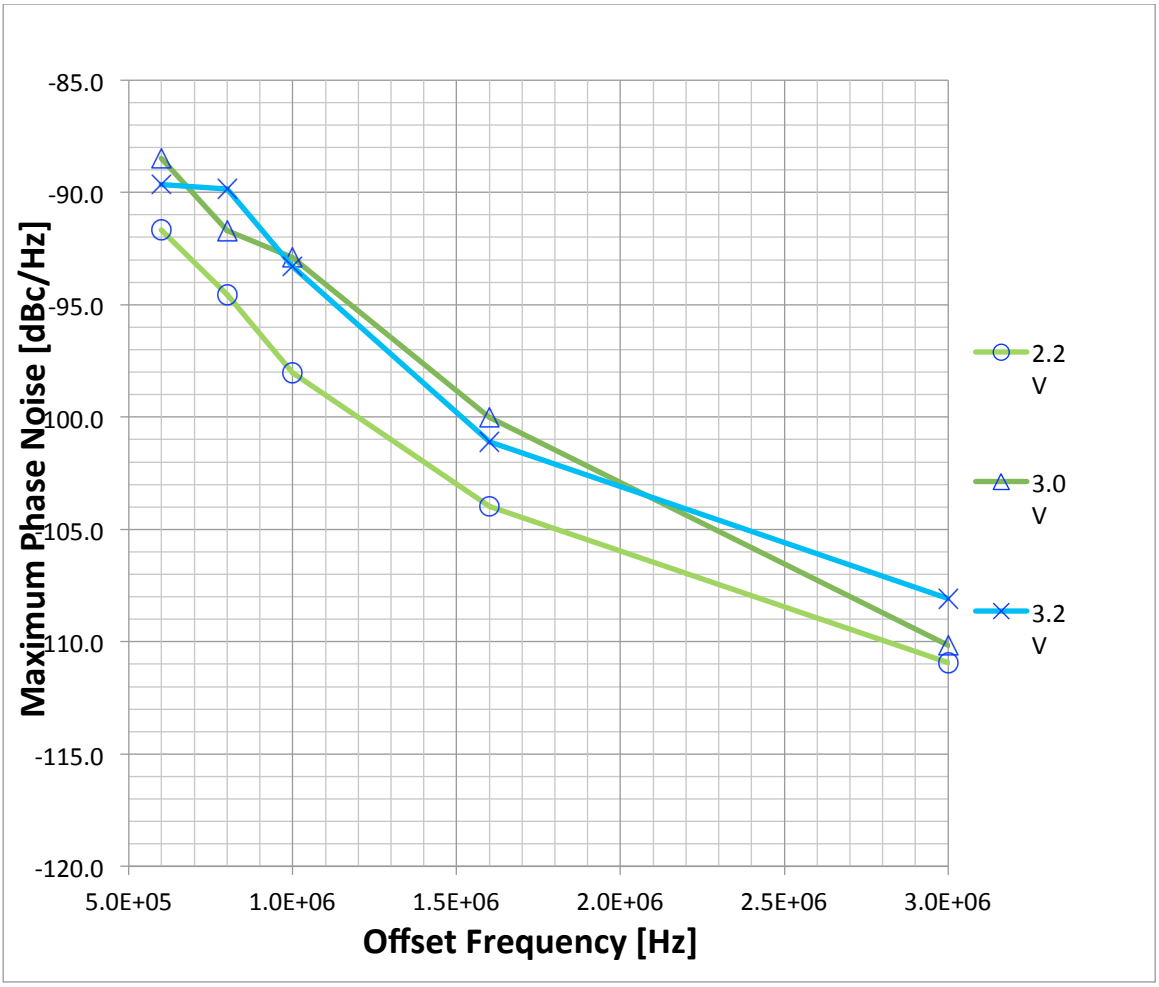


Figure 8: Maximum Phase Noise vs Offset Frequency,
Span 5M, RBW 100 kHz, VBW 10 kHz, SWP 20ms

Phase Noise [dBc/Hz]						
	Max. Hold			Average		
Offset Freq.	2.2V	3.0V	3.2V	2.2V	3.0V	3.2V
6.00E+05	-91.7	-88.5	-89.6	-93.0	-91.2	-92.4
8.00E+05	-94.5	-91.7	-89.9	-97.6	-94.5	-95.9
1.00E+06	-98.0	-92.9	-93.3	-100.6	-97.0	-98.6
1.60E+06	-104.0	-100.0	-101.1	-106.5	-103.3	-104.3
3.00E+06	-110.9	-110.2	-108.1	-114.2	-113.5	-113.1

Figure 9: Phase Noise dBc/Hz,
Span 5 MHz, RBW **10 kHz**, SWP 20 ms

9.7.3 Fractional PLL

The RF transmitter frequency is generated via a fractional-N PLL circuit. The frequency is generated by comparing a divided version of an RF VCO to a crystal reference frequency. The default reference frequency is 30 MHz.

The PLL may be operated in integer mode by clearing the FRACEN bit (**TXCTRL0** register) or in fractional mode by setting FRACEN bit (**TXCTRL0** register).

The frequency generated is calculated depending on the PLL setup of NX and NF bits (located in **PLLCTRL0**, **PLLCTRL1** and **PLLCTRL2** registers):

PLL in integer mode:

$$Fvco = Fxtal \times (NX[4...0] + 8)$$

PLL in fractional mode:

$$Fvco = Fxtal \times (NX[4...0] + 9 + \frac{NF[13...0]}{2^{14}})$$

In the ISM band operating a center frequency of 433.92 MHz in the band of 433.05–434.79 MHz, HeimdallSlave can generate nominal frequencies from 433.049927 MHz up to 434.789429 MHz in steps of ~1831Hz.

Code Example: Setting the RF Transmitter to transmit at 433.92 MHz in fractional mode from a 30 MHz clock:

```
test = RF_Set_Frequency( 30000000, 433920000, PLL_FRACTIONAL);
```

9.7.4 Transmission State Machine Operation

The transmitter implemented in HeimdallSlave operates as a FSM (**F**inite **S**tate **M**achine) implemented in hardware. This FSM provides a group of registers that allow for a very flexible and powerful yet simple transmission of data. The FSM provides automatic transmission of up to 16 bits without reloading transmission registers. The FSM may interact with the microcontroller via 2 interrupts.

- The interrupts are:
    - Single Byte Left: when the buffer has only a single byte left to transmit (so that the micro can choose to refill both bytes as they are double-buffered)
    - End of Burst: After sending all bits the FIFO is cleared and the transmitter waits NWAIT samples. At this moment an interrupt is also generated. The micro can refill the data buffer with samples and reprogram the clock division rate for the next burst if required or, if no new data is to be sent then it can disable the system.

The steps to properly set this FSM are as follows:

Select the modulation scheme: ASK/OOK, PSK or FSK using the MODUL[1…0] bits (TXCTRL0 register).

Set the PLL for the required frequency.

Select the bit rate using TXCDIV[10…0]. (TXCLKDIV and TXCTRL3)

Define the total number of bits per burst with NTXBITS[9…0] bits (NBURST and TXCTRL2).

Define the number of bits between bursts using NWAIT[5…0] (TXCTRL2).

Select the appropriate values for the different timing elements: TDET1, TDET2, TDET3, TDET4, TDET5 and TPATCH. (TXCTRL3, TXCTRL4, TXCTRL5)

If required enable the interrupts associated with the transmission (end of burst and end of first byte transmission).

Load the first two bytes of data to be transmitted in TXDATA[15…0] bits (TXDATL and TXDATH).

Set the STARTX bit (TXCTRL0 register) to initiate transmission.

9.7.4.1 Modulation Selection

Three different modulation schemes are available: ASK/OOK, FSK and PSK. In many instances ASK/OOK modulation is used in key fob applications while home security systems use PSK but the decision has to be made in the system level design by the user.

To select one of them simply load the MODUL[1…0] bits in the TXCTRL0 register :

Code example 1: (To set the PSK modulation)

```
*TXCTRL0 &= ~MODULASK;          //Clear previous modulation
*TXCRTL0 |= MODULPSK;           //Enable PSK modulation
```

Code example 2: (To set the ASK modulation now using access functions)

```
ret = RF_Set_Modulation( ASK );
```

9.7.4.2 Chip Rate Selection

To program the required bit rate the following equation must be used:

$$chiprate = \frac{1.25MHz}{\mathrm{TXCDIV}}$$

For instance, to obtain a chip rate of 19200bits/sec:

$$chiprate = \frac{1.25MHz}{\mathrm{TXCDIV}} = 19200 \Rightarrow \mathrm{TXCDIV} = \frac{1.25MHz}{19200} \cong 65$$

(This in fact will generate a chip rate of ~19231bits/sec, an error of 0.16%)

Code example: (From the calculation above)

```
      TXTIMING->CLKDIV.HOWRD = 0xF800; // Clear CLKDIV
      TXTIMING->CLKDIV.BYTE[0] = 65;    // Load the proper value in the lower 8
bits
```

NOTE:

Loading TXDIV with '0' will result in the maximum divider (1024), resulting in the minimum chip rate of ~1220bits/sec.

Baud rate will be fraction of above chip rate and depends on the format of coding used. For example if the coding used is Manchester, baud rate would be chip rate divided by two. If coding used is 1/3 2/3 coding, baud rate will be chip rate divided by three.

9.7.4.3 Burst Size Selection

The number of bits to be transmitted per burst is defined in NTXBITS[9…0].

The following code example selects a 700 bits/burst transmission: (700 = 0x2BC)

```
      *TXCTRL2 &= 0xFC;              //Clear the upper 2 bits of the burst size
      *TXCTRL2 |= 0x02;              //Load 0x02 into them
      *NBURST = 0xBC;                //Load 0xBC into lower 8 bits (700d = 0x2BC)
```

Example Code using access function:

```
      RF_Set_BurstSize( 700 );
```

9.7.4.4 Inter-Bursts Bit-Number Selection

The time waited between bursts is defined in number of bits at the current bit-rate. This time is defined as a function of NWAIT and the timing elements. The calculation of the total time between bursts is explained in the following chapter.

Code Example: To create a 7 bits NWAIT time:

```
      *TXCTRL2 &= ~0xFC;             //Clear the 5 bits of NWAIT
      *TXCTRL2 |= (0x07<<2);         //Load 7d (0x07) into them
```

9.7.4.5 Timing Elements Selection

The RF transmitter is made of several subsystems. In order to operate properly they have to be activated and the proper settling time defined for each one of them. The figure below shows the several timing elements:

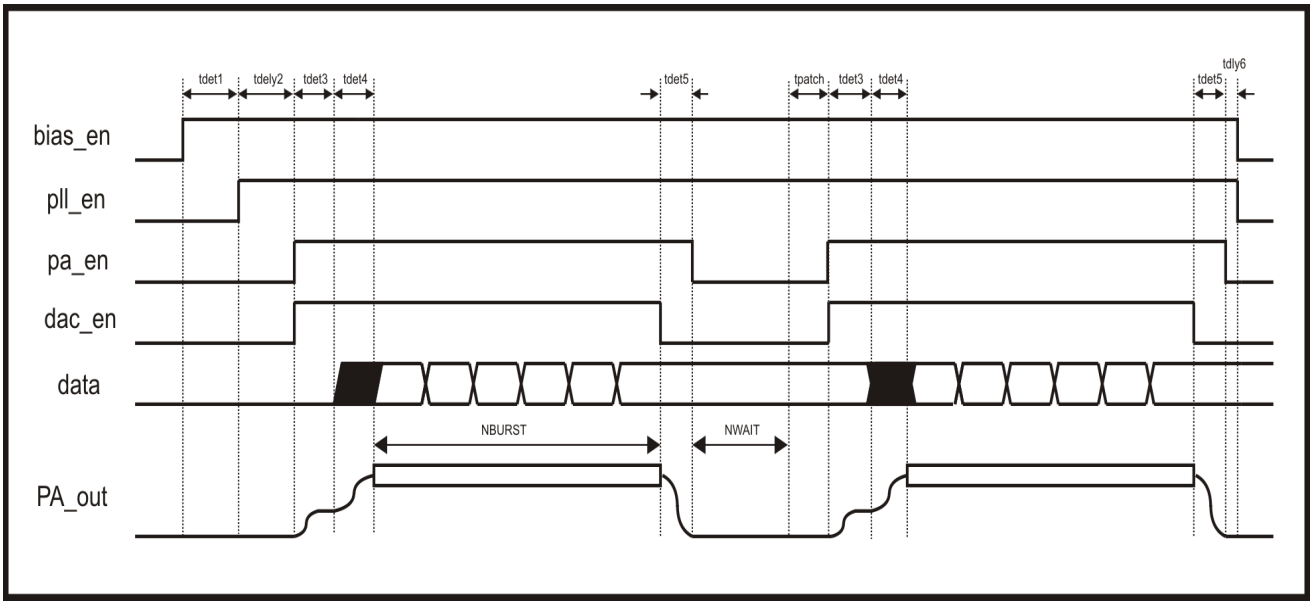


Figure 10: Transmitter Timing

Once the transmission is started several subsystems are automatically activated:

PA (**P**ower **A**mplifier)

PLL (**P**hase **L**ocked **L**oop)

DAC (**D**igital-**A**nalog **C**onverter)

The user can see in the figure above each one of these subsystems require different delay times in order to reach their proper operational condition. These delay times are defined by timing elements as follows:

Tdet1 – This time element allows for the PA bias to reach the proper levels. The minimum recommended time for this delay is 1usec, therefore selecting TDET1 = ‘b0 (1.6usec) is adequate.

