iND83207 B0

indie's highly integrated, microcontroller-based, ultrasonic parking assist ASSP

31st January 2022 v1.1 Datasheet

Table of Contents

iND83207 documentation

Contents:

1 Revision History

Rev #	Date	By	Description
1.1	31 Jan 2022	MT	Release for PPAP, including updated package outline drawing for dual source packaging house
1.0	01 Sep 2021	MT	Release for PPAP, including diagram of device orientation in tape & reel packaging
0.9	04 May 2021	MT	Corrected USTRX FIFO description from little to big endian
0.8	23 Feb 2021	MT	Minor formatting update
0.7	17 Feb 2021	MT	Added appendix for errata
0.6	05 Feb 2021	MT	Updated block diagram and register map
0.5	04 Jan 2021	MT	Pinout updated
0.4	07 Dec 2020	MT	Range updated
0.3	09 Sep 2020	MT	Interrupt table added
0.2	21 Aug 2020	MT	Range updated following testing
0.1	17 Aug 2020	MT	First release for B0

Ultrasonic Park Assist ASIC

2 iND83207 System Overview

CPU Architecture:

- ARM Cortex-M0 processor running at 16MHz
- System Tick Timer (SysTick – 24 bits, interruptible)
- Standard SWD Serial Wire Debugger
- Built-in Nested Vectored Interrupt Controller (NVIC)
- Programmable Watch-Dog Timer

Memory:

- 64kBytes of FLASH Memory (Includes data area software definable by customer)
- 16kBytes of SRAM

Features:

- Integrated power management with direct connect to +12V car battery

- Two regulated current drivers for driving the primary side of a center-tapped transformer, with the secondary side connected to an ultrasonic transducer. Capable of 30kHz to 70kHz burst frequencies with programmable current level and programmable burst length
- Low noise ultrasound receiver with programmable gain
  - Digital signal processing with integrated digital filtering
  - One Master and one Slave LIN 2.2 controllers
  - Two LIN pins, each can be configured to operate as
  - LIN master or LIN slave
- firmware-controlled GPIO
- LIN input/output function can mux between LIN controller, PWM, or firmware-controlled GPIO
- Five 3.3V-level GPIOs with analog input support to housekeeping ADC
- Three VBAT-level GPIOs with analog input support to housekeeping ADC
- Integrated junction temperature sensor
- Integrated high-accuracy calibrated reference oscillator for accurate time measurement using a frequency counter function
- Integrated one-time programmable (OTP) memory for unique device traceability

Quality / Reliability compliance to:

- AEC-Q100 Grade 2 Qualified (Operating Ambient Temperature range -40C to +105C)
- LIN Specification Package 2.2A and J2602-1 Nov 2012 and J2602-2 Nov 2005
- ISO 17987 (2016): Road Vehicles – Local Interconnect Network (LIN)
- Compliant to all major environmental regulations (RoHS, Sony Green, Conflict Free)

Package:

- 24-pin QFN 4x4mm with exposed pad

2.1 Application Description

iND83207 provides a completely integrated solution for high performance ultrasonic automotive parking assist applications. iND83207 integrates a powerful 32-bit ARM M0 with 64kB of FLASH and 16kB of SRAM. A portion of the FLASH integrates proprietary DSP algorithms which eases the hardware requirements of the devices and lowers overall system cost.

iND83207 also includes an integrated power management block (PMU) directly connected to the car battery, from which all the supplies required by the ASIC are generated. iND83207 integrates two regulated current drivers for driving the primary side of a center-tapped transformer, with the secondary side connected to an ultrasonic transducer. The driving frequency and number of pulses can be accurately programmed to match the transducer resonant frequency and the application requirements. iND83207 also enables the measurement of transducer reverberation so that the resonant frequency can be accurately measured, and the driving frequency tuned to match, for maximum power transfer.

Sensing of objects is performed by receiving the reflected echo signals via the ultrasonic transducer. The received echo is amplified with a high precision analog front end which includes a low noise amplifier (LNA) followed some stages of programmable gain amplifiers (PGAs) and the signal is then fed into an ADC. Digital signal processing (DSP) of the signals follows the ADC and the output of this DSP can be captured into microcontroller memory. Echo detection and association algorithms are implemented in firmware.

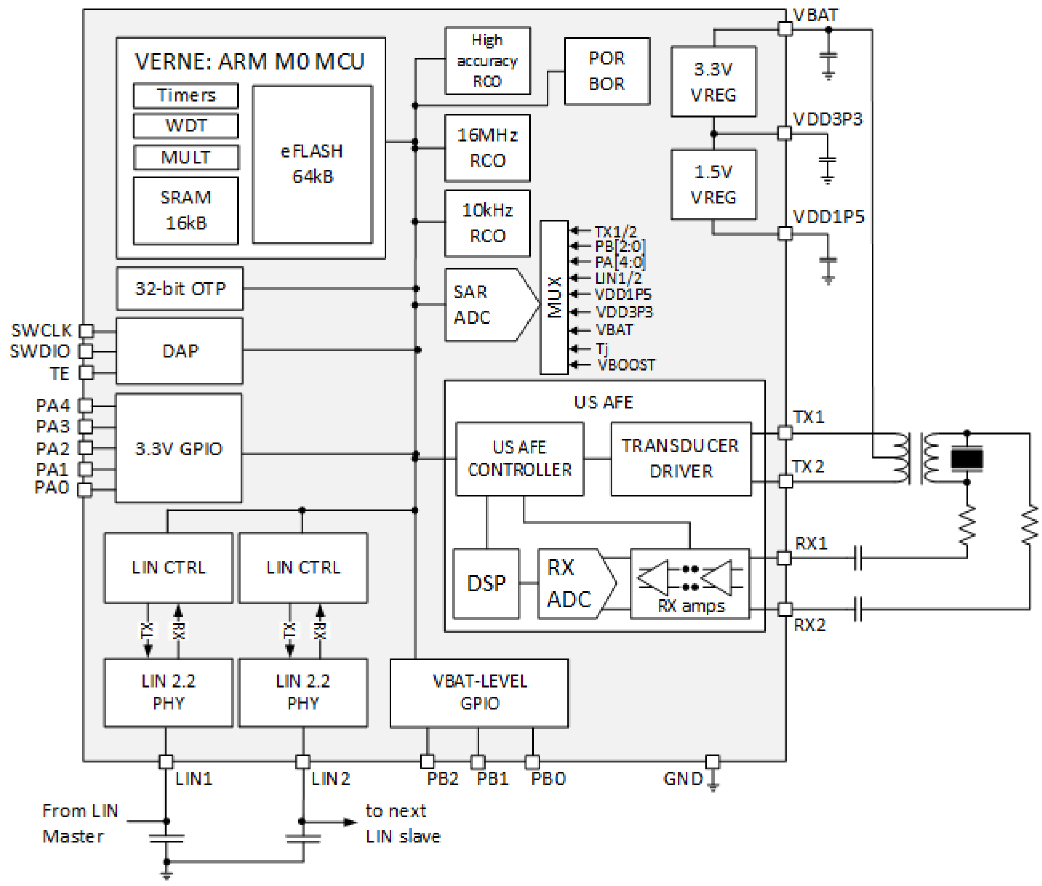
Additionally, the IC integrates a LIN master controller, a LIN slave controller and associated transceivers. The two LIN pins can operate as LIN transceivers, or can be used as high-voltage GPIOs. There are three additional high-voltage (VBAT-level) GPIOs which can provide 10mA of current source capability.

Five 3.3V-level GPIOs are provided for any requirements the customer wants to add, such as some methods of slave node addressing.

An 8-bit housekeeping ADC allows firmware to monitor all supply voltages, a temperature sensor, the LIN1/2 pins, the PA[4:0] pins and the PB[2:0] pins. This enables system diagnostics to be performed.

2.2 Functional Diagram

The figure below shows the block diagram of the IC where the entire ultrasonic parking assist application is created with an external transducer and transformer and a few passive components.



iND83207 block diagram

2.3 Pin Description

..table:: Pin List

Pin number	Pin Name	Type	Voltage	Direction	Description
23	VBAT	Supply	VBAT	Input	Battery supply input
6	VDD3P3	Supply	VDD3P3	I/O	3.3V regulator decoupling capacitor
16	VDD1P5	Supply	VDD1P5	I/O	1.5V regulator decoupling capacitor
1	TX1	Analog	VBAT	I/O	Ultrasonic transducer driver pin 1
2	TX2	Analog	VBAT	I/O	Ultrasonic transducer driver pin 2
4	RX1	Analog	VDD3P3	Input	Ultrasound receiver pin 1
5	RX2	Analog	VDD3P3	Input	Ultrasound receiver pin 2
19	LIN1	Analog	VBAT	I/O	LIN1 Pin
18	LIN2	Analog	VBAT	I/O	LIN2 Pin
11	PA0	3V3IO	VDD3P3	I/O	General purpose IO
10	PA1	3V3IO	VDD3P3	I/O	General purpose IO
9	PA2	3V3IO	VDD3P3	I/O	General purpose IO
8	PA3	3V3IO	VDD3P3	I/O	General purpose IO
7	PA4	3V3IO	VDD3P3	I/O	General purpose IO
22	PB0	HVIO	VBAT	I/O	General purpose HV IO
21	PB1	HVIO	VBAT	I/O	General purpose HV IO
20	PB2	HVIO	VBAT	I/O	General purpose HV IO
3	TE	3V3IO	VDD3P3	I/O	Test enable
14	SWDIO	3V3IO	VDD3P3	I/O	Debugger data
13	SWCLK	3V3IO	VDD3P3	Input	Debugger clock
15	PROG	Supply	•	I/O	Programming power supply for fuses, engineering samples only
EP	GND	Ground			Exposed pad
12	NC				No Connect
17	NC				No Connect
24	NC				No Connect

Pin state upon power-on reset:

- Unless otherwise noted, all pins default to tristate/Isolation mode (Hi-Z) upon power-on reset.

3 Electrical Specifications

3.1 Absolute Maximum Ratings (AMR)

Parameter	Conditions	Min.	Max.	Unit
VBAT	Up to 500ms	-0.3	45	V
VBAT	ISO 7637-2 pulse 1, VBAT=13.5V, TA=23°+/-5C, test pulse applied to VBAT via reverse polarity diode and more than 1uF capacitor	-100		V
VBAT	ISO 7637-2 pulse 2 VBAT=13.5V, TA=23°+/-5C, test pulse applied to VBAT via reverse polarity diode and more than 1uF capacitor		75	V
VBAT	ISO 7637-2 pulses 3A, 3B, VBAT=13.5V, TA=(23+/-5)°C, test pulse applied to VBAT via reverse polarity diode and more than 1uF capacitor	-150	100	V
VBAT	ISO 7637-2 pulses 5b VBAT=13.5V, TA=(23+/-5)°C, test pulse applied to VBAT via reverse polarity diode and more than 1uF capacitor		45	V
LIN1, LIN2	Up to 500ms. Note that the negative end of the LIN voltage spec is applicable only when the GND is lost.	-40	40	V
LIN1, LIN2	ISO 7637-2 pulse 1, VBAT=13.5V, TA=23°+/-5C, test pulse applied to LIN via 1nF capacitor	-100		V
LIN1, LIN2	ISO 7637-2 pulse 2, VBAT=13.5V, TA=23°+/-5C, test pulse applied via 1nF capacitor		75	V
LIN1, LIN2	ISO 7637-2 pulses 3A, 3B, VBAT=13.5V, TA=(23+/-5)°C, test pulse applied via 1nF capacitor	-150	100	V
VDD3P3		-0.3	3.6	V
TX1, TX2	Up to 500ms	-0.3	45	V
PA[4:0], SWCLK, SWDIO, RX1, RX2		-0.3	3.6	V
PB[2:0]	Up to 500ms	-0.3	VBAT+0.3	V
Ambient Temperature		-40	105	°C
Junction Temperature		-40	125	°C
Storage Temperature		-55	150	°C

3.2 Reliability (ESD and Latch Up)

Parameter	Conditions	min	max	unit
All pins	HBM, JEDEC/ AEC-Q100 Conditions	-2	+2	kV
All Pins	CDM, JEDEC/ AEC-Q100 Conditions	-500	+500	V
All pins	Latchup, JEDEC JESD78 Level A	-100	+100	mA

3.3 EMC Information

The target of the iND83207 product is to minimize the external components required to meet the required EMC specifications. Final recommendations on board layout will be provided to the customer to minimize the need for external components.

3.4 Electrical Characteristics

Electrical Characteristics are valid over the full junction temperature range of Tj = -40°C to +125°C and a supply range of 7V ≤ VBAT ≤ 18V, with typical values quoted for VBAT=12V, unless otherwise noted.

Parameter	Conditions	min	typ	max	unit
VBAT Input Voltage	No damage to sensor			30	V

Parameter	Conditions	min	typ	max	unit
VDD3P3 Output Voltage	Across full output load range	3.0	3.3	3.6	V
Current Consumption	MCU on, US TX and RX active,		10	17.5	mA
Recommended VDD3P3 decoupling capacitor			1		µF
Clocks					
System RC oscillator (HFRC) frequency		15.2	16	16.8	MHz
Reference oscillator			550		kHz
Auxiliary RC Oscillator (LFRC) frequency		3.5	10	16.5	kHz
POR/BORs					
POR (VDD3P3)	Voltage below which ASIC is reset		1.55	1.75	V
BOR (VDD3P3)	Voltage below which ASIC operation is halted	2.3	2.4		V
BOR (VDD1P5)	Voltage below which ASIC operation is halted	1.28	1.35		V
Junction Temperature Sensor					
Temperature range		-40		125	°C
Temperature Accuracy		-10		+10	°C
Ultrasound system					
Distance detection range		0.2		2.0	m
Distance detection accuracy			1		cm
Transducer Driver pins TX1 / TX2					
Driver frequency range, Fdrv	Guaranteed by design	40		70	kHz
Driver Current at min code	V(TXx)=6V, IDRV code=0, inferred ATE measure	175	200	225	mA
Driver Current at max code	V(TXx)=6V, IDRV code=31, inferred ATE measure	395	450	505	mA
Current Adjustment Steps			8.1		mA
Leakage Current	V(TXx)=VBAT, Tj=105°C max	-1		1	µA
Driver Voltage Drop	Output current drops by less than 20%, code=0. Guaranteed by initial characterization.		0.6	1.2	V
Number of drive periods in a burst		1		64	
Receiver pins RX1 / RX2					
Minimum Gain	Fdrv = 51.2kHz. Setting AFE_gain[4] at room temperature	44	52	61	dB
Maximum Gain	Fdrv = 51.2kHz. Setting AFE_gain[11] at room temperature	78	88	98	dB
Analog gain change from calibrated value	Fdrv = 51.2kHz. Using RX gain compensation in FW for temperature. Settings from AFE_gain[5] to AFE_gain[10]	-2		2	dB
Gain step size	Gain adjustment is implemented using coarse steps in the analog part of the receiver and fine steps using digital adjustment in FW			0.1	dB

Parameter	Conditions	min	typ	max	unit
Input Impedance	Code 0 Code 1 Code 2 Code 4		525 130 86.5 16.7		kΩ kΩ kΩ kΩ
Input-referred noise	@ 51.2kHz, at final output from DSP filtering. Guaranteed by design.		5	9	nV/(Hz)$^{1/2}$
Reverb frequency measurement accuracy	Reverb frequency measurement is achieved by measuring many individual cycles of the reverberation signal and accuracy is assured by firmware.	-200		200	Hz
8-bit SAR ADC					
Conversion rate	Guaranteed by design			200	ks/s
INL	Slowest selectable conversion rate	-1		1	LSB
DNL	Slowest selectable conversion rate	-1		1	LSB
VDD3P3-level GPIOs, PA[4:0]					
Input voltage for logic low				0.3* VDD3P3	V
Input voltage for logic high		0.7* VDD3P3			V
Logic low output level	Iol=-5mA			0.4	V
Logic high output level	Ioh=5mA. Current is supplied from VDD3P3.	2.4			V
Pull-down resistance			10		kΩ
Pull-up resistance			10		kΩ
VBAT-level GPIOs, PB[2:0]					
Input voltage for logic low				0.8	V
Input voltage for logic high		2			V
Strong pull-down current			5		mA
Strong pull-up current			-10		mA
Weak pull-down current			100		µA
Weak pull-up current			- 100		µA
LIN1, LIN2 – refer to LIN 2.2 specification, VBUS=LIN pin					
IBUS_LIM	Current limitation for driver dominant state driver on VBUS = VBAT_max=18V	40		200	mA
Rslave	LIN configured as a slave	20	30	60	kΩ
Rmaster	LIN configured as a master	0.75	1	1.25	kΩ
IBUS_PAS_dom	Input Leakage Current at the Receiver including Pull-Up Resistor driver off VBUS = 0V VBAT= 12V	-1			mA
IBUS_PAS_rec	Driver off, VBUS>VBAT 8V<VBAT<16V 8V<VBUS<16V			20	µA
Device Bus Leakage Current Ground Disconnected	VBAT= VGND=12V, 0V<VBUS<18V J2602	-100		100	µA
Device Bus Leakage current VBAT disconnected	0V<VBUS<18V, VBAT=VGND=0V	-23		23	µA