Tdet2 – This time element allows for the PLL to be enabled and stabilize properly. The minimum time for it to do so is 1msec. and the recommended value for this field is TDET2 = ‘b010 (1.6msec).

Tdet3 – This time element defines the “Pedestal” period. This time is necessary to ensure the PA reached a midway power level prior to being enabled into full power. The recommended time for this parameter is ‘b101 (4usec).

Tdet4 and Tdet5* – These time elements are responsible for allowing for the minimum delay required to bring the PA to full power and minimum power. Both time elements should in principle be given the same value. The recommended value is ‘b10100 (32usec).

*TDET5 is not directly accessible in HeimdallSlave and it reflects the value of TDET4.

Tpatch – This time element is responsible for adjusting the wait time between bursts to create an integer multiple of bit times interval.

From Figure 10 (Transmitter Timing) we can see that the time interval between bursts is equal to:

$$T = t\det 5 + NWAIT + tpatch + t\det 3 + t\det 4 = NWAIT + tpatch + t\det 3 + 2 \times t\det 4$$

Assuming the recommended values for Tdet3 and 4 we have then:

$$T = NWAIT + tpatch + 68u\sec$$

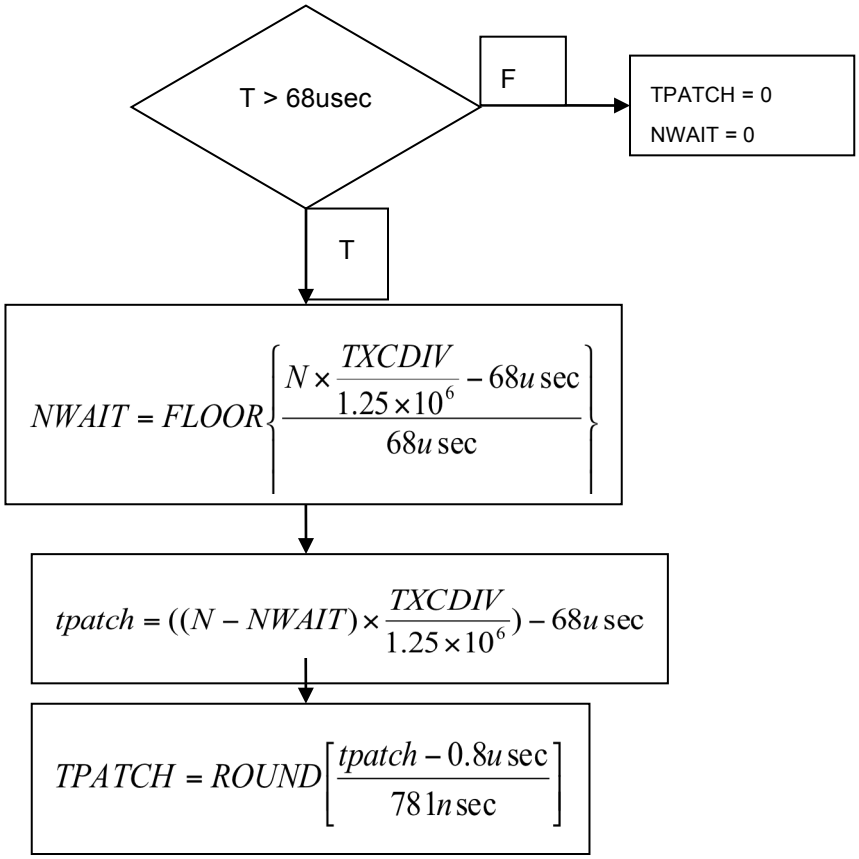
For a given number N of bit-times required between bursts the calculation becomes:

$$T = N \times \frac{TXCDIV}{1.25 \times 10^6} = NWAIT + tpatch + 68u\sec$$

And:

$$Tpatch = 781 \times 10^{-9} \times (TPATCH[10...0]) + 0.8 \times 10^{-6}$$

The following flowchart defines the procedure to adjust the parameters:



For instance, if the user wants a baud rate of 19200bits/sec and 7 bit-time of interval between bursts assuming the recommended values described above the calculation becomes:

$$T = (7) \times \frac{65}{1.25\times10^6} = 364u\,\text{sec}$$

T is larger then 68usec, therefore we can go ahead:

$$NWAIT = FLOOR\left\{\frac{\left[7 \times \frac{65}{1.25\times10^6} - 68u\,\text{sec}\right]}{68u\,\text{sec}}\right\} = 4$$

We have then 4 bit-time waits. Now calculating the tpatch time:

$$tpatch = \left[(7-4) \times \frac{65}{1.25\times10^6}\right] - 68u\,\text{sec} = 88u\,\text{sec}$$

$$TPATCH = ROUND\left[\frac{88u\,\text{sec} - 0.8u\,\text{sec}}{781n\,\text{sec}}\right] = 112$$

This gives us the following:

$$T = 68u\,\text{sec} + 4 \times \frac{65}{1.25 \times 10^6} + 781 \times 10^{-9} \times 112 + 0.8u\,\text{sec} = 364.3u\,\text{sec}$$

Code Example:

```
*TXCTRL3 &= 0x07;              //Clear upper 5 bits
*TXCTRL3 |= (0x10<<3);         //Set TPATCH = 112d = 0x70 (Low 5 bits)
*TXCTRL1 &= 0xC0;              //Clear TPATCH[10…5] bits
*TXCTRL1 |= 0x03;              //Load 0x03 (High 6 bits)
*TXCTRL4 &= 0xF0;              //Clear lower 4bits
*TXCTRL4 |= (0x02<<1)+0x00;    //Set TDET2='b010  and TDET1=0
*TXCTRL5 |= (0x05<<5)+0x14;    //Set TDET3='b101  and TDET4=0x14
```

9.7.4.6 TX Interrupts

The RF transmission module allows for two different interrupts: Byte Reload and Burst Transmission Done.

Code Example: The following code snippet enables the RF Transmission Interrupts (TX_Reload and TX_Done):

```
#include "heimdall_slave.h"

void main(void)
{
  //Code that execute other initializations…

  NVIC_EnableIRQ( TX_RELOAD_IRQn );
  NVIC_EnableIRQ( TX_DONE_IRQn   );

  //And more code…

}
```

The following code shows examples of ISR (**I**nterrupt **S**ervice **R**outines) associated with both interrupts:

```
void TX_Reload_Handler( void )
{
  TXDATABUF->HWORD = rf_data[rf_data_idx];   // Write data and increment index
  rf_data_idx = rf_data_idx + 1;
  rf_data_idx &= 0x0F;                      // 15 + 1 = 0;
}

void TX_Done_Handler( void )
{
  TXDATABUF->HWORD = rf_data[0];
  rf_data_idx = 1;
}
```

9.7.4.7 TRAMP Register

When the system is operating in PSK modulation there is a chance of a slight drop in the power during a transition between bits. To avoid this event a special field called TRAMP was implemented. This register defines a small delay time during which the PA feedback loop is opened (PSK mode only). This feature prevents the PA output power ramp up/down when the data bit changes. The recommended value for this register is ‘b1000 which represents a delay of approximately 0.25usec.

Code Example:

```
*TXCTRL4 &= 0x0F;              //Clear upper 4bits
*TXCTRL4 |= (0x08<<4);         //Set TRAMP='b1000
```

9.7.5 Transmitter Registers

The transmitter provides several registers that provide a flexible and simple to use interface:

Table 12 Transmitter Register Map

Address	Register Name	Field Name	Description
0x50000004	TXCTRL0	STARTX	Single bit which starts the transmission
		NXTBURST	Next Burst Flag
		MODUL	Modulation Selection
		FRACEN	PLL Fractional Enable bit
		[2…0]	Reserved
0x50000005	TXCTRL1	PEDES	Pedestal Time
		TPATCH	Delay Time between disabling the PLL and turning off the bias circuit
0x50000006	POWLEV	POWLEV	Power Level
0x50000007	POWAP	POWAP	Power level for ASK/OOK and PSK transmission of '1' (High Level)
0x50000008	TXDATL	TXDATA	Low Byte of Transmission Data
0x50000009	TXDATH	TXDATA	High Byte of Transmission Data
0x5000000A	NBURST	NTXBITS	Number of bits of transmission in the next burst low byte
0x5000000B	TXCTRL2	NWAIT	Number of bits between bursts
		NTXBITS	Number of bits of transmission in the next burst higher 2 bits
0x5000000C	TXCLKDIV	TXCDIV	Low byte of clock divider
0x5000000D	TXCTRL3	TPATCH	Lower 5 bits of the delay between disabling the PLL and turning off the bias
		TXCDIV	Upper three bits of the Transmission Clock Divider. (data rate is the 1.25Mbps/CDIV)
0x5000000E	TXCTRL4	TRAMP	Delay PA feedback loop opening period in PSK mode. It prevents the PA output power ramp up/down when the data bit changes.

Table 12 Transmitter Register Map			
		TDET2	Delay between enabling the PLL and enabling the power control DAC & power amplifier
		TDET1	Delay between enabling the bias and enabling the PLL
0x5000000F	TXCTRL5	TDET3	Delay between enabling the power control DAC and enabling the power amplifier
		TDET4	PA output power rising and falling time
0x50010004	PLLCTRL0	NF	Low byte of the Fractional Portion of Feedback Divider
0x50010005	PLLCTRL1	NX	Lower two bits of the Integer portion of feedback divider
		NF	Six higher bits of the Fractional Portion of Feedback Divider
0x50010006	PLLCTRL2	NX	Higher three bits of the Integer portion of feedback divider

A more detailed description of each register follows below:

Register 9 First Transmitter Control Register							
TXCTRL0		0x50000004			0x08		
R/W	R/W	R/W	R/W	R/W	Reserved	Reserved	Reserved
STARTX	NXTBURST	MODUL1	MODUL0	FRAC_EN	-	-	-
MSB							LSB

Bit7 **STARTX**: Start Transmission.

0 = Do not start transmit

1 = Start Transmitting

Bit6 **NXTBURST**: Set to indicate there is a next burst.

0 = No next burst

1 = There will be a next burst

Bit5-4 **MODUL[1:0]:** Modulation selected.
00 = Reserved

01 = PSK

10 = FSK

11 = ASK/OOK

Bit3 **FRAC_EN**: Fractional PLL enabled.

0 = Fractional PLL disabled

1 = Fractional PLL enabled

Bit3-0 **Reserved**, write 0b000

Register 10 Second Transmitter Control Register							
TXCTRL1		0x50000005			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
PEDES1	PEDES0	TPATCH10	TPATCH9	TPATCH8	TPATCH7	TPATCH6	TPATCH5
MSB							LSB

Bit7-6 **PEDES[1: 0]**: Pedestal Time:

00 = 12.8usec.

01 = 14.4usec

10 = 16usec.

11 = 17.6usec.

Bit5-0 **TPATCH[10:5]**: Delay between disabling the PLL and turning off the bias:

0x000 = 0.8usec

:

0x7FF = 1.6msec

Register 11 Power Level of FSK Transmission and Logic '0' power level in ASK/OOK and PSK transmission

POWLEV		0x50000006			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POWLEV7	POWLEV6	POWLEV5	POWLEV4	POWLEV3	POWLEV2	POWLEV1	POWLEV0
MSB							LSB

Bit7-0 **POWLEV[7:0]**: Power Level, defined as follows:

POWLEV[7:0] (HEX)	Pout (dBm)
06-00	<-11
0A-07	-9.5
0B-08	-8
10-09	-6.5
12-0A	-5
12-11	-3.5
17-12	-2
18-14	-0.5
1B-18	+1
1F-1B	+2.5
24-1F	+4
28-26	+5.5
36-31	+7
45-37	+8.5
55-39	+10
70-54	+11.5

Register 12 Power Level of ASK/OOK and PSK transmission in level '1'							
POWAP		0x50000007			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POWAP7	POWAP6	POWAP5	POWAP4	POWAP3	POWAP2	POWAP1	POWAP0
MSB							LSB
Bit7-0 **POWAP[7:0]**: Refer to **POWLEV** register.							

Register 13 TX data lower byte register							
TXDATL		0x50000008			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
TXDATA7	TXDATA6	TXDATA5	TXDATA4	TXDATA3	TXDATA2	TXDATA1	TXDATA0
MSB							LSB
Bit7-0 **TXDATA[7:0]**: Lower byte of data to be transmitted.							

Register 14 TX data upper byte register							
TXDATU		0x50000009			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
TXDATA15	TXDATA14	TXDATA13	TXDATA12	TXDATA11	TXDATA10	TXDATA9	TXDATA8
MSB							LSB
Bit7-0 **TXDATA[15:8]**: Upper byte of data to be transmitted.							

Register 15 Number of bits in the next burst							
TXDATU		0x5000000A			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
NTXBITS7	NTXBITS6	NTXBITS5	NTXBITS4	NTXBITS3	NTXBITS2	NTXBITS1	NTXBITS0
MSB							LSB
Bit7-0 **NTXBITS[7:0]**: Lower byte of the counter of number of bits to be transmitted in the next burst.							

Register 16 Third Control Register for RF Transmission							
TXCTRL2		0x5000000B			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
NWAIT5	NWAIT4	NWAIT3	NWAIT2	NWAIT1	NWAIT0	NTXBITS9	NTXBITS8
MSB							LSB
Bit7-2 **NWAIT[5:0]**: Number of bits (time) between bursts. Bit1-0 **NTXBITS[9:8]**: Upper two bits of the counter of number of bits to be transmitted in the next burst.							