Parameter	Conditions	min	typ	max	unit
BUS_VOH Transmitter dominant voltage	Load 500Ohms, driver open drain active	0.8		1.0	VSUP
BUS_VOL Transmitter recessive voltage	Driver open drain high impedance	0.0		0.2	VSUP
LIN pin input capacitance	Note that LIN 2.2A spec 220pF typ, 250pF max as total node capacitance at the connector including the physical bus driver and all other components including C_{LIN}			20	pF
VBUSdom	Receiver dominant state			0.4	VSUP
VBUSrec	Receiver recessive state	0.6			VSUP
VBUS_CNT	Center point Receiver VBUS_CNT = (Vth_dom+ Vth_rec)/2	0.475	0.5	0.525	VSUP
Vhys	Receiver hysteresis VHYS = Vth_rec -Vth_dom	0.075		0.175	VSUP
Trx_pd	propagation delay of receiver. Guaranteed by initial characterization. C_{RXD} load 20pF (RX output of transceiver, internal node, access in test mode) minimum slew rate for the LIN rising and falling edges is 50V/µs			6	µs
Trx_sym	symmetry of receiver propagation delay. Guaranteed by initial characterization. rising edge w.r.t. falling edge C_{RXD} load 20pF C_{RXD} load 20pF (RX output of transceiver, internal node, access in test mode)	-2		+2	µs
LIN1, LIN2 Timing parameters (CBUS ; RBUS): (1nF; 1kΩ/ (6.8nF;660Ω / (10nF;500Ω)	**Guaranteed by initial characterization.**				

D1 Duty Cycle (20kbits/s)	THRec(max) = 0.744 x VSUP;
	THDom(max) = 0.581 x VSUP;
	VSUP = 7.0V...16V; tBit = 50µs;
	D1 = tBus_rec(min) / (2 x tBit)
		0.396			

D2 Duty Cycle (20kbits/s)	THRec(min) = 0.422 x VSUP;
	THDom(min) = 0.284 x VSUP;
	VSUP = 7.6V...16V; tBit = 50µs;
	D2 = tBus_rec(max) / (2 x tBit)
				0.581	

D3 Duty Cycle (10.4kbits/s)	THRec(max) = 0.778 x VSUP;
	THDom(max) = 0.616 x VSUP;
	VSUP = 7.0V...16V; tBit = 96µs;
	D3 = tBus_rec(min) / (2 x tBit)
		0.417			

D4 Duty Cycle (10.4kbits/s)	THRec(min) = 0.389 x VSUP;
	THDom(min) = 0.251 x VSUP;
	VSUP = 7.6V...16V; tBit = 96µs;
	D4 = tBus_rec(max) / (2 x tBit)
				0.590	

tBus_rec(min)-tBus_dom(min)	Δt3, 10.4kbs operation, low speed mode			15.9	µs
tBus_rec(min)-tBus_dom(max)	Δt4, 10.4kbs operation, low speed mode			17.28	µs
Piezo speaker driver function on LIN1/2					
Current output	Code 0, V_{LIN}=3.5V Code 1, V_{LIN}=3.5V Code 2, V_{LIN}=3.5V Code 3, V_{LIN}=7V	. . 20 .	10 20 40 80	. . 60 .	mA mA mA mA
Drive modulated frequency	PWM with duty from 0.2 to 0.8	0.5		3.5	kHz
LIN1 / LIN2 high-speed mode					

Parameter	Conditions	min	typ	max	unit
Baud rate	VBAT=12.0V, external pull-up resistor 220Ω to VBAT, T_A=20C-30C, C_{LIN} < 4nF This mode will use the UART controller instead of the LIN master or slave controllers.			62	kbaud

4 Device Functional Description

4.1 Ultrasound analog front-end

iND83207 makes use of ultrasonic distance measuring principles by transmitting a pulse and then measuring the time it takes to receive the reflected or echo, pulse back at the transducer. The physical relationship between the distance from the transducer to the object of interest is given by distance X = (c * t) / 2, where c is the speed of sound in air (c = 343 m/s at 20°C) and t is the time it takes for a transmitted pulse to return back to the transducer.

4.1.1 Transducer Driver and Burst Generation

The drive patterns are produced by the ultrasound AFE controller. A transmit cycle is of programmable length from 1 to 64 cycles with accurate frequency control.

The ultrasound transducer is driven through a transformer. The primary side of the transformer has a centre-tap connected to VBAT, and iND83207 drives the two ends alternately with current-controlled pull-downs. The drive current strength is controllable through user registers.

4.1.2 Ultrasound Receiver

The signal at the transducer is AC coupled to iND83207 via a series resistor and capacitor from both sides of the transducer, as shown in the block diagram in Figure 1.

At the receiver, the differentially sensed signal goes through a series of amplifiers with programmable gain followed by an ADC. After digitization, the signal is processed by a series of digital filter operations. The final output from the filters is a stream of numbers which indicate the level of signal at the transducer resonant frequency. This “envelope” data will be available to the microcontroller at a time resolution which enables the time of flight to be calculated to the required accuracy.

The iND83207 receiver will be designed for the given noise target, where this will include all external components. Another target for the iND83207 design is to significantly reduce the harmonic component of the input signal in the final output envelope data (this is an improvement relative to the original Qin design).

The implementation of the receive chain is shown below.

./images/USRX_diagram.png

US Rx Diagram

Receiver input offset cancellation

Voltage offset at the receiver input is automatically cancelled by iND83207.

Receiver gain control

The receiver gain can set by firmware writing to a specific control register. The gain may be fixed for the full length of an envelope capture. Alternatively, firmware can adjust the gain throughout the envelope capture period.

The LNA and four amplifiers are cascaded to generate the receiver gain as shown in US Rx Diagram. LNA gain is defined by PMOS based gm with a resistive load. The following stages,PGA1/PGA2 gain is calculated by PMOS based resitive load gm with a constant gm current bias. The last two stages(PGA3/PGA4) gain is simply the ratio of the feedback resistor divided-by the input resistor. PGA3 and PGA4 are NMOS differential OP Amp with resitive feedback for linearity. There is an option to bypass PGA1 and PGA2 for a lower RX gain.

Dynamic gain control (DGC)

Dynamic adjustment of gain in the receiver is supported through hardware which adjusts the signal gain. The user controls both gain levels and points in time where the change is applied.

The gain adjustment may be applied either to the gain through the RX amplifier chain, or through the gain control in the DSP.

This function enables the user to increase the gain as the time from transmission increases, so that the gain is higher when the expected echo signal is smaller.

Dynamic gain control will be combined with dynamic adjustment of the thresholds in the echo detection algorithm to optimize the system.

4.1.3 Echo detection and echo association

The envelope data will be read from the DSP in system RAM for processing which will normally be done after the full envelope has been captured.

Echo detection and echo association algorithms of any form can be implemented in firmware.

4.1.4 Transducer reverb frequency measurement

Measurement of the resonant frequency of the transducer is vital for proper functioning of the system. The system uses this information to adjust the driving frequency (to maximize the output power).

iND83207 will enable the measurement of the transducer reverb frequency at the end of any drive phase when active damping is not in operation. The ability to measure this feature has been demonstrated in Qin A1 to work very accurately using a firmware-based method.

iND83207 will implement a hardware method which will give the same accuracy as is demonstrated on Qin A1. This will enable the gathering of a sequence of measurements of the individual reverberation periods and will use the micro-controller block-transfer engine (BTE) to transfer these from the ASIC die to SRAM on the micro-controller die. Firmware needs only to set this up, it is not involved as the data is captured to SRAM. At the end of the reverberation period, firmware can read all the captured data and calculate the reverberation frequency from that data.

Experimentation with Qin A1 has shown that the reverberation frequency is not static immediately after the end of the transmit period. It has been shown that the reverberation frequency in the first few cycles can be in error by more than 2kHz, and that it can take 8 cycles before this frequency settles to within 200Hz of the natural frequency. The proposed solution gives the ability to measure as many cycles of the reverberation frequency as desired, and firmware can analyze the results to accurately estimate the reverberation frequency.

4.1.5 Transducer reverb duration measurement

The transducer reverb duration may be used as a diagnostic measure of the health of the transducer. This duration can be measured from the envelope data captured on any transmit phase, using firmware to measure when the energy at the resonant frequency falls below a threshold.

5 Cortex-M0 Processor Subsystem

The iND83207 SiP includes an embedded microcontroller subsystem based upon the ARM Cortex M0 core. This processor includes a 64 kByte program flash memory and 16 kByte SRAM. It also includes three general-purpose 32-bit timers plus a dedicated watchdog timer. Additionally, it includes a Nested Vector Interrupt Controller (NVIC) to schedule hardware interrupts. The ARM Cortex M0 implementation supports 8-bit, 16-bit, and 32-bit reads/writes to peripherals.