Register 17 Transmission Clock Divider							
TXCLKDIV		0x5000000C			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
TXCDIV7	TXCDIV6	TXCDIV5	TXCDIV4	TXCDIV3	TXCDIV2	TXCDIV1	TXCDIV0
MSB							LSB
Bit7-0 **TXCDIV[7:0]**: Low byte of clock divider. (data rate is defined as 1.25Mbps/ TXCLKDIV)							

Register 18 Fourth Control Register for RF Transmission							
TXCTRL3		0x5000000D			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
TPATCH4	TPATCH3	TPATCH2	TPATCH1	TPATCH0	TXCDIV10	TXCDIV9	TXCDIV8
MSB							LSB

Bit7-3 **TPATCH[4:0]**: Lower 5 bits of the delay between disabling the PLL and turning off the bias.

0x000 = 0.8usec

:

0x7FF = 1.6msec

Bit2-0 **TXCDIV[10:8]**: Upper three bits of the Transmission Clock Divider. (data rate is 1.25Mbps/CDIV)

Example: If we want to transmit data at a rate of 19200bits/sec we should load **TXCDIV** with:

$$baud = \frac{1.25MHz}{\mathrm{TXCDIV}} = 19200 \Rightarrow \mathrm{TXCDIV} = \frac{1.25MHz}{19200} \cong 65$$

This in fact will generate a baud rate of ~19231bits/sec. (0.16% error)

NOTE: The equation for TPATCH is:

$$Tpatch = 78 \times 10^{-9} \times (TPATCH[10...0]) + 0.8 \times 10^{-6}$$

Register 19 Fifth Control Register for RF Transmission							
TXCTRL4		0x5000000E			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
TRAMP3	TRAMP2	TRAMP1	TRAMP0	TDET2_2	TDET2_1	TDET2_0	TDET1
MSB							LSB

Bit7-4 **TRAMP[3:0]**: Delay PA feedback loop opening period in PSK mode. It prevents the PA output power ramp up/down when the data bit changes. This delay is calculated as:

$$Delay = \frac{TRAMP}{SysClk}$$

This leads to the following values: (Assuming System Clock = 30 MHz)

0000 = 0usec

:

1000 = 0.27usec

:

1111 = 0.5usec

Bit3-1 **TDET2[2:0]**: Delay between enabling the PLL and enabling the power control DAC & power amplifier. This delay is calculated as:

$$Delay = \frac{TDET2}{1.25MHz} \times 1024$$

This leads to the following values:

000 = 0msec

:

100 = 3.28usec

:

111 = 5.73usec

Bit0 **TDET1**: Delay between enabling the bias and enabling the PLL:

0 = 1.6usec

1 = 8usec

Register 20 Sixth Control Register for RF Transmission							
TXCTRL5		0x5000000F			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
TDET3_2	TDET3_1	TDET3_0	TDET4_4	TDET4_3	TDET4_2	TDET4_1	TDET4_0
MSB							LSB

Bit7-5 **TDET3[2:0]**: Delay between enabling the power control DAC and enabling the power amplifier:

$$Delay = \frac{TDET3}{1.25MHz}$$

This leads to the following values:

000 = 0usec

:

100 = 3.2usec

:

111 = 5.6usec

Bit4-0 **TDET4[4:0]**: PA output power rising delay time:

$$RiseFall = \frac{TDET4}{1.25MHz} \times 2$$

Register 21 Modulation RC filter capacitor trim							
MDRCTRIM		0x50010003			0x44		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
MDRC7	MDRC6	MDRC5	MDRC4	MDRC3	MDRC2	MDRC1	MDRC0
MSB							LSB

Bit7-0 **MDRC[7:0]**: PSK and ASK/OOK RC filter capacitor trim bits.

Register 22 PLL Control Register 0							
PLLCTRL0		0x50010004			0x06		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
NF7	NF6	NF5	NF4	NF3	NF2	NF1	NF0
MSB							LSB
Bit7-0 **NF[7:0]**: Fractional Portion of Feedback Divider lower bits.							

Register 23 PLL Control Register 1							
PLLCTRL1		0x50010005			0x2D		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
NX1	NX0	NF13	NF12	NF11	NF10	NF9	NF8
MSB							LSB
Bit7-6 **NX[1:0]**: Integer portion of feedback divider low bits. Bit5-0 **NF[13:8]**: Fractional Portion of feedback divider high bits.							

Register 24 PLL Control Register 2							
PLLCTRL2		0x50010006			0xCB		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
LPFTRIM1	LPFTRIM0	CPTRIM2	CPTRIM1	CPTRIM0	NX4	NX3	NX2
MSB							LSB
Bit7-6 **LPFTRIM[1:0]**: Low pass filter trim bits. Bit5-3 **CPTRIM[2:0]**: Charge pump trim bits. Bit2-0 **NX[4:2]**: Integer portion of feedback divider high bits.							

Register 25 PLL Control Register 3							
PLLCTRL3		0x50010007			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DF5	DF4	DF3	DF2	DF1	DF0	LPFTRIM3	LPFTRIM2
MSB							LSB
Bit7-2 **DF[5:0]**: Frequency deviation control Bit1-0 **LPFTRIM[3:2]**: Low pass filter trim bits.							

9.7.6 Modulation

9.7.6.1 OOK Modulation

OOK is the simplest form of Amplitude Shift Keying (ASK) Modulation that represents digital data as the presence or absence of a carrier wave. The Heimdall Slave OOK is implemented by setting the Modulation as noted below:

Code example: (To set the ASK modulation now using access functions)

```
TXCTRL->TXCRTL0 |= TXCTRL0_MODUL_ASK;           // Enable ASK modulation;
```

Set the Power levels as follows:

- POWAP -> 0x39 shall set the peak power level for ASK/OOK ‘1’
- POWLEV -> 0x00 shall set the low power level for ASK/OOK ‘0’

9.7.6.2 OOK Modulation measurement data

The following measurements setup:

- Frequency = 434.08 MHz
- Pout = 10.12 dBm
- Vbat = 3.0V
- Temp = 27°C
- IVBAT = 18mA
- TX_DATA_BUF[15:0] = AAAA (1010 1010 1010 1010)

Results

- 20dB bandwidth = 400 kHz
- Rise Time = 5.6 us
- Fall Time = 13.8 us

Measurement data Below:

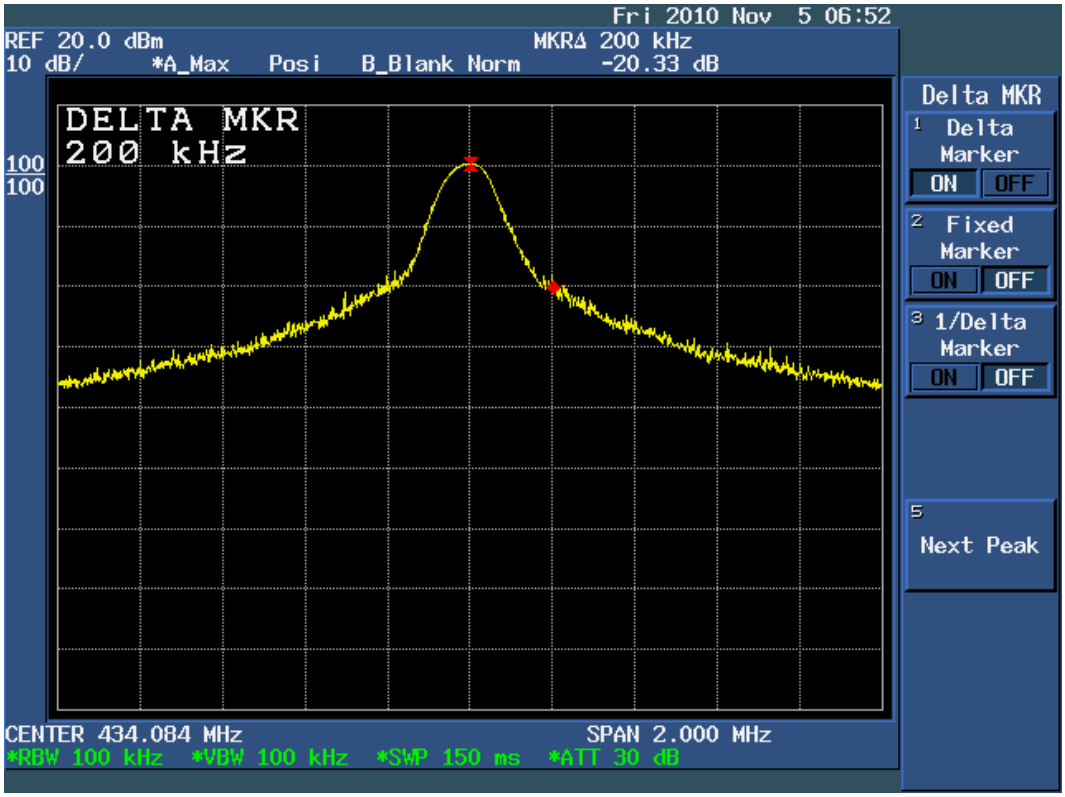


Figure 11: 20dB Bandwidth = 400 kHz, FCC limit is 1.08475 MHz

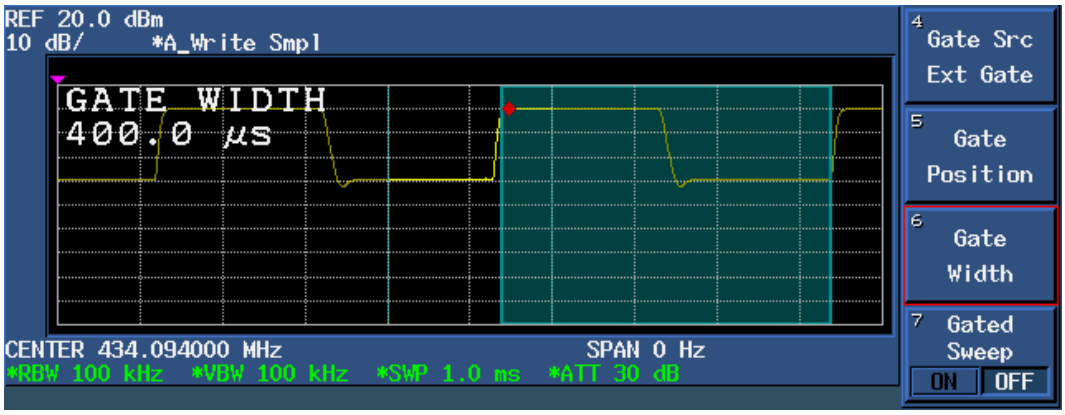


Figure 12: OOK Bit Rate ~5 kbps, deviation = 30dB

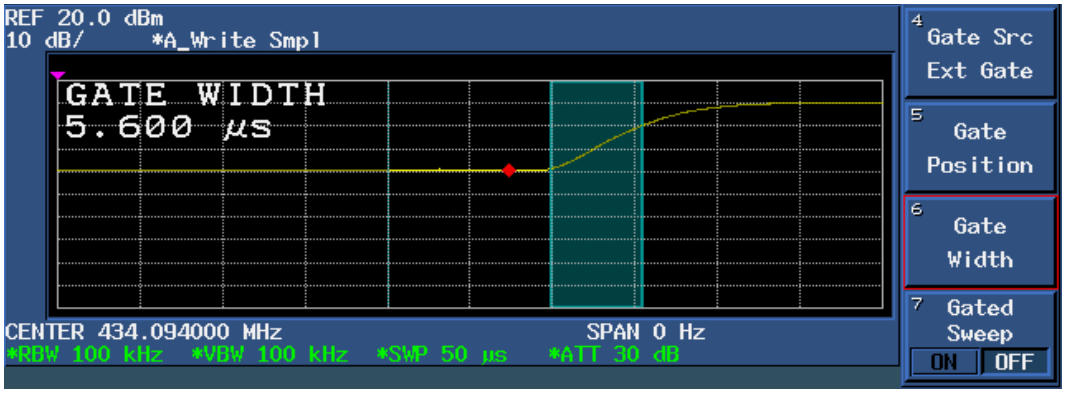


Figure 13: OOK Rise Time = 5.6 us

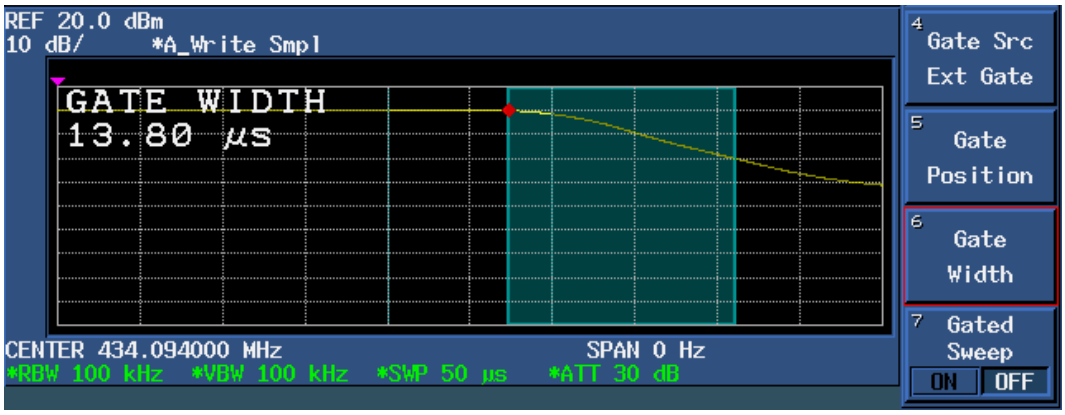


Figure 14: OOK fall time = 13.8 us

9.7.7 BPSK Modulation Measurement

Phase Shift Keying uses a finite number of phases to transmit data, encoding one unique bit pattern in each phase.

Figure 15: Demodulated BPSK from TX Data Stream

(2 byte) of 0101_1001_1010_0011 (59A3), Vbat = 3.0, Temp = 27C, Fout = 434.08 MHz

NOTE: 16 Bit Demodulated BPSK matches with the transmitted data at the rate of 25kbps

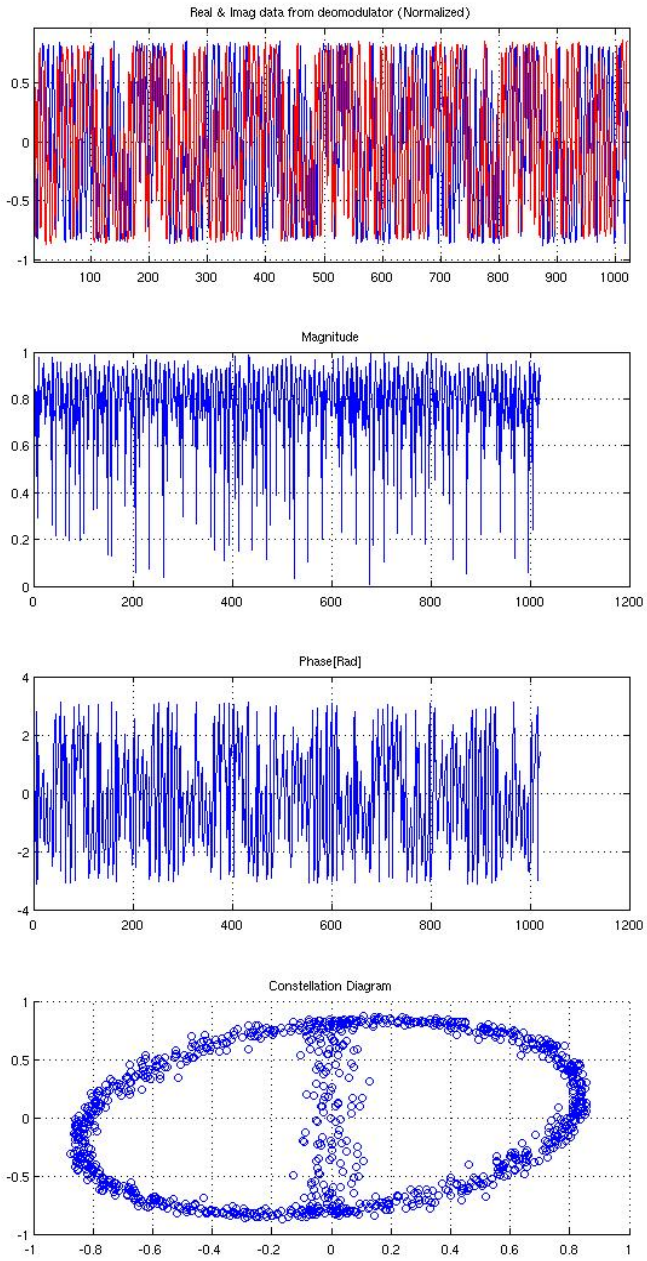


Figure 16: Real and Imaginary Data from Demodulator (Normalized)

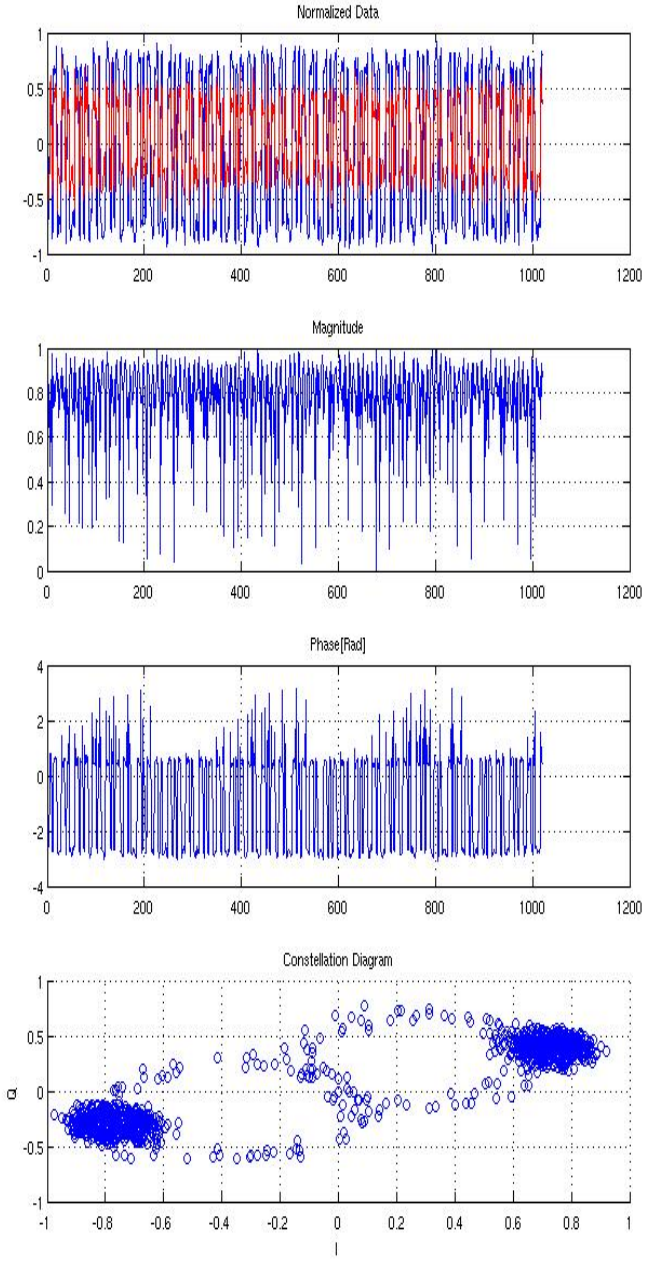


Figure 17: Real and Imaginary Data from Demodulator (Normalized), Compensated Frequency Offest

9.8 PIR SENSOR INTERFACE

HeimdallSlave implements an Interface to a PIR (**P**assive **I**nfra-**R**ed) sensor, which is used to detect human motion in applications such as home security systems and occupancy sensors for light switches.

It consists of two signal conditioning operational amplifiers, which are intended to be configured, as band pass filters with approximately 40dB of gain per stage between 0.1 and 10Hz. The gain and roll-off are set by external components.

The output of the filter stages is routed through a Programmable Passive Attenuator to the DC coupled window comparator. The window comparator determines if the wanted signal exceeds a preset threshold.

The output of the window comparator feeds a digital section where the pulse length is measured. If the pulse length exceeds the programmed value set in PIRC[9…0] (**PIRCNTR** and **PRCTRL**) then an event is signaled.

If an event is signaled a second digital block times a period through which new events are ignored. This second period is known as the Inhibit Length. The Inhibit Length may be programmed through INHIB[3…0] (**PIRCTRL**).

The figure below shows the block diagram for the peripheral:

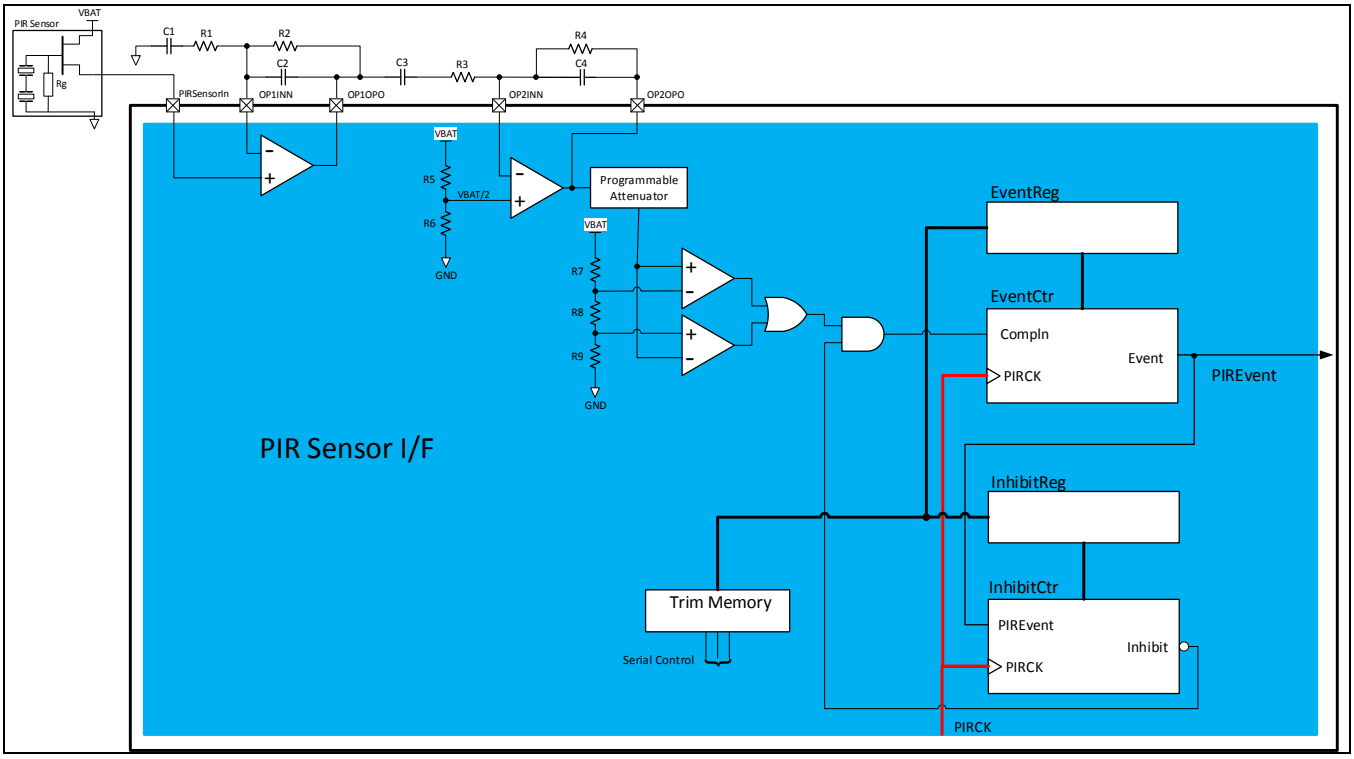
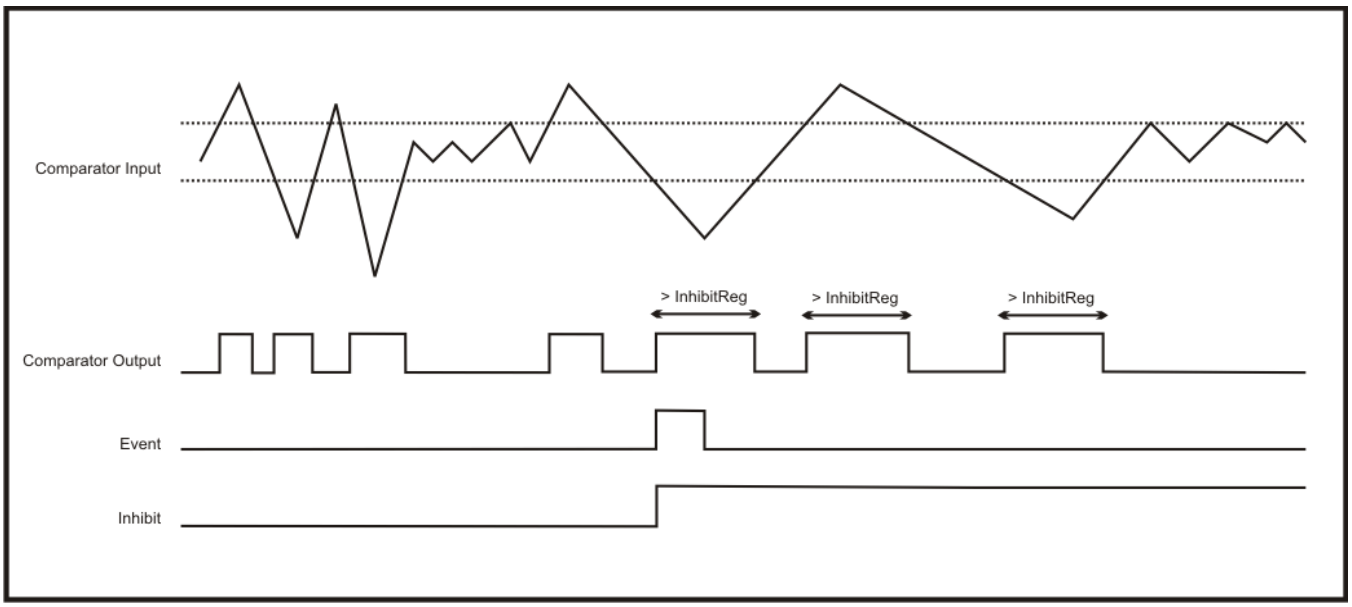


Figure 18: HeimdallSlave with External PIR and External Filter Components

The circuit seen in Figure below contains the components necessary to create the band pass filters used by the PIR Peripheral Interface. The recommended values are listed in the table below.

PIR Interface External Components			
R1	6.4 kΩ	C1	22 µF
R2	1 MΩ	C2	22 nF
R3	10 kΩ	C3	9.4 µF
R4	1 MΩ	C4	22 nF

This peripheral is enabled after power on reset, but may be disabled by software. The timing diagram in **Error! Reference source not found.** demonstrates the overall behavior of the interface:



PIR Circuit Timing

The individual sub-blocks of the PIR Sensor Interface are described in detail below:

9.8.1 First Stage Operational Amplifier

The first stage operational amplifier provides a fully exposed OpAmp with the inverting, non-inverting and output nodes available at the defined pins.