Memory Map Summary

Address range	Memory	Description
0x00000000 - 0x0000FFFF	Flash	64 Kbytes of Flash Memory, user programmable
0x00010000 - 0x000101FF	Flash NVR 1	512 bytes of the NVR 1 sector
0x00010200 - 0x000103FF	Flash NVR 2	512 bytes of the NVR 2 sector
0x00010400 - 0x000105FF	Flash NVR 3	512 bytes of the NVR 3 sector
0x00010600 - 0x000107FF	Flash NVR 4	512 bytes of the NVR 4 sector
0x00010800 - 0x000109FF	Flash NVR 5	512 bytes of the NVR 5 sector
0x00010A00 - 0x1FFFFFFF	N/A	Reserved
0x20000000 - 0x20003FFF	SRAM	16 Kbytes of SRAM
0x20004000 - 0x4FFFFFFF	N/A	Reserved
0x50000000 - 0x5000007F	ASIC Peripherals	128 Byte peripheral fast access
0x50000080 - 0x5000FFFF	MCU Peripherals	Block Transfer Engine
0x50010000 - 0x5001FFFF	ASIC Peripherals	64 Kbyte peripheral slow access
0x50020000 - 0x5002001F	MCU Peripherals	General Purpose Timers Controls
0x50020020 - 0x500200FF	Flash	Flash Programming/Erase Control
0x50020100 - 0xDFFFFFFF	N/A	Reserved
0xE0000000 - 0xE00FFFFF	Private Peripheral Bus	ARM peripherals
0xE0100000 - 0xEFFFFFFF	N/A	Reserved
0xF0000000 - 0xF0001FFF	System ROM tables	ARM core IDs
0xF0002000 - 0xFFFFFFFF	N/A	Reserved

5.1 ARM M0 Processor “Verne” Specification:

2 System Overview

2.1 ARM Architecture

- ARM Cortex-M0 processor
- System Tick Timer (SysTick 24 bits, interruptible)
- Serial Wire Debugger (2 pins)
- Built-in Nested Vectored Interrupt Controller (NVIC)
- Four Breakpoints and two Watch points

2.2 Additional Features

- Programmable Watch-Dog Timer (Trigger time: 213, 219 , 222 , and 232 clock cycles)
- Three General Purpose 32-bit Timers

2.3 Memory

- 64kB of Flash Program Memory (16kx32) operating at a maximum MCU freq. No Cache.
- 16kB of SRAM split into two 8KB macro (x32)

2.4 Speed

- 30MHz maximum CPU frequency with no flash read wait state
- MCU-ASIC bus frequency operates at the same speed than CPU

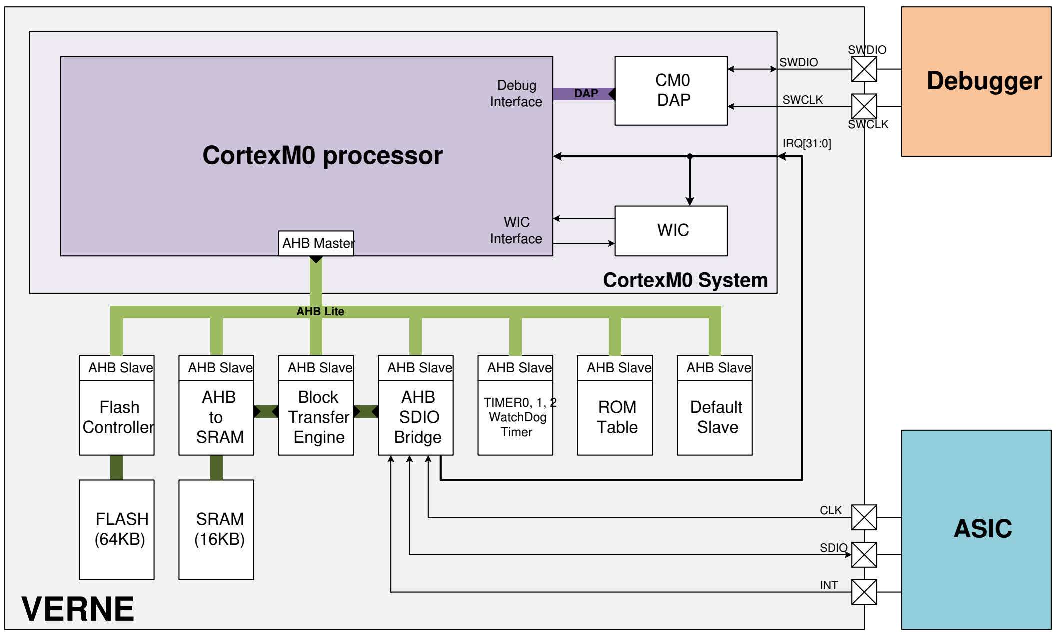
2.5 Operating Junction Temperature

- -40 °C to +125 °C

2.6 Power Use cases

- Power Off
- Idle/standby (WFI), driven by ASIC (hold the clock). Flash in standby mode.
- Active

2.7 Top Level Block Diagram



3 Memory Description and map

ARM Cortex M0 uses a unified memory model with a linear address space (Von Neumann architecture) including Flash and RAM memories as well as registers address space. The indie implementation of the Cortex M0 core uses a high density 64KB Flash cell along with 16KB of SRAM. The following table defines the several regions of the address space:

Memory map

Address range	Memory	Description
0x00000000 - 0x0000FFFF	Flash	64 Kbytes of Flash Memory, user programmable
0x00010000 - 0x000101FF	Flash NVR 1	512 bytes of the NVR 1 sector. [for internal Indie use only]
0x00010200 - 0x000103FF	Flash NVR 2	512 bytes of the NVR 2 sector. [for internal Indie use only]
0x00010400 - 0x000105FF	Flash NVR 3	512 bytes of the NVR 3 sector. [for internal Indie use only]
0x00010600 - 0x000107FF	Flash NVR 4	512 bytes of the NVR 4 sector. [for internal Indie use only]
0x00010800 - 0x000109FF	Flash NVR 5	512 bytes of the NVR 5 sector. [for internal Indie use only]
0x00010A00 - 0x1FFFFFFF	N/A	Reserved
0x20000000 - 0x20003FFF	SRAM	16 Kbytes of SRAM
0x20004000 - 0x4FFFFFFF	N/A	Reserved
0x50000000 - 0x5000007F	ASIC Peripherals	(Implementation Dependent)
0x50000080 - 0x5000FFFF	MCU Peripherals	Block Transfer Engine
0x50010000 - 0x5001FFFF	ASIC Peripherals	(Implementation Dependent)
0x50020000 - 0x5002001F	MCU Peripherals	General Purpose Timers Controls
0x50020020 - 0x500200FF	Flash	Flash Programming/Erase Control
0x50020100 - 0xDFFFFFFF	N/A	Reserved
0xE0000000 - 0xE00FFFFF	Private peripheral bus	ARM peripherals
0xE0100000 - 0xEFFFFFFF	N/A	Reserved
0xF0000000 - 0xF0001FFF	System ROM tables	ARM core IDs
0xF0002000 - 0xFFFFFFFF	N/A	Reserved

3.1 System Memory (SRAM)

MCU core implements 16kbytes of SRAM. MCU can execute codes from the SRAM memories. The SRAM macros can be put in retained state into which the supply is maintained and clock source stopped (idle). The design does not support power switch to cut off supply on non-active logic in such state and “put the burden” on the ASIC side to optimize power consumption including lowering the supply voltage without SRAM and other FF data losses.

3.2 Flash Non Volatile Memory

MCU implements a Programmable Flash Memory with x32 configuration, sector and chip erase and byte program capability. It integrates five 512bytes nonvolatile registers (NVR) sectors which are only for internal Indie use.

In normal operation the ARM core fetches instructions (or data permanently stored) from the Flash memory but it is also possible for a program to alter the content of the flash memory. The following operations can be performed in the Flash Memory:

- Byte Write
- Sector Erase
- Code Protect

For a description of the flash memory registers, please refer to the product register map. Here is a simple description of the basic features supported:

- Registers support to write/erase data to a byte, sector address
- Support programmable read wait states (The design is implemented such that the timings associated with the flash macro meet the maximum speed of the system clock requirements)
- Support system clock divider for write/erase functions
- Protection mechanism to unlock flash memory write and start flash memory byte-write
- Protection mechanism to unlock flash memory sector erase

The flash macro is 32-bit wide but the write can be done on 8-bit, 16-bit or the full 32-bit.

4 Reset

The system can be reset through the following events:

- power on reset. When the 1.5V power supply gets sensed and reach a specific threshold, a POR cell will release the reset few micro second after the 1.5V is stable.
- The POR circuit also monitors the IO voltage and it can trigger a reset if this voltage gets to a lower value than the nominal 3.3V. There is a configuration bit in the first NVR of the flash (bit 17) which will mask this reset. If this bit is set to high, the ASIC can disconnect the IO supply to save power without triggering a reset. This feature can be used when the processor is in Deep Sleep mode.

Note

There is no specific hardware to sequence the events in order to remove the IO supply. This needs to be done based on time since the removal of this IO supply will cut the communication between the processor and the ASIC.

- assertion of the RST_N pin. This pin may not be available for all the application (legacy chips do not have this pin). This pin has an internal pull-up. The reset is active low.
- System reset request of the Cortex-M0 system through the Debugger
- The ASIC can request a system reset through the Interrupt interface (code=0x1C). This is implementation dependent. All ASIC do not have this feature implemented. Check the specification of the ASIC for more information.