This stage is intended to be configured as a band pass filter with around 40dB of in-band gain.

The input is P-type only, and as such has a limited common-mode input range. Using P-type only devices improves noise and offset compared to the second stage. The output of this stage will support rail-to-rail operation.

The following table provides the basic operating conditions and performance figures of the first stage:

Table 13 Specification First Stage Op Amp

Name	Conditions	min	typ	max	unit
Current consumption	Enabled		500	800	nA
Gain bandwidth product		8	15		kHz
Slew Rate				3	V/ms
Input referred offset			±3	±7	mV
Input referred noise	At 0.1Hz		500	800	nV/√Hz
Input referred noise	At 10Hz			100	nV/√Hz
CMRR		60	75	80	dB
Input voltage range		-0.5		1.8	V
Open Loop DC gain			100		dB
Output voltage Range		10		VBAT-10	mV
Linear Output voltage Range		100		VBAT-100	mV

9.8.2 Second Stage Operational Amplifier

The second stage operational amplifier provides access to the inverting and output nodes of the OpAmp. The non-inverting input is internally biased to VBAT/2.

This stage is intended to be configured as a band pass filter with around 40dB of in-band gain.

The input is both N-type and P-type, which means it will operate rail-to-rail. The output stage will also support rail to rail operation.

The following table provides the basic operating conditions and performance figures of the second stage:

Table 14 Specification Second Stage Op Amp

Name	Conditions	min	typ	max	unit
Current consumption	Enabled		700	900	nA
Gain bandwidth product			15		kHz
Slew Rate				3	V/ms
Input referred offset			±5	±15	mV
Input referred noise	At 0.1Hz		800	TBD	nV/√Hz
Input referred noise	At 10Hz			TBD	nV/√Hz
CMRR		60	75	80	dB
Input voltage range		-0.5		VBAT	V
Open Loop DC gain			100		dB
Output voltage Range		10		VBAT-10	mV
Linear Output voltage Range		100		VBAT-100	mV

9.8.3 Programmable Tree Attenuator

The Programmable Tree Attenuator is a configurable passive attenuator used for low current signal path conditioning.

The attenuator is programmed by a 4 bit control that steps the attenuation from 0dB to 26dB in 2dB steps.

The DC level is maintained by constructing a resistor divider structure which consists of a programmable resistor structure which is placed in series with the input signal, and two resistors of equal value, which are connected in series between Vdd and ground, with the common connection being connected to the programmable resistor.

The programmable resistor is generated via several equal value segments, which are tapped via minimum size complementary pass gates. The only quiescent current is consumed in the fixed resistor string between Vdd and ground. This resistor string is equipped with an enable switch to eliminate current consumption when inactive.

Table 15 Specification Programmable Tree Attenuator

Name	Conditions	min	typ	max	Unit
Current consumption	VDD=3.0V, TA=27°C, B=0H		500	650	nA
Maximum Frequency	VDD=3.0V, TA=27°C, B=0H			1	kHz
Attenuation Step Accuracy	VDD=2.5-3.2V, TA=-10-+60°C	1.8	2	2.2	dB
Step switching time	VDD=3.0V, TA=27°C			1	ms

Programmable Tree Attenuator Settings

B[3:0]	Attenuation
0	0dB
1	2dB
2	4dB
3	6dB
4	8dB
5	10dB
6	12dB
7	14dB
8	16dB
9	18dB
A	20dB
B	22dB
C	24dB
D	26dB
E	26dB
F	26dB

9.8.4 Window Comparator

The window comparator consists of two dynamic comparators, which are clocked with a divided version of the internal oscillator clock (10 kHz) defined in PIRDIV[1…0] (PIRCTRL register). As the comparators are dynamic, they consume negligible current.

The threshold for the comparators is set at Vdd/3 and 2Vdd/3 via a resistor string. If the wanted signal exceeds the higher threshold or is lower than the lower threshold then the comparator will output logic high.

Table 16 Specification Window Comparator

Name	Min.	Typ.	Max.	Unit
Low Threshold	VBAT/3-5%	VBAT/3	VBAT/3+5%	V
High Threshold	2*VBAT/3-5%	2*VBAT/3	2*VBAT/3+5%	V
Current Consumption (enable)			100	nA

9.8.5 Event and Inhibit Counters

The Event Counter counts as long as one of the comparator outputs is logic high. If the pulse length from the comparator output is longer than a pre-programmed value, the event flag is set. If the event flag is set, the Inhibit counter begins to count for a pre-programmed period, during which all subsequent event alarms are ignored.

The inhibit counter may be disabled via software, in which case the system must decide whether to ignore subsequent alarms.

9.8.6 PIR Interface Timing Calculations

The following equations define the relation between the event time, inhibit time and the INHB[3…0], PIRDIV[1…0] and PIRC[9…0] bits (**PIRCNRT** and **PIRCTRL0** registers):

$$Tevent = \frac{PIRC \times PIRDIV}{Frc}$$

$$Tinhibit = \frac{INHB \times PIRDIV}{Frc} \times 2048$$

Example: If an event time of ~1 second and an inhibit time of ~5seconds is required:

$$PIRC = \frac{1 \times 10kHz}{24} \cong 417 \text{ (Tevent = 1.04sec.)}$$

$$INHB = \frac{5 \times 10kHz}{2048 \times 24} \cong 1 \text{ (Tinhibit = 5.12sec.)}$$

Example Code: Selecting the threshold and enabling the PIR Interface:

```
PIR_Threshold(417);                 //PIR Threshold value
PIR_Config(PIREN, 1, PIRDIV24);     //PIR enabled, INHB = 1, 10 kHz/24 clock
```

The following table presents the values of Inhibit time as a function of the INHB bits and the PIRDIV bits: (RC clock at 10 kHz)

Inhibit Time (sec)				
INHB	**PIRDIV = 8**	**PIRDIV = 16**	**PIRDIV = 24**	**PIRDIV = 32**
0	0	0	0	0
1	1.8432	3.4816	5.12	6.7584
2	3.6864	6.9632	10.24	13.5168
3	5.5296	10.4448	15.36	20.2752
4	7.3728	13.9264	20.48	27.0336
5	9.216	17.408	25.6	33.792
6	11.0592	20.8896	30.72	40.5504
7	12.9024	24.3712	35.84	47.3088
8	14.7456	27.8528	40.96	54.0672
9	16.5888	31.3344	46.08	60.8256
10	18.432	34.816	51.2	67.584
11	20.2752	38.2976	56.32	74.3424
12	22.1184	41.7792	61.44	81.1008
13	23.9616	45.2608	66.56	87.8592
14	25.8048	48.7424	71.68	94.6176
15	27.648	52.224	76.8	101.376

9.8.7 PIR Interface Registers

The following registers control the PIR Interface behavior:

Register 26 PIR counter register							
PIRCNTR		0x50000000			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
PIRC7	PIRC6	PIRC5	PIRC4	PIRC3	PIRC2	PIRC1	PIRC0
MSB							LSB
Bit7-0 **PIRC[7:0]**: PIR event threshold counter low 8 bits.							

Register 27 PIR control register							
PIRCTRL		0x50000001			0x88		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
INHB3	INHB2	INHB1	INHB0	PIRDIV1	PIRDIV0	PIRC9	PIRC8
MSB							LSB

Bit7-4 **INHB[3:0]**: Inhibit bits.

Bit3-2 **PIRDIV[1:0]**: PIR Divider bits:

00 = 10 kHz/8

01 = 10 kHz/16

10 = 10 kHz/24

11 = 10 kHz/32

Bit1-0 **PIRC[9:8]**: PIR counter upper 2 bits.

Bits 7-4 of Register 28 (**GPIORE1**) controls the Programmable Attenuator for the PIR Interface.

Register 28 GPIO Read Enable Register 1							
GPIORE1		0x50010013			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
PROGATT3	PROGATT2	PROGATT1	PROGATT0	RE11	RE10	RE9	RE8
MSB							LSB

Bit7-4 **PROGATT[3:0]**: PIR ProgTreeAtten, when not in use write 0000b

B[3:0]	Attenuation
0	**0dB**
1	**2dB**
2	**4dB**
3	**6dB**
4	**8dB**
5	**10dB**
6	**12dB**
7	**14dB**
8	**16dB**
9	**18dB**
A	**20dB**
B	**22dB**
C	**24dB**
D	**26dB**
E	**26dB**
F	**26dB**

Bit3-0 **RE[11:8]**: GPIO read enable bit.

0 = Read Disabled

1 = Read Enabled

9.9 ADC

HeimdallSlave includes an analog to digital Converter (ADC). The ADC has 8-bit resolution with single ended input. The main features are described below:

- 8-bit resolution
- Single ended input
- Up to 80 ksps
- Configurable reference (VREF = VREFHI – VREFLO)
    - Either based on the bandgap voltage (VBG) or regulated supply voltage (VDD)
    - Reference may be scaled in order to provide more resolution around a smaller input voltage range
- Maximum ADC input range is from 0V to its supply voltage
- Using VBG as the reference, Supply Voltage can be measured in calibration mode via ADCCAL = 1
- Temperature Sensor implemented using PTAT input to ADC via ADCMUX = 1101
- 10 channels (GPIOs) plus PTAT reference in order that temperature may be measured

ADC Description

HeimdallSlave's ADC uses the standard charge redistribution technique, with a single-ended input and internally generated positive and negative reference voltages. The ADC accommodates 11 analog input channels. The user can select which input channels to be sampled by setting the ADCCHANNEL register. The ADC has its own internally generated reference voltages (VREFHI and VREFLO). The performance table is shown below.

Table 17 ADC Performance Specification, Recommended Operating Conditions

unless otherwise specified

Parameter	Conditions	min	typ	max	unit
Conversion speed				80	ksps
Clock Frequency				1	MHz
Input voltage range		0		VDD	V
Resolution				8	bits
INL				1	LSB
DNL				1	LSB

9.9.1 ADC Usage Description

There are several steps required for the user to use the ADC. The general sequence is described below:

- Step 1: Select the input channel to be measured
- Step 2: Configure ADC settings.
    a. Set ADC clock frequency.
    b. Configure references by programming the ADCREFHI, ADCREFLO, ADCPGN, and ADCREFS bits.
- Step 3: Start the ADC conversion.
- Step 4: Check the ADC status bit and read the data.

The following section will describe each configuration steps in detail.

9.9.1.1 Input Channel Selection

All GPIOs (GPIO0-2 & GPIO5-11) and the PTAT reference are available as an inputs to the ADC. The user can control which input is connected for the conversion by programming the control bits, ADCMUX, in **ADCMUX** register.
Code Example: Selecting PA0 for ADC input channel

```
ADC_Select_Channel( ADC_GPIO0 );
```

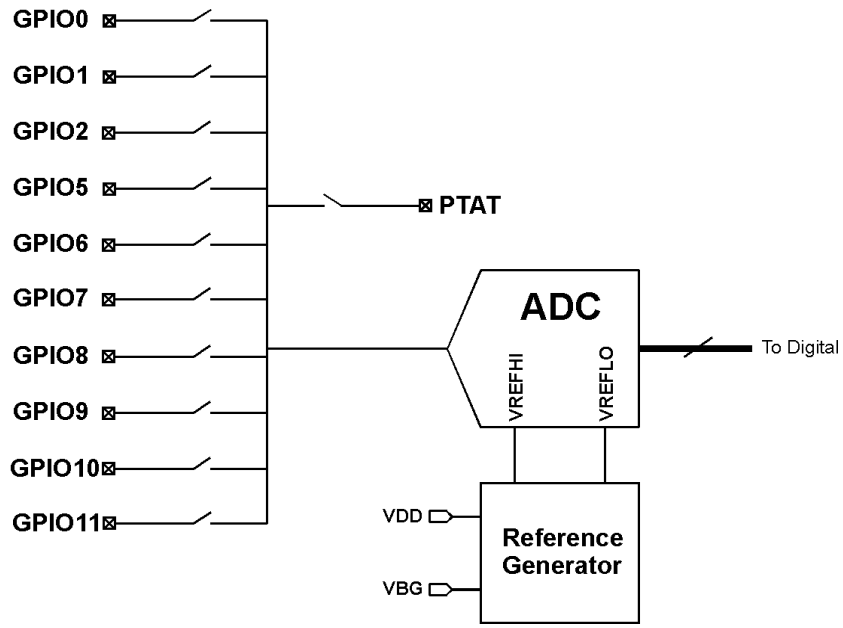


Figure 19: ADC Block Diagram

9.9.1.2 ADC Clock and Sampling Period

The conversion algorithm has a basic period of 9 cycles (one for sampling, one for each bit). There is a two-cycle latency from the last bit measurement until the data becomes available to the user. Additionally, there is a single idle cycle to allow biasing before any conversion is initiated. Thus a single conversion will take 13 cycles.

The converter will use a single clock cycle to sample the input into an input capacitor. When the channel is selected, the source must drive the sample/hold capacitor through the source resistance of the signal to be measured. The sampling time varies with this source resistance. The input to ADC must have sufficiently low driving impedance and settling time to settle the input to within 1 LSB of the data conversion during the input sampling stage. An equivalent circuit and related equations are depicted in **Error! Reference source not found.**.

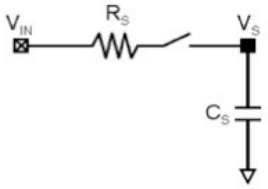
The ADC clock frequency can be programmed through **ADCCTRL0** register. As an example, if ADC clock is derived from 30 MHz RC oscillator divided by 32, the input has 1.07μs to settle. Since the C_s = 10pF in this device, the maximum source resistance to guarantee 8-bit performance can be calculated as below:

$$R_s = \frac{1.07\mu s}{10pF \times 5.5} = 19.4k\Omega$$

If the source impedance is larger, the user can increase the sampling time by either reducing the ADC clock frequency or increasing the sampling cycles which is controlled through **ADCCTRL2** register.

Code Example: Configure ADC clock to 30 MHz crystal oscillator frequency divided by 32.

```
ADC_Clock_Divider ( ADCDIV32 );
```



$$V_S = V_{IN}(1 - e^{-\frac{t}{\tau}}) \quad , where\ \tau = R_S C_S$$

V_S *is within 1 LSB of* V_{IN} *after* $t = 5.5\tau$

Figure 20: ADC input settling time

9.9.1.3 Configuration of Reference Voltage for the ADC

The ADC can generate its reference voltages (VREFHI and VREFLO) from two different sources, the regulated supply voltage (VDD) or the bandgap voltage (VBG). The ADCREFS bit in **ADCCTRL1** register selects the source. Once the reference source is selected, the reference voltages can be programmed through ADCREFHI, ADCREFLO, and ADCPGN bits according to Figure 22. It should be noted that when VBG is used as the reference source, care must be taken so that the internal voltages do not saturate.

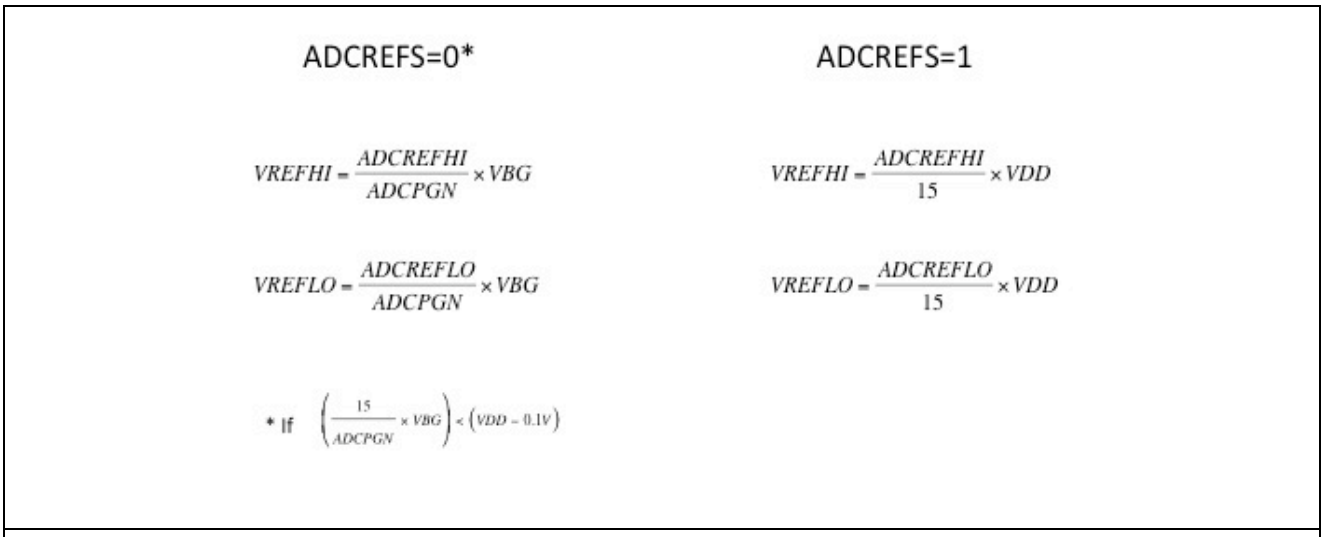


Figure 21: ADC Reference Voltage

Once the reference voltages are established, the ADC conversion equation for input voltage (VIN) can be defined as:

$$ADCDT = floor\left(255 \times \frac{(VIN - VREFLO)}{(VREFHI - VREFLO)}\right)$$

Here are few examples:

- Example 1: In a system operating with VDD=3V, there is a signal that moves between 0V and 2.94V. In this case it would be recommended to select VDD as the reference source and ADCREFH=15, ADCREFL=0, and ADCPGN=15. This selection would allow for the maximum range of measurements (0V to VDD).
  - `ADC_Reference_Config( ADC1, 15, 0, 15, ADCREFVDD);`
- Example 2: In a system operating with VDD=3V and VBG= 1.21V, there is a signal that moves between 0V and 2.5V. In this case VBG can be selected as the reference source with ADCREFH=13, ADCREFL=0 and ADCPGN=7. This configuration would allow the signal range from 0V to 2.6V:
  - `ADC_Reference_Config( 13, 0, 7, ADCREFBG);`

- Example 3: In a system operating with VDD=3V and VBG=1.21V, there is a signal that moves between 1.71V and 2.2V. In this case selecting VBG as the reference source and select ADCREFH=15, ADCREFL=11, and ADCPGN=8 we can achieve higher resolution:
    - `ADC_Reference_Config( 15, 11, 8, ADCREFBG);`

The resolution in Example 3 can be calculated as follow:

$$RESOLUTION = \left(\frac{VREFHI - VREFLO}{255}\right) = \left(\frac{\left(\frac{15}{8}\times 1.21\right) - \left(\frac{11}{8}\times 1.21\right)}{255}\right) = \frac{2.27 - 1.66}{255} = 2.38mV$$

Note that in this particular case we have the 8-bit ADC effectively generating a digital value with the precision of a 10-bit ADC operating from 0V to VDD.
It is clear from the examples how flexible the ADC can be in a range of applications. The user can devise several schemes to cleverly measure the range of signal of interest and then narrow the reference values to get the optimum resolution if the conversion time is within range.

9.9.1.4 ADC Start and Status

The ADC is enabled by the ADCEN bit in register **ADCCTRL0**. The ADCCONT bit in register **ADCCTRL0** starts the conversion process. Once completed the value of the conversion is loaded into the **ADCRES** register and a new conversion starts right after that. If the ADCCONT bit is reset the conversion currently being executed is completed, the result loaded into **ADCRES** and after that the ADC stops converting.

Code Example: Enable the converter and start conversion

```
ADC_Control(ADCEN, ADC_START);      //ADC enabled and start
ADC_Control(ADCEN, ADC_STOP);       //ADC enabled and stopped. Current
                                    //conversion is completed
```

9.9.2 ADC Registers

The following registers define the behavior of the ADC:

Table 18 : ADC Control Register Map				
Address	Register Name	Description	Reset Value	Reference
0x50000012	ADCRES			
0x50000013	ADCCTRL0			
0x50000014	ADCMUX			
0x50010008	ADCVREFG	ADC Reference Voltage Control		
0x50010009	ADCCTRL1	ADC Control Register 1		
0x5001000A	ADCCTRL2	ADC Control Register 2, LED		

Register 29 ADC result register							
ADCRES		0x50000012			0xXX		
R	R	R	R	R	R	R	R
ADCRES7	ADCRES6	ADCRES5	ADCRES4	ADCRES3	ADCRES2	ADCRES1	ADCRES0
MSB							LSB
Bit3-2 **ADCRES[7:0]** ADC data							

Register 30 ADC Control Register 0							
ADCCTRL0		0x50000013			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
TMX1	TMX0	QPCLK1	QPCLK0	ADCCLK1	ADCCLK0	ADCCONT	ADCEN
MSB							LSB

Bit7-6 **TMX[1:0]:** Test mux control bits (Controls GPIO0 output)

00 = normal

01 = micro clock

10 = PIR event indicator

11 = Wakeup indicator

Bit5-4 **QPCLK[1:0]:** Charge pump clock divider bits

00 = (System Clock)/8

01 = (System Clock)/16

10 = (System Clock)/32

11 = (System Clock)/64

Bit3-2 **ADCCLK[1:0]:** ADC clock divider bits

00 = (System Clock)/8

01 = (System Clock)/16

10 = (System Clock)/32

11 = (System Clock)/64

Bit1 **ADCCCONT**: Continuous conversion mode.

0 = Conversion stops after completion of the current one

1 = Conversion is executed continuously

Bit0 **ADCEN**: Enables the ADC peripheral. This bit is to be used in conjunction with the ADCCONT bit to control the start and stop of the conversion process.

Register 31 ADC Mux Control Register							
ADCMUX		0x50000014			0x00		
Reserved	Reserved	Reserved	Reserved	R/W	R/W	R/W	R/W
-	-	-	-	ADCMUX3	ADCMUX2	ADCMUX1	ADCMUX0
MSB							LSB

Bit3-0 **ADCMUX[3:0]:** ADC MUX

0000 = GPIO0
0001 = GPIO1
0010 = GPIO2
0011 = Reserved
0100 = Reserved
0101 = GPIO5
0110 = GPIO6
0111 = GPIO7
1000 = GPIO8
1001 = GPIO9
1010 = GPIO10
1011 = GPIO11
1101 = Temperature Sensor

NOTE: GPIO3 and GPIO4 are **not** implemented

Register 32 ADC Reference Voltage Control							
ADCVREFG		0x50010008			0x0F		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADCREFLG3	ADCREFLG2	ADCREFLG1	ADCREFLG0	ADCREFHG3	ADCREFHG2	ADCREFHG1	ADCREFHG0
MSB							LSB

Bit7-4 **ADCREFLG[3:0]**: Low reference voltage gain.

Bit3-0 **ADCREFHG[3:0]**: High reference voltage gain.

Register 33 ADC Control Register 1							
ADCCTRL1		0x50010009			0x2F		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADCCTRL1_7	ADCREFS	ADCCTRL1_5	ADCCTRL1_4	ADCPGN3	ADCPGN2	ADCPGN1	ADCPGN0
MSB							LSB

Bit7 **ADCCTRL1_7**: This bit is reserved and must be always written '0'.

Bit6 **ADCREFS**: Positive reference source:

0 = Band Gap

1 = Vdd

Bit5 **ADCCTRL0_5**: This bit is reserved and must be always written '1'.

Bit4 **ADCCTRL0_4**: This bit is reserved and must be always written '0'.

Bit3-0 **ADCPGN[3:0]:** ADC Gain

Register 34 ADC Control Register 2							
ADCCTRL2		0x5001000A			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADCXTRA0	ADCXTRA1	ADCXTRA2	ADCCAL	ADCCTRL2	BLUEWR	REDWR	ADCCONTS
MSB							LSB

Bit7-5 **ADCXTRA[0:2]**: Number of extra sampling clock cycles to be added to the normal one ADC clock used to sample the input signal.

Bit4 **ADCCAL**: ADC calibration bit

0 = normal

1 = Bandgap as an input to ADC

Bit3 **ADCCTRL2**: Reserved

Bit2 **BLUEWR**: Blue LED range

0 = 2 – 30mA in 2mA step

1 = 0.2 – 3mA in 0.2mA step

Bit1 **REDWR**: Red LED Range

0 = 2 – 30mA in 2mA step

1 = 0.2 – 3mA in 0.2mA step

Bit0 **ADCCONTS:** ADC continuous sampling control. While asserted the sampling process continues.

9.10 LOW BATTERY DETECTION

Low Battery Detection is accomplished by comparing the Battery Voltage (VBAT) with the Band Gap Voltage (VBG). To accomplish this the ADC must be set in calibration mode via setting bit ADCCAL = 1.
This function allows the calculation of the Battery Voltage (VBAT) as referenced to the Band Gap Voltage (VBG).

9.11 RESET

9.11.1 Brown out Reset

The Heimdall Slave IC has an integrated Brown Out Reset (BOR) circuit that may be enabled or disabled by SW (Register TBD). The value of the BoR is set to 2.2V

9.12 TEMPERATURE SENSOR

Temperature sensor is implemented using the ADC with the VBG as a reference and the PTAT input. To accomplish this the register ADCMUX[3:0] bits must be set to a value of 1101 to select the Temperature Sensor input. The PTAT may be characterized during production

The plot below shows the ADC Output Voltage vs Temperature.