5 Interrupt

5.1 Interrupt Vector

The first 148 bytes of Flash Memory are organized following the standard created by ARM. In this standard the Address 0x00000 contains the top-of-stack address (four bytes). The following addresses contain interrupt vectors used by the microcontroller:

Interrupt Vector

Vector Name	Address	Comments
STACK_VALUE	0x00000	Typically set to 0x20000FFFF (Top of SRAM)

Vector Name	Address	Comments
Reset_Handler	0x00004	
Reserved	0x00008	
HardFault Handler	0x0000C	
Reserved	0x00010 to 0x00028	
SVC_Handler	0x0002C	
Reserved	0x00030 and 0x00034	
PendSV_Handler	0x00038	
SysTick_Handler	0x0003C	
Dependent on product implementation	0x00040 to 0x0007C	
Timer0_Handler	0x00080	
Timer1_Handler	0x00084	
Timer2_Handler	0x00088	
Watchdog_Handler	0x0008C	
BTE_Handler	0x00090	Block Transfer. Contact indie to get more information.
Reserved	0x00094	

All other addresses in the flash memory can be used for the user's program. The meanings of the standard interrupt vectors (Provided with the ARM Cortex M0 core) are defined in ARM's documentation. One of the sources of information is: Cortex-M0 Devices Generic User Guide

5.2 Interrupt Enabling/Disabling Process

Cortex-M0 implements a NVIC (Nested Vector Interrupt Controller) peripheral capable of handling up to 16 peripheral's interrupts. Upon reset the microcontroller can answer only to Reset, NMI (Non-Maskable Interrupt) and Hard-Fault interrupts/exceptions. All other interrupts must be enabled. To enable and disable the interrupts the user must use access the ISER (Interrupt Set Enable Register) and ICER (Interrupt Clear Enable Interrupt) registers associated with the desired interrupt.

Note

Both inline functions and all parameters are defined in the product_file.h file, which must be included in the source files. Besides that the product_file.h file contains a list of available interrupts. The format of this list is as follows:

```
typedef enum IRQn {
//******  Cortex-M0 Processor Exceptions Numbers ****************************
NonMaskableInt_IRQn          = -14,    // Non Maskable Interrupt
HardFault_IRQn               = -13,    // Hard Fault Interrupt
SVCall_IRQn                  = -5,     // SV Call Interrupt
PendSV_IRQn                  = -2,     // Pend SV Interrupt
SysTick_IRQn                 = -1,     // System Tick Interrupt

//*****  CM0IKMCU Cortex-M0 specific Interrupt Numbers **********************
IRQ00_IRQn                   = 0,      // Product specific
IRQ01_IRQn                   = 1,      // Product specific
IRQ02_IRQn                   = 2,      // Product specific
IRQ03_IRQn                   = 3,      // Product specific
IRQ04_IRQn                   = 4,      // Product specific
IRQ05_IRQn                   = 5,      // Product Specific
IRQ06_IRQn                   = 6,      // Product Specific
IRQ07_IRQn                   = 7,      // Product Specific
IRQ08_IRQn                   = 8,      // Product Specific
IRQ09_IRQn                   = 9,      // Product Specific
IRQ10_IRQn                   = 10,     // Product Specific
IRQ11_IRQn                   = 11,     // Product Specific
IRQ12_IRQn                   = 12,     // Product Specific
IRQ13_IRQn                   = 13,     // Product Specific
IRQ14_IRQn                   = 14,     // Product Specific
IRQ15_IRQn                   = 15,     // Product Specific
TIMER0_IRQn                  = 16,     // Timer 0
TIMER1_IRQn                  = 17,     // Timer 1
TIMER2_IRQn                  = 18,     // Timer 2
WATCHDOG_IRQn                = 19,     // Watchdog timer
BTE_IRQn                     = 20,     // Block Transfer Engine
SDIO_IRQn                    = 21      // Serial Data IO
} IRQn_Type;
```

6 Flash

The MCU contains a 64kB flash memory. This memory is configured as a ROM memory in the address space from 0x0000000 to 0x0000FFFF. Since this space includes the boot vector, the microcontroller boots from an address configured in this flash memory, and user code will typically execute from this memory as well. The flash memory is arranged in 512-byte sectors. Memory can be erased one sector at a time, and can be written either one byte, 16-bit or a full 32-bit word at a time. The writing and erasing of memory is handled by a memory-mapped peripheral.

6.1 Flash Read Only Memory

The flash memory is memory mapped into a read-only address range below 0x0000FFFF. These addresses cannot be written to directly. To modify the flash contents, the flash controller peripheral must be used. There are five sectors starting at 0x00010000 which are reserved for AyDeeKay production test use and cannot be written or erased by the user code. Consequentially, the sector starting at 0x00010000 cannot be erased using the sector erase command. All other sectors in this block can still be erased and bytes in these sectors can be written to by the user code.

Flash Memory Map

Starting Address	Ending Address	Description
0x00000000	0x0000FFFF	Read only access FLASH. Each sector can be erased and can be written using flash controller peripheral.
0x00010000	0x000109FF	Read only access FLASH. These sectors cannot be modified and typically contains Indie Semi information written during production test.

6.2 Flash Memory Control

The flash memory can be written and erased using a memory-mapped flash control peripheral. To avoid unintentional modification to the flash memory, an unlocking scheme is implemented which requires multiple sequential operations in a fixed order in order to erase or write the flash contents. Flash writes complete within 20µs, while sector erase operations can take up to 10ms to complete. The flash cannot be read during a write or erase operation. Since program code typically resides in flash, this will result in the program stalling for the duration of the write / erase operation unless counter-measures are taken. If continued program execution during flash modification is required, then any code which must continue to run should execute from SRAM during the flash write/erase to avoid the processor stalling from attempting to fetch its code from the flash memory. Furthermore, the interrupt vector table is located in flash memory. If an interrupt occurs and it is enabled, then the MCU will attempt to retrieve the interrupt service routine's address from the flash memory. If the flash is busy performing a write or erase when this occurs, the processor will halt until the interrupt vector can be read from the flash. The interrupts numbered 0, 1, 2 and 16 are cached so that the vector fetch will not be stalled by the flash unavailability. So for these interrupts, if the service routine is mapped to a function in SRAM then the interrupt servicing will not stall the MCU during flash modification operations. Other interrupts should be disabled prior to initiating the flash operation if a MCU stall caused by their servicing must be avoided. Disabled interrupts will still pend, so re-enabling them after completion of the flash modification operation will cause any of the disabled interrupts which had occurred to be serviced at that time.

6.3 Flash Memory Operations & Examples

The following operations can be performed in the Flash Memory:

- Sector Erase
- Byte / Half-word / word Program
- Code Protect

Sector Erase

To erase a 512 byte sector the following sequence must be followed:

1. Write an address inside the sector to be erased to the ADDR register.
2. Unlock the sector for erasure by writing the 0x66666666 pattern to the UNLOCK_ERASE register.
3. Start the sector erase process by writing the 0x99999999 pattern to the ERASE_START register.

```
FLASH_SFRS->ADDR         = 0x000050A0;          //Point to the block starting at 0x00005000
                                                //(any address 0x5000 to 0x50FF will work)
FLASH_SFRS->UNLOCK_ERASE = 0x66666666;          //Unlock sector erase
FLASH_SFRS->ERASE_START  = 0x99999999;          //Start erase process
```

Note

The erase process of a sector can take up to 10msec.

Write Byte

To write a byte the following sequence must be followed:

1. Write the flash address to be programmed to the ADDR register.
2. Write the value to be written into the DATA register. (Bits 31 to 8 are ignored)
3. Unlock the write by writing the 0x55555555 pattern to the UNLOCK_WRITE register.
4. Start the writing process by writing the 0xAAAAAAAA pattern into the WRITE_START register.

```
FLASH_SFRS->ADDR            = 0x000050BB;       //Point to the byte address
FLASH_SFRS->DATA            = 0xAB000000;       //Load byte (0xAB) to be written
                                                //Need to align with address
FLASH_SFRS->UNLOCK_WRITE    = 0x55555555;       //Unlock byte write
FLASH_SFRS->WRITE_START     = 0xAAAAAAAA;       //Start write process
```

Note

The byte write process can take up to 20µs.

Write Half-Word (16-bit) or Word (32-bit)

To write a byte the following sequence must be followed:

1. Unlock the write to the Control Operation register by writting 0xACDC_1972 in the UNLOCK_CTRL_OP register.
2. Configure the size of the write in the CTRL_OP.SIZE register. Refer to table below to define the correct value.
3. Write the flash address to be programmed to the ADDR register.
4. Write the value to be written into the DATA register.
5. Unlock the write by writing the 0x55555555 pattern to the UNLOCK_WRITE register.
6. Start the writing process by writing the 0xAAAAAAAA pattern into the WRITE_START register.