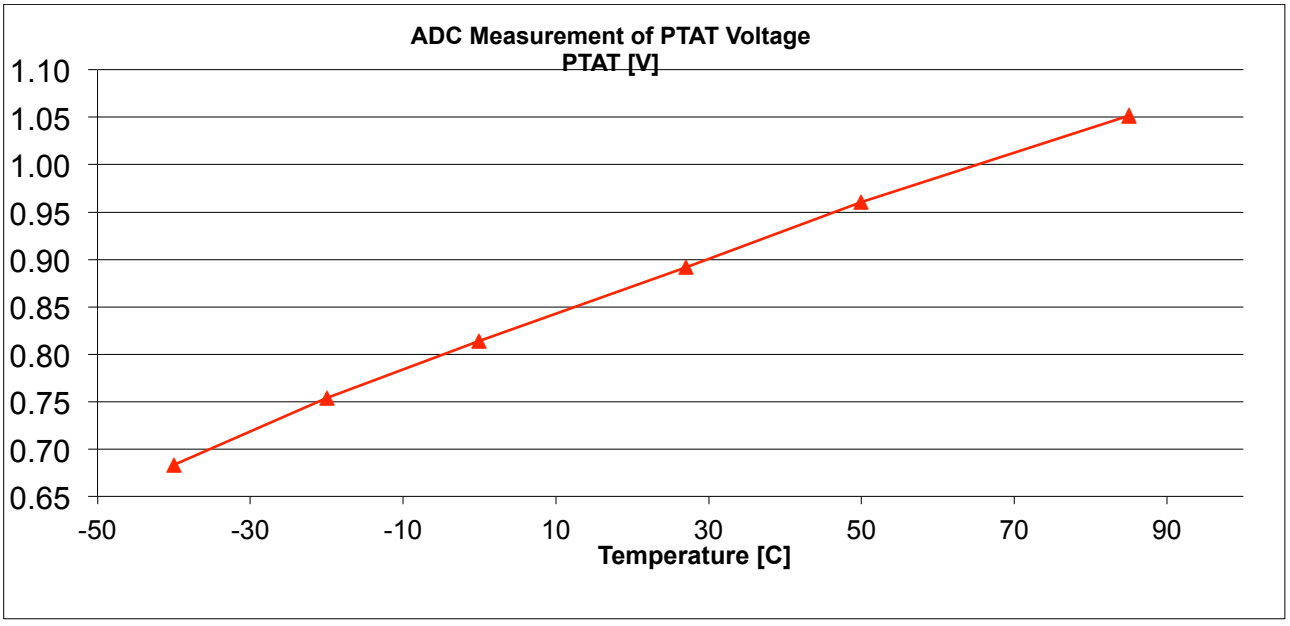


Figure 23: ADC Temperature vs PTAT Voltage

9.13 CLOCK SOURCES

HeimdallSlave provides two clock sources:

- Internal RC oscillator running at 10kHz or ~250KHz
  - 10 kHz Oscillator
    - Selectable auto calibration referenced to the crystal oscillator
  - ~250kHz Oscillator
- External Crystal driven by an internal oscillator circuit

HeimdallSlave C4 (late variant) starts from Power-On reset using the internal RC oscillator. From this point on the user may select to enable the crystal oscillator and switch to it if a higher execution speed is necessary to the application.

The user may also improve the precision of the 10 kHz internal oscillator by enabling an auto-calibration process.

9.13.1 Clock Sources Characteristics

The following table defines the main characteristics of the clock sources:

Table 19 Clock Performance Specification

name	conditions	min	typ	max	unit
Crystal Oscillator frequency			30		MHz
Frequency stability	Using appropriate crystal			50	ppm
Current Consumption Crystal				1	mA
Sleep Current Consumption Crystal	Crystal oscillator disabled			1	nA
RC Oscillator frequency		4	10	21	kHz
RC Oscillator accuracy	Post-calibration to 10 kHz, T_A=27°C			5	%
RC Oscillator current consumption	Enabled		500	TBD	nA

9.13.2 Clock Sources Usage Description

Upon Reset or Power-On Reset the system starts using the internal RC 10 kHz oscillator.
Depending on the application requirements the designer can:

- Enable or disable the internal RC oscillator
- Enable or disable the external crystal
- Select the system clock source: RC or Crystal
- Select to automatically enable the external crystal upon returning from sleep mode

Calibrate the internal RC oscillator in order to increase its precision.

NOTE: The PIR circuit uses the 10kHz clock as reference and therefore it must be enabled when this peripheral is in use. Similarly the RF transmitter peripheral requires the Crystal oscillator in order to operate.
The application code can directly access the registers (defined in Heimdall_Slave.H) and/or use the provided access functions.

Code Example1a: Enable oscillators, automatic crystal enable, select crystal as source, and perform an automatic calibration of the RC (10 kHz) oscillator:

```
      * UCNCTRL |= 0x1C;            // Enable both crystal and RC oscillators,
                                    as // well as auto. Crystal enable
      for ( i = 0; I < 20000; i++); // Delay to stabilize Crystal osc.
      *CLKCTRL |= 0x40;             // Select crystal as clock source, divided
by 1
      * RCOCTRL = 0x80;             // Execute automatic calib. of RC osc.
```

Code Example1b: Same tasks using the access functions:

```
     CLK_Control( RCEN, AUTOXTEN, XTEN );
     for ( i = 0; I < 20000; i++);
     CLK_Config( XTAL_CLK, CLKDIV1);
     CLK_Rc_AutoCal( STCAL, AUTOCAL, 0);
```

9.13.3 Clock Related Registers

The following registers are used to control the behavior of the clock sources:

Register 35 Processor Control Register							
UCCTRL		0x50000015			0x31		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
SWPWRDN	PIREN	VSRAMEN	RCOSCEN	AUTOXTAL	SWXTALEN	SOFTCLR	WKTEN
MSB							LSB

Bit7 **SWPWRDN**: Software forced power down

0 = Normal Operation

1 = Power down

Bit6 **PIREN**: PIR Enable Bit

0 = PIR Disabled

1 = PIR Enabled

Bit5 **VSRAMEN**: Enable the auxiliary regulator to keep SRAM values.

0 = Disabled

1 = Enabled

Bit4 **RCOSCEN**: RC oscillator enabled.

0 = RC oscillator disabled

1 = RC oscillator enabled

Bit3 **AUTOXTAL**: Automatic crystal enable

0 = Automatic crystal enable is disabled

1 = Automatic crystal enable is enabled

Bit2 **SWXTALEN**: Software Crystal enabled

0 = Crystal disabled

1 = Crystal enabled

Bit1 **SOFTCLR**: Software Clear

0 = Normal

1 = Set registers and states machines to default

Bit0 **WKTEN**: Wake-up timer enable

0 = Disabled

1 = Enabled

Register 36 Clock Control Register							
UCCTRL		0x50000002			0x01		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
-	SRC	CLKDIV1	CLKDIV0	TRIMDIV3	TRIMDIV2	TRIMDIV1	TRIMDIV0
MSB							LSB

Bit6 **SRC**: Clock Source:

0 = RC Clock

1 = Crystal

Bit5-4 **CLKDIV[1:0]**: Clock Divider

00 = Divide by 1

01 = Divide by 2

10 = Reserved

11 = Divide by 8

Bit3-0 **TRIMDIV[3:0]**: Reserved, always write 0001b.

Register 37 RC Oscillator Calibration Register							
RCCAL		0x50000010			0x3F		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
STARTCAL	FREQSEL	CALMODE	CALVAL4	CALVAL3	CALVAL2	CALVAL1	CALVAL0
MSB							LSB

Bit7 **STARTCAL**: Start Calibration of the Internal RC Oscillator:

0 = Do not start calibration

1 = Start calibration

Bit6 **FREQSEL**: Inform the peripheral of the Crystal Frequency:

0 = 30 MHz

1 = 20 MHz

Bit5 **CALMODE**: Calibration Mode:

0 = Automatic Mode

1 = Manual Mode

Bit4-0 **CALVAL[4:0]**: Calibration value. If CALMODE = 1 then the user must provide the value. If CALMODE = 0 then the value will be automatically loaded by the peripheral during the automatic calibration process. The general behavior is the following:

CALVAL[4:0] = 11111b → Lowest frequency calibration

:

CALVAL[4:0] = 10000b → Nominal value to achieve 10 kHz

:

CALVAL[4:0] = 00000b → Highest frequency calibration

Example Code:

```
CLK_Rc_AutoCal( STCAL, AUTOCAL,0);  //Execute auto calibration
```

9.14 GPIO PINS

HeimdallSlave provides ten (10) GPIOs, many of which share functionality with the PIR and LED functions.

Each GPIO may be used as a digital output, digital input, or analog input to the ADC. Additionally, several pins have additional functions. The pin direction, write or read, is set by a register for each port. There is a read enable and a write enable register for each port, and there are two registers per port, which control the activity of a pull up or pull down function, to ensure the pin is at a known state in input mode, even if the driving source is floating. Additionally, each pin can wakeup the system on a state change if enabled through registers.

GPIO5-9 also operate as the PIR circuitry. More details may be found in Section **Error! Reference source not found.**.

GPIO10 and GPIO11 also act as an LED driver, which features a programmable current sink to set LED brightness. More details may be found in section 9.15.

The following table defines the main characteristics of the GPIO pins:

Table 20 GPIO Main Characteristics

Name	Conditions	min	typ	max	unit
V_{IL}				0.3Vdd	V
V_{IH}		0.7Vdd			V
I_{OL}			10		mA
I_{OH}			10		mA
Pull-Down			100		kΩ
Pull-Up			100		kΩ

9.14.1 GPIO Usage Description

HeimdallSlave provides a total of 10 I/O pins. All I/O pins can be connected to the ADC converter. In addition to this capability several I/O pins can provide connections for other peripherals as follows:

Table 21 GPIO Alternative Functions

GPIO Pin	Alternative Connection
GPIO5	OP2OPO – Second Operational Amplifier Output (PIR)
GPIO6	OP2INN – Second Operational Amplifier Negative Input (PIR)
GPIO7	OP1OPO – First Operational Amplifier Output (PIR)
GPIO8	OP1INN – First Operational Amplifier Negative Input (PIR)
GPIO9	PIRSensorin – PIR Sensor Input (PIR)
GPIO10	RED LED Output
GPIO11	BLUE LED Output

Any I/O pins not used for its alternative function can be used as a general purpose I/0. The following parameters must be selected for the I/O pins:

- Output Selection (OE) – Upon Reset all I/O pins start in tri-state (High Impedance, Input). In order to use a pin as output this functionality has to be enabled.
- Pull-Down Selection (PD) – When used as inputs the I/O pins may have their Pull-Down resistors enabled.
  Pull-Up Selection (PUB) – When used as inputs the I/O pins may have their Pull-Up resistors enabled.
  Wake-Up on Pin Change Selection – Defines which pins will wake-up the processor
- Idle State Selection – The state against which the values of the I/O pins will be compared to wake-up the part
- Read Enable Selection (RE) – Defines the pins whose inputs will be readable

Code Example1: Selecting GPIO0 and 1 as outputs and GPIO2 as readable input with pull-up enabled and wake-up on change with idle state = 1, meaning wake-up when '0' is detected in the pin.

```
GPIO_Config( 0x0003, 0, 0x0004, 0x0004, 0x0004, 0x0004);
//Parameters order: outputs, pulldown, pullup, wakeup, idle_state, read_enable
```

Code Example 2: Writing '1' to GPIO0, writing '0' to GPIO1 and reading GPIO2 inside an 'if' statement.

```
GPIO_Write(0,1);  //Writing '1' to GPIO0
GPIO_Write(1,1);  // Writing '1' to GPIO1
If (GPIO_Read(2))
{
        //Act accordingly
}
```

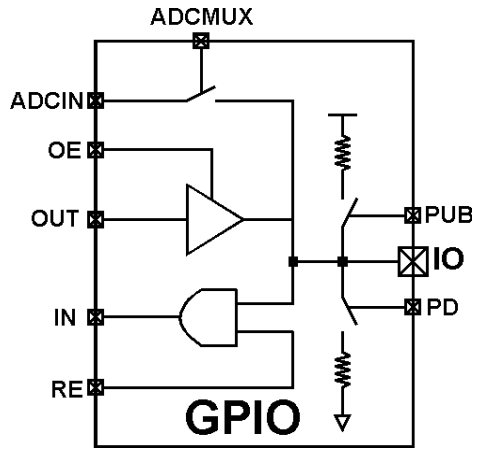


Figure 24 : GPIO Pin

9.14.2 GPIO Registers

The following registers control the behavior of the GPIO pins:

Register 38 GPIO Output Enable Register							
GPIOOE		0x50000016			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
GPIOOE7	GPIOOE6	GPIOOE5	GPIOOE4	GPIOOE3	GPIOOE2	GPIOOE1	GPIOOE0
MSB							LSB
Bit7-0 **GPIOOE[7:0]**: Output enable control. 0 = Pin output is disabled 1 = Pin output is enabled							

Register 39 GPIO Control Register							
GPIOCTRL0		0x50000017			0x00		
Reserved	R/W	R/W	R/W	R/W	R/W	R/W	R/W
-	LEDCPEN	RLEDEN	BLEDEN	GPIOOE11	GPIOOE10	GPIOOE9	GPIOOE8
MSB							LSB

Bit6 **LEDCPEN**: LED's Charge Pump enabled bit, necessary for the operation of the Blue LED output if the Vdd voltage is lower than 3.0V.

0 = Charge Pump disabled

1 = Charge Pump enabled

Bit5 **RLEDEN**: Red LED enable bit.

0 = LED disabled

1 = LED enabled

Bit4 **BLEDEN**: Blue LED enable bit.

0 = LED disabled

1 = LED enabled

Bit3-0 **GPIOOE[11:8]**: Output enable control.

0 = Pin output is disabled

1 = Pin output is enabled

Example Code: The following code fragment enables GPIOs 0 and 2 as outputs, enables the Charge Pump used to power the Blue LED and enables both Blue and Red LEDs:

```
*GPIOOE = 0x05;
*GPIOCTRL0 = 0x70;
```

Register 40 GPIO 0-7 Register							
GPIOLOW		0x50000018			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
MSB							LSB

Bit7-0 **GPIO[7:0]**: General Purpose IO data. When the pin is configured as output its values drive the physical pin. When the pin is configured as Input it contains the logical state of the pin.

Both the **GPIOHIGH** and the **GPIOLOW** registers are controlled by the GPIO 16-bit structure defined in heimdall_slave.H header file: (To be included by the user in its application code)

Register 41 GPIO 11-8 Register							
GPIOHIGH		0x50000019			0x00		
Reserved	Reserved	Reserved	Reserved	R/W	R/W	R/W	R/W
-	-	-	-	GPIO11	GPIO10	GPIO9	GPIO8
MSB							LSB

Bit7-0 **GPIO[11:8]**: General Purpose IO data. When the pin is configured as output its values drive the physical pin. When the pin is configured as Input it contains the logical state of the pin.