Flash Memory Map

CTRL_OP.SIZE[1:0	Number of Bytes	ADDR[1:0]	Description
00	1	00	One Byte written at address ADDR
00	1	01	One Byte written at address ADDR
00	1	10	One Byte written at address ADDR
00	1	11	One Byte written at address ADDR
01	2	00	Two Bytes written at address ADDR and (ADDR + 1)
01	2	01	Two Bytes written at address ADDR and (ADDR + 1)
01	2	10	Two Bytes written at address ADDR and (ADDR + 1)
01	2	11	FORBIDDEN
10	3	00	Three Bytes written at address ADDR, (ADDR + 1) and (ADDR + 2)
10	3	01	Three Bytes written at address ADDR, (ADDR + 1) and (ADDR + 2)
10	3	10	FORBIDDEN
10	3	11	FORBIDDEN
11	4	00	Full word written at address ADDR
11	4	01	FORBIDDEN
11	4	10	FORBIDDEN
11	4	11	FORBIDDEN

```
FLASH_SFRS->UNLOCK_CTRL_OP  = 0xACDC1972;       // Unlock Control Operation access
FLASH_SFRS->CTRL_OP.SIZE    = 0x3;              // Configure the write operation to be word
based
FLASH_SFRS->ADDR            = 0x000050B8;       // Point to the address
FLASH_SFRS->DATA            = 0xDEADBEEF;       // Load word to be written
FLASH_SFRS->UNLOCK_WRITE    = 0x55555555;       // Unlock write
FLASH_SFRS->WRITE_START     = 0xAAAAAAAA;       // Start write process
```

Note

The word write process can take up to 40µs.

6.4 Flash Code Portection

The controlled access to the flash content is based on disabling all communications with the debug interface, therefore preventing any external attack. Hence, the application code is still able to modify the Flash content. Upon Power-On Reset or Normal Reset, MCU core disables the communication with the debug interface for a small time interval (8192 system clock cycles). If the application needs to be protected it is mandatory to set the protection register with the appropriate code in the beginning of the initialization process and before the internal hardware enable the debug communication. In other words, if during this time interval the protection register is loaded with a specific pattern, then the communication remains disabled after the end of this interval and stays disabled until this register is loaded with a different pattern. To allow for debug communication the application has only to write a different value in the lock register. If a part is protected, the emulator can still erase and program the part, but first it will be required to erase the Flash content, therefore protecting it.

```
FLASH_SFRS->CODE_PROT 0xF2E11047;          //Code protection activated
```

Note

If code protection is necessary it is important to consider the time it takes from the moment the part is reset until the execution reaches the main function. In some exceptional cases where the initialization of the system (stack, heap, global variables) take more than 8K clock cycles it may be necessary to add code directly in the initialization routine.

6.5 Flash Characteristics

The flash endurance sector is greater than 20,000 cycles. The data retention is greater than 100 years at 25C.

7 Flash Oscillator

The Flash Controller needs a clock to measure some timings to controller the flash operation properly. The clock of the processor cannot be used since it may vary for a given product. The dedicated Oscillator is used to provide a 4Mhz clock to the flash controller. This section describes the behavior of the Flash Oscillator and is not relevant to the end user of the product. The Flash controller is in charge of starting and stopping the oscillator during the write/erase operations.

8 Systick Timer

This timer is an optional peripheral created by ARM and implemented in the Cortex M0 160/8. It is fully described in the Cortex-M0 Devices Generic User Guide (Chapter 4.4 Optional System Timer, Systick)

Its interrupt can be enabled as follows:

```
NVIC_EnableIRQ(SysTick_IRQn);
```

9 Timers (0, 1 and 2)

The MCU implements three identical timers: Timer0, Timer1 and Timer2. All three timers operate using the system clock as clock source. They increment at the system clock rate starting from the loaded value in the counter until they roll over from 0xFFFFFFFF to 0x00000000. At this point an interrupt is generated if enabled. The interrupt routine is responsible for reloading the value if needed as this timer does not auto-reload the original content.

9.1 Timer Operation

The operation of the Timers is quite straightforward. Load the TIMERx.COUNT with the required initial value, enable the timer and if required also enable the related interrupt. If the interrupt is enabled an additional code to handle the interrupt must be added.

Code Example1

If we are running from a 15MHz System Clock and we want to use timer1 to generate a time delay of 2msec without interrupt we can calculate the necessary timer counter:

T = 2msec×15MHz = 30000

Given that the timer counts to roll from 0xFFFFFFFF to 0x00000000 we have to take the negative value. Code fragment:

```
*TMR1REG = -30000;      //Load counting value
*TMR1CTRL = 1;          //Enable timer
while( TMR1REG < 0 );   //Wait to reach 0
*TMR1CTRL = 0;          //Disable timer
```

Using the interrupt would require to have it enabled and the interrupt handler defined. The following code generates an interrupt every 2msec under the same conditions:

```
*TMR1REG = -30000;              //Load counting value
*TMR1CTRL = 1;                  //Enable timer
NVIC_EnableIRQ(TIMER1_IRQn);    //Enable interrupt
void Timer1_Handler (void)      //Interrupt Handler
{
TMR1REG = -30000;               //Load counting value (-30000 = 0xFFFF8AD0)

//From this point the function goes to process other tasks required for this timer.

}
```

10 Watch Dog Timer

The MCU implements a WDT (Watch Dog Timer) that can operate in one of two ways:

- Interrupt Mode: In the event of a WDT rollover an interrupt will be generated.
- Reset Mode: In the event of a WDT rollover the microcontroller will reset.

The WDT supports Reset, Enable, status/flag and clear functions. It integrates a pre-scaler that can divide the system clock by 213, 219, 222 or 232. It means that the WDT internal counter will count from 0 to the pre-scaler value at the system clock speed and trigger if not cleared. For instance, a system running from a 30MHz system clock and 222 pre-scaler value will trigger the WDT after approximately 0.14 seconds if not cleared properly and in time by the application.

11 Block Transfer Engine

11.1 Introduction

Due to Verne’s serial interface, there is a large overhead for individual data transfers between the ASIC die and MCU. For most MCU application transfers this is acceptable, but a block transfer mode has been implemented for those cases where this is not (e.g. Ethernet or USB).

MCU configures:

- base address
- number of bytes
- whether to increment address (or access same address repeatedly)
- read/write

MCU then writes a start indication to initiate the transfer.

- Special format message for reduced overhead for this transfer

When block transfer happens, stall CPU only if

- It is going to access serial bus or SRAM interface
- The block transfer read data not yet received

11.2 Interrupt

An interrupt is generated at the end of each transferred block. This can be used to allow the software do things like scatter gather by software. Otherwise, the software can poll the start bit.

11.3 Usage Example

```
// read block of data from ASIC to SRAM
BTE_SFRS->BTE_CTRL.BXADD    = (uint16_t) ADDR_IN_ASIC;    // 16-bit LSB only
BTE_SFRS->BXSRAMADDR        = (uint16_t) buffer_in_sram;  // 16-bit LSB only
BTE_SFRS->BTE_CTRL.BXNUM    = SIZE_OF_ARRAY;              // number of 32-bit words to
transfer

BTE_SFRS->BTE_CTRL.TX_DIR   = BXCONF_READ;      // read => ASIC->SRAM
BTE_SFRS->BTE_CTRL.INC_ADDR = BXCONF_INCR;      // increment ASIC die address each transfer
BTE_SFRS->BTE_CTRL.BLOCKING = BXCONF_NBLOCKING; // non-blocking => transfer pauses if
                                                // MCU wants to use the ASIC interface
BTE_SFRS->BTE_CTRL.START    = BXCONF_START;     // start bit to initiate an operation
```

12 Electrical Characteristics

over junction temperature range -40°C to 125°C and recommended supply voltage (unless otherwise noted)

Electrical Characteristics

Parameter	Test Conditions	Min	Typ	Max	Unit
MCU core supply	VDD	1.35	1.5	1.65	V
Flash Supply	VFLASH	1.35	1.5	1.65	V
IO Supply	VIO	3.0	3.3	3.6	V
System Clock				30	MHz
Active Current			60	80	uA/MHz
Power On Reset					
Threshold Voltage	DC level	0.9		1.0	V
	dv/dt based			tbd	V/s
RC Oscillator					
Frequency			1		MHz
Frequency accuracy	Post trim	-10		10	%
Flash Memory (Only)					
Sector Endurance	100K Preferred May need additional qualification	20K			Cycles
Data Retention	@25degC	100			Years

Parameter	Test Conditions	Min	Typ	Max	Unit
Active Read Current	@30MHz, 125C			3.5	mA
Active Write Current				2.5	mA
Active Erase Current				2.0	mA
Standby Current			80	150	uA
Deep Standby Current			0.05	6	uA
Read access time				25	ns
Sector Erase time		4		5	ms
Chip Erase time		20		40	ms
Byte Program time				7.5	us
Wake up time from Deep standby current				10	us

6 LIN Interfaces

iND83207 contains two integrated LIN PHY on the LIN1 and LIN2 pins, plus a LIN master controller and a LIN slave controller. These circuits enable iND83207 to connect to two different busses for low speed vehicle serial data network communication using the LIN protocol.

The LIN controllers implement the datalink layer of the LIN Protocol Specification. LIN uses a single master / multiple slave concept for the message transfer between nodes of the LIN network. The LIN controller core interfaces to the micro-controller to enable the transmission and reception of message frames. It includes a wake-up function using a dominant (low) bus pin message.

Each LIN transceiver can be configured as master or slave. Each transceiver can be connected to either the master or slave controller. By this mechanism, LIN1 and LIN2 can be configured to operate as master and slave, with the master operation on either LIN1 or LIN2.

6.1 LIN features

- 2 LIN controllers and transceivers (1 slave and 1 master with swappable controllers and configurable transceivers)
- Support for LIN specification 2.2A
- Compliant with SAE J2602 (rev J2602-1_201211).
- Backward compatibility to LIN 1.3
- Programmable data rate between 1 Kbit/s and 20 Kbit/s (for master)
- Automatic bit-rate detection (for slave)
- 8-byte data buffer
- 8-bit host controller interface
- The voltage at the LIN1/LIN2 pins can be read using the housekeeping ADC, which is useful for system diagnostics

6.2 Support for Slave Node Position Detection (SNPD)

In any system which uses identical ultrasound modules built around an iND83207 device, it must be possible to place an identical module at any position within the system and for that module to be able to work out where it sits within that system. The modules in these systems can use 4 pins or more. iND83207 includes circuits which enable this function.

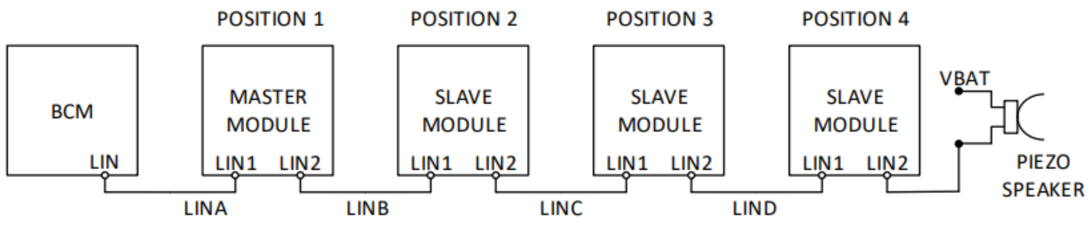
6.2.1 4-pin module topologies

A 4-pin module connects the following pins to iND83207: VBAT, GND, LIN1 and LIN2. These modules can be used in a variety of system topologies. The iND83207 LIN SNPD solution is required to support a number of different system topologies, including

- Standalone systems (no LIN to BCM)
- Single LIN busses connecting up to 12 devices
  - allowing a single LIN bus for front and rear bumpers, each with up to 6 sensors

The topologies which must be supported are detailed below.

Topology 1

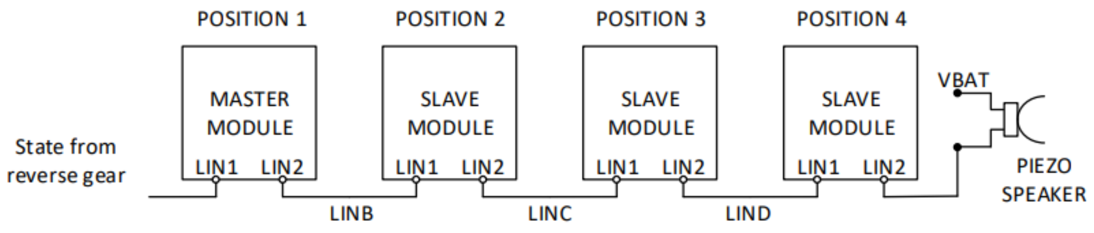


Module topology 1

Module topology 1 has the following features:

- Master module in position 1 communicates with BCM via LIN (on bus LINA)
- All bumper modules communicate over a shared local LIN bus where the master module operates as the LIN master and the slave modules all operate as LIN slaves
  - LINB, LINC, LIND operate as a single LIN bus – to be explained later
- The final slave module (position 4 in this example) may be required to drive a speaker from its unused (LIN2) module pin
- The number of modules on this bus can reach 12 maximum
  - 6 modules/bumper with both bumpers communicating on a single local LIN bus

Topology 2

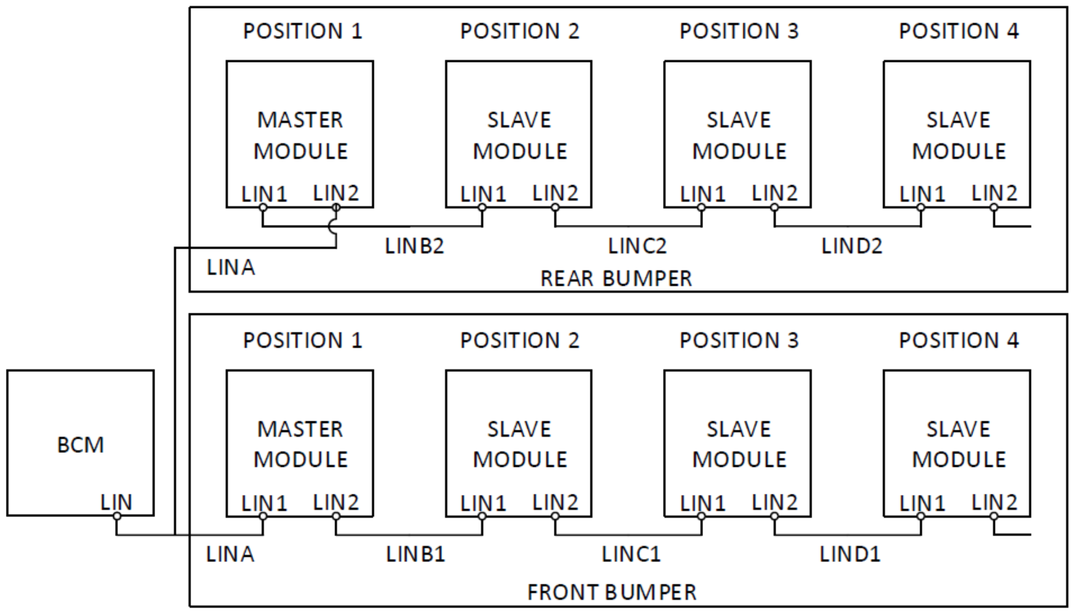


Module topology 2

Module topology 2 has the following features:

- “Standalone system”: no BCM to activate the system operation; master module in position 1 is enabled by some car event such as reverse gear engaged
  - Commonly this would be used to drive the LIN1 pin of the master module low against the LIN master or slave pull-up
- Otherwise this topology is the same as topology 1
- No communication back to a BCM is possible
  - The only driver feedback is via the speaker

Topology 3



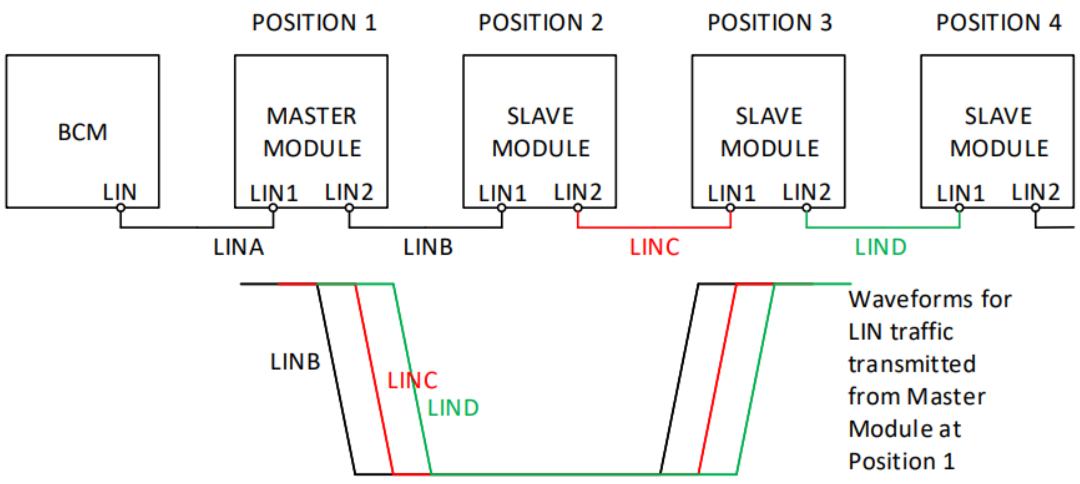
Module topology 3

Module topology 3 has the following features:

- Similar to topology 1 but with both front and rear bumpers on the same LIN bus to the BCM
- Front / rear bumper master modules identify themselves through BCM connection to LIN1 (FRONT) vs LIN2 (REAR)

6.2.2 Indie’s SNPD

- Each slave module operates as a LIN “repeater”
- Each module can
  - Drive the bus when responding to a master request
  - Drive a “dominant” state observed on LIN1 out on LIN2 (to propagate signals “to the right”)
  - Drive a “dominant” state observed on LIN2 out on LIN1 (to propagate signals “to the left”)
- Figure 7 shows a LIN packet being transmitted from the master module to all the slaves