Both the **GPIOHIGH** and the **GPIOLOW** registers are controlled by the GPIO 16-bit structure defined in heimdall_slave.H header file: (To be included by the user in its application code)

Register 42 GPIO Control Register 1							
GPIOCTRL1		0x5000001A			0xFF		
W	W	W	W	W	W	W	W
IODEF7	IODEF6	IODEF5	IODEF4	IODEF3	IODEF2	IODEF1	IODEF0
MSB							LSB

Bit7-0 **IODEF[7:0]**: GPIO pins default state, used to detect pin change.

Register 43 GPIO Control Register 2							
GPIOCTRL2		0x5000001B			0x0F		
W	W	W	W	W	W	W	W
IOCEN11	IOCEN10	IOCEN9	IOCEN8	IODEF11	IODEF10	IODEF9	IODEF8
MSB							LSB

Bit7-4 **IOCEN[11:8]**: GPIO wakeup-on-change enable pin.

0 = Wakeup disabled

1 = Wakeup enabled

Bit3-0 **IODEF[11:8]**: GPIO pins default state, used to detect pin change.

Register 44 GPIO Control Register 3							
GPIOCTRL3		0x5000001C			0x00		
W	W	W	W	W	W	W	W
IOCEN7	IOCEN6	IOCEN5	IOCEN4	IOCEN3	IOCEN2	IOCEN1	IOCEN0
MSB							LSB

Bit7-0 **IOCEN[7:0]**: GPIO wakeup-on-change enable pin.

0 = Wakeup disabled

1 = Wakeup enabled

Register 45 GPIO Pull-Up and Pull-Down Control Register 1							
GPIOPUL1		0x5001000D			0xAA		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
PU11	PD11	PU10	PD10	PU1	PD1	PU0	PD0
MSB							LSB

Bit7 **PU11**: IO pin Pull-Up control bit.
0 = Pull-Up Active
1 = Pull-Up Inactive

Bit6 **PD11**: IO pin Pull-Down control bit.
0 = Pull-Down Inactive
1 = Pull-Down Active

Bit5 **PU10**: IO pin Pull-Up control bit.
0 = Pull-Up Active
1 = Pull-Up Inactive

Bit4 **PD10**: IO pin Pull-Down control bit.
0 = Pull-Down Inactive
1 = Pull-Down Active

Bit3 **PU1**: IO pin Pull-Up control bit.
0 = Pull-Up Active
1 = Pull-Up Inactive

Bit2 **PD1**: IO pin Pull-Down control bit.
0 = Pull-Down Inactive
1 = Pull-Down Active

Bit1 **PU0**: IO pin Pull-Up control bit.
0 = Pull-Up Active
1 = Pull-Up Inactive

Bit0 **PD0**: IO pin Pull-Down control bit.
0 = Pull-Down Inactive
1 = Pull-Down Active

Register 46 GPIO Pull-Up and Pull-Down Control Register 2							
GPIOPUL2		0x50010010			0xAA		
R/W	R/W	Reserved	Reserved	Reserved	Reserved	R/W	R/W
PU5	PD5	-	-	-	-	PU2	PD2
MSB							LSB

Bit7 **PU5**: IO pin Pull-Up control bit.
0 = Pull-Up Active
1 = Pull-Up Inactive

Bit6 **PD5**: IO pin Pull-Down control bit.
0 = Pull-Down Inactive
1 = Pull-Down Active

Bit1 **PU2**: IO pin Pull-Up control bit.
0 = Pull-Up Active
1 = Pull-Up Inactive

Bit0 **PD2**: IO pin Pull-Down control bit.
0 = Pull-Down Inactive
1 = Pull-Down Active

Register 47 GPIO Pull-Up and Pull-Down Control Register 3							
GPIOPUL3		0x50010011			0xAA		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
PUB9	PD9	PUB8	PD8	PUB7	PD7	PUB6	PD6
MSB							LSB

Bit7 **PUB11**: IO pin Pull-Up control bit.
Bit6 **PD11**: IO pin Pull-Up control bit.
Bit5 **PUB10**: IO pin Pull-Up control bit.
Bit4 **PD10**: IO pin Pull-Up control bit.
Bit3 **PUB1**: IO pin Pull-Up control bit.
Bit2 **PD1**: IO pin Pull-Up control bit.
Bit1 **PUB11**: IO pin Pull-Up control bit.
Bit0 **PUB11**: IO pin Pull-Up control bit.

Register 48 GPIO Read Enable Register 0							
GPIORE0		0x50010012			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0
MSB							LSB

Bit7-0 **RE[7:0]**: GPIO read enable bit.

0 = Read Disabled

1 = Read Enabled

Register 49 GPIO Read Enable Register 1							
GPIORE1		0x50010013			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
PROGATT3	PROGATT2	PROGATT1	PROGATT0	RE11	RE10	RE9	RE8
MSB							LSB

Bit7-4 **PROGATT[3:0]**: PIR ProgTreeAtten, when not in use write 0000b

Attenuation settings	
B[3:0]	**Attenuation**
0	**0dB**
1	**2dB**
2	**4dB**
3	**6dB**
4	**8dB**
5	**10dB**
6	**12dB**
7	**14dB**
8	**16dB**
9	**18dB**
A	**20dB**
B	**22dB**
C	**24dB**
D	**26dB**
E	**26dB**
F	**26dB**

Bit3-0 **RE[11:8]**: GPIO read enable bit.

0 = Read Disabled

1 = Read Enabled

9.15 CHARGE PUMP

HeimdallSlave implements a charge pump used to generate 3V used by the Blue LED. This voltage is necessary in the event of the power supply voltage drop below 3V and the system needs to turn on the Blue LED, which requires a minimum of 3V to operate properly.

The following table contains the characteristics of the charge pump:

Table 22 Charge Pump Main Characteristics

Name	Conditions	Min	Typ	Max	Unit
Vout	Vdd = 2.2V	2.6			V
Iout			1		mA

9.15.1 Charge Pump Registers

Only the following Register is required to enable the charge pump.

Register 50 GPIO Control Register							
GPIOCTRL		0x50000017			0x00		
Reserved	R/W	R/W	R/W	R/W	R/W	R/W	R/W
-	LEDCPEN	RLEDEN	BLEDEN	GPIOOE11	GPIOOE10	GPIOOE9	GPIOOE8
MSB							LSB

Bit6 **LEDCPEN**: LED's Charge Pump enabled bit, necessary for the operation of the Blue LED output if the Vdd voltage is lower than 3.0V.
0 = Charge Pump disabled
1 = Charge Pump enabled

Bit5 **RLEDEN**: Red LED enable bit.
0 = LED disabled
1 = LED enabled

Bit4 **BLEDEN**: Blue LED enable bit.
0 = LED disabled
1 = LED enabled

Bit3-0 **GPIOOE[11:8]**: Output enable control.
0 = Pin output is disabled
1 = Pin output is enabled

Example Code: Enable both LEDs and the charge pump for the Blue LED.

```
LED_Control( CPON, RLEDON, BLEDON ); //Enable both LEDs and the charge pump
```

9.16 LED TRIM

LED current levels are controlled by LED Trim Register

Register 51 LED TRIM							
LEDTRIM		0x50010002			0x82		
Reserved	R/W	R/W	R/W	R/W	R/W	R/W	R/W
LEDR_ISEL3	LEDR_ISEL2	LEDR_ISEL1	LEDR_ISEL0	LEDB_ISEL3	LEDB_ISEL2	LEDB_ISEL1	LEDB_ISEL0
MSB							LSB

Bit7-6 **LEDR_ISEL[3:0]**: Red LED current level control.

Bit3-0 **LEDB_ISEL[3:0]**: Blue LED current level control.

The current range and resolution are determined by BLUEWR(Bit 2) and REDWR(Bit 1) of ADCCTRL2 (0x5001000A).

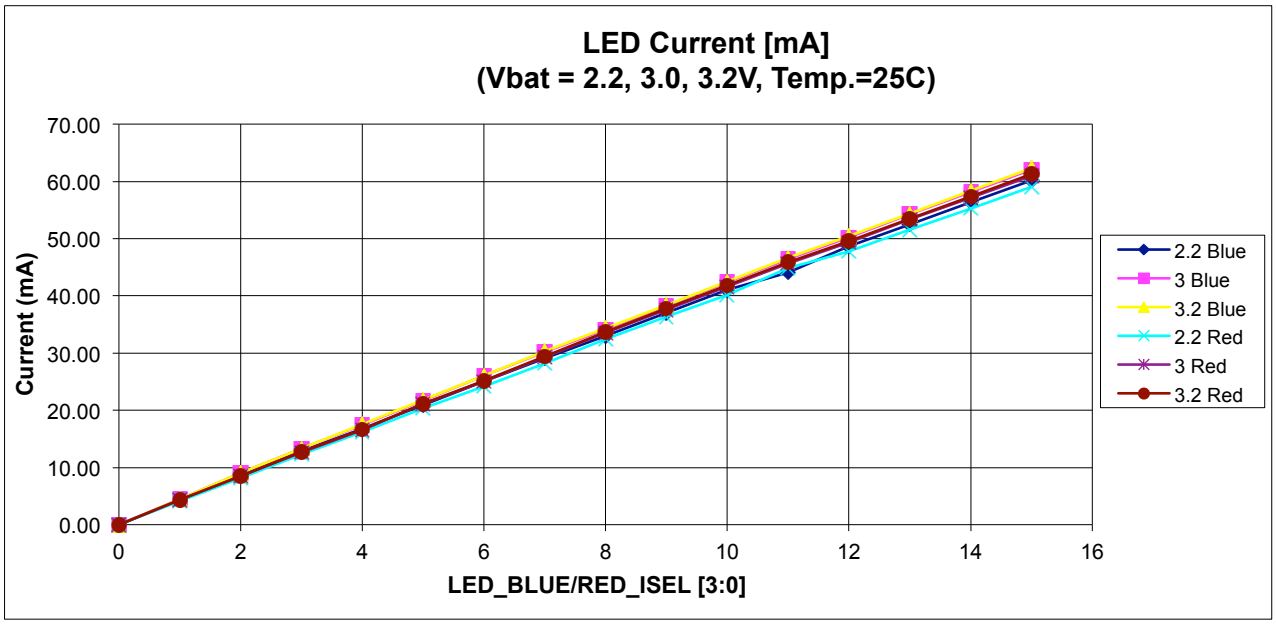


Figure 25: LED Current Consumption vs LED_BLUE_ISEL[3:0] / LED_RED_ISEL[3:0]

9.17 WAKE-UP TIMER

In addition to the Timer0/1/2 and the WDT, HeimdallSlave implements a timer capable of waking-up the microcontroller from deep sleep (halt) state.

The Wake UP timer is a Timer used to allow for recovery from deep sleep (halt), including when the microcontroller is disconnected from its power supply.

9.17.1 Wake-Up Registers

The following register controls the wake-up timer:

Register 52 Wake Up Timer							
WKTIME		0x50000011			0x00		
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
WPT7	WPT6	WPT5	WPT4	WPT3	WPT2	WPT1	WPT0
MSB							LSB

Bit7-4 **WPT[7:4]**: Mantissa

Bit3-0 **WPT[3:0]**: Exponent

The wake up time is defined as

where the valid exponent is between 3 and 12.

Additionally the user can enable this timer to generate an interrupt:

```
NVIC_EnableIRQ( WAKEUP_IRQn );
```

NOTE: It uses the 10 kHz Internal RC oscillator as clock source.

10.0 REFERENCES

ISM Band references:

http://en.wikipedia.org/wiki/ISM_band

http://ecfr.gpoaccess.gov/cgi/t/text/text-idx?c=ecfr&sid=8a0249759d545fa2c9ea0840b08bb817&rgn=div5&view=text&node=47:1.0.1.1.14&idno=47)

11.0 CONTACTS

United States
32 Journey
Aliso Viejo, California 92656, USA
Tel: +1 949-608-0854
sales@indiesemi.com

China
232 Room, Donghai Wanhao Plaza,
South Hi-tech 11th Road, Hi-tech Industry Park,
Nanshan District, Shenzhen, China.
Tel: +86 755-86116939

Scotland
5th Floor, The Auction House,
63a George Street
Edinburgh EH2 2JG
Tel: +44 131 718 6378

http://www.indiesemi.com/

Important Notice

indie semiconductor reserves the right to make changes, corrections, enhancements, modifications, and improvements to indie semiconductor products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on indie semiconductor products before placing orders. indie semiconductor products are sold pursuant to indie semiconductor's terms and conditions of sale in place at the time of order acknowledgement. Purchasers are solely responsible for the choice, selection, and use of indie semiconductor products and services described herein. indie semiconductor assumes no liability for the choice, selection, application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by indie semiconductor by this document.

The materials, products and information are provided "as is" without warranty of any kind, whether express, implied, statutory, or otherwise, including fitness for a particular purpose or use, merchantability, performance, quality or non-infringement of any intellectual property right. Indie semiconductor does not warrant the accuracy or completeness of the information, text, graphics or other items contained herein. indie semiconductor shall not be liable for any damages, including but not limited to any special, indirect, incidental, statutory, or consequential damages, including without limitation, lost of revenues or lost profits that may result from the use of the materials or information , whether or not the recipient of material has been advised of the possibility of such damage.

Unless expressly approved in writing by two authorized indie semiconductor representatives, indie semiconductor products are not designed, intended, warranted, or authorized for use as components in military, space, or aircraft, in systems intended to support or sustain life, or for any other application in which the failure or malfunction of the indie semiconductor product may result in personal injury, death, or severe property or environmental damage.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.