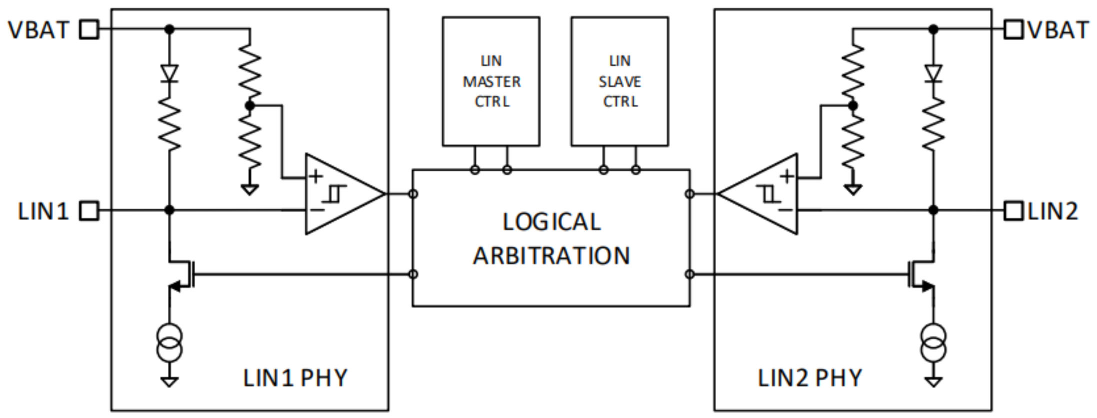
LIN traffic being transmitted from the master module

Indie’s SNPD is different from a normal LIN bus in the following ways:

- LINB, LINC, LIND are each independent LIN busses which each have only two nodes
- One of these nodes would be defined with a master pull-up, and one with a slave pull-up
  - LIN1 = slave pull-up; LIN2 = master pull-up
- For the slave modules
  - all LIN1 ports would be connected to the LIN slave controller
  - LIN1 could be driven by the slave controller

- LIN2 would “repeat” a dominant state on LIN1
- Passing messages from left to right
- LIN1 would “repeat” a dominant state on LIN2
- Passing messages from right to left

Indie’s SNPD implementation is shown in Figure 8 and operates as described below.



indie’s SNPD implementation using a repeater architecture

- Each LIN pin (LIN1/LIN2) retains the full LIN electrical interface
- LIN1 / LIN2 PHYs are multiplexed to the LIN master or LIN slave controllers
- A simple logical arbitration block is added, which performs the following tasks
    - Passes through RX and TX data to the relevant controller
    - passes a dominant state from LIN1 to LIN2, or from LIN2 to LIN1, when it is enabled to do so
    - Measures the repeater delay in each direction

The repeater delay illustrated in Figure 7 needs to be considered to ensure that it does not adversely affect the system operation in any way. There is a delay as the LIN signal is “repeated” along the sequence of independent LIN busses

- LINB -> LINC -> LIND, for a signal sent out from the master module at position 1
- This means that not all slaves receive the message from the master at the same time, which could be an issue for one of two reasons
    1. The LIN protocol timings require a slave to respond within a given time
    2. All slaves should receive the commands at the same time
        - If not, this could become an error in TOF calculations for the modules which are listening (but not transmitting)

iND83207 will ensure that the TOF error will be insignificant, and LIN specifications will be met. The design calculations and ideas for how to ensure this are given below:

- The delays will be short enough that the TOF error is insignificant
    - The delays are dependent on LIN bus capacitance, which is not known ahead of time, so the delay will not be fixed
    - We shall calculate the maximum possible delay and ensure that the LIN system operates within specification for the longest possible bus (12 modules)
- We shall also measure the propagation delay (in both directions) to enable the system to null out the delays to ensure TOF calculations are accurate to 1cm
    - We will measure the delay between the signal “received” on one LIN port, and the signal “received” at the other LIN port, of the same device
    - Apart from a very small error (due to the propagation of the LIN signal along the wiring harness) this is an accurate measure of the delay between the LIN command received at that module and the same LIN command received at the next module

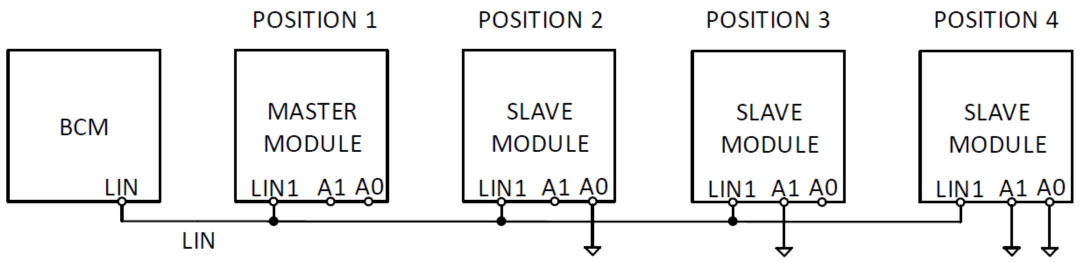
Note: if it is discovered during the design phase that the proposed repeater architecture does not work, there are other options for implementing the required SNPD functionality. The most obvious is to implement a switch that electrically connects LIN1 to LIN2. Control of this switch would be via firmware.

6.2.3 Other module types

There are other module types where SNPD may be done in a different way. These systems have additional module pins which can be used to identify the address of any module via bumper wiring.

3 LV GPIOs are provided which can be used to encode up to 8 locations via 3 binary settings. These 3 LV GPIO pins are wired to the module pins through a protection resistor. Pin pull-ups are enabled which sets the default state of that pin to “1”, which is detected when that module pin is left floating. The wiring harness can pull any of the module pins to 0V, which can be detected at the pin using the HKADC to identify a “0”.

Such a topology is shown below, with this example using 2 GPIOs to select 4 addresses.



SNPD using LV GPIOs

6.3 LIN1 / LIN2 GPIO mode

Additionally, the LIN1 and LIN2 pins can be operated as GPIOs. The outputs can be pulled up using either the master or slave pull-up. The outputs can be pulled down using the LIN pull-down driver. These operations can be directly controlled from configuration registers, bypassing the LIN controller.

6.4 LIN1 / LIN2 pins as a speaker driver

Some system topologies (especially “standalone”) need to drive a speaker. One desire is to do this in a way where every sensor module is identical. This could be done if the speaker connected to the last module on the LIN network, with the speaker connected to the module’s unused LIN1 or LIN2 pin.

Speaker types that need to be driven from the iND83207 pins are:

- “Active” speaker modules which incorporate electronics to drive the power required by the speaker
    - These require only a “logical” input which can be driven directly from the LIN1 / LIN2 pin
- Piezo speaker modules
    - These can be driven with current-limited pull-down drive

6.4.1 Electrical requirements

For driving a “logic” signal to control an active speaker, the same output structures as are available for GPIO mode should be used.

For driving a piezo speaker, the LIN1 / LIN2 pins will have a current-limited pull-down mode.

The selectable pull-down values have been listed in Electrical Specifications.

6.4.2 Drive patterns

For both GPIO and Piezo drive iND83207 will include functionality to enable different bit patterns to be driven on to the LIN1 / LIN2 pins:

- Bit-banging direct from firmware
- PWM controlled

6.5 High-speed mode for end-of-line (EOL) testing

LIN communication is limited to a maximum of 20kHz under all conditions.

A guaranteed faster baud-rate capability is required for iND83207 to support EOL testing. This will be achievable on iND83207 within certain constraints:

- Limitation of environmental conditions (supply, temperature, wiring parasitic capacitance and resistance, perhaps external pull-up resistors)
- Limitation on protocol capability
  - Full LIN protocol may not be supported; shorter messages using the standard UART will enable faster baud rates

7 GPIOs

7.1 Low-voltage GPIOs, PA[4:0]

iND83207 provides five general-purpose I/O pins on PA[4:0]. These pins are supplied from the VDD3P3 voltage rail and provide the following features

- strong push/pull output drive
- pin pull-up and pull-down resistors which can be independently enabled
- input logic detection with pin state readback to the microcontroller
- analog input to ADC
- maskable interrupt on pin state change either level-based or edge-triggered

7.2 High-voltage GPIOs, PB[2:0]

iND83207 provides three general-purpose high-voltage I/O pins on PB[2:0]. These pins are supplied from the VBAT voltage rail and provide the following features

- strong push/pull output drive
- pin pull-up and pull-down weak currents
- input logic detection with pin state readback to the microcontroller
- analog input to ADC
- maskable interrupt on pin state change either level-based or edge-triggered

7.2.1 High voltage GPIOs as module input pins

These pins can be used as module input pins. There is a requirement for a series resistor in the region of 100kΩ to protect this pin from load-dump testing. It is also anticipated that a capacitor of at least 47nF is placed at the module pin.

7.2.2 High voltage GPIOs as module output pins

These pins can be connected up to be used as module input pins. The pins provide a controlled current source which can be used to drive external ground-referenced loads.

Short-circuit protection can be implemented by firmware using the housekeeping ADC to read the voltage at the device pin.

8 Clock sources

iND83207 provides three integrated clock sources.

There is an auxiliary oscillator running at 10 kHz. This oscillator is always running and is used for system power-up and power-down control. The system clock is normally provided from a second oscillator running at 16MHz. In addition, there is a high-accuracy reference clock for timing measurement.

It is possible for firmware to disable the 16MHz oscillator, in which case the system will revert to running from the auxiliary oscillator.

8.1 Correction for clock inaccuracy using LIN SYNC

The iND83207 system clock is trimmed at production test to the 16MHz target frequency, but this frequency may drift over temperature.

iND83207 provides the facility to use the LIN SYNC field as a clock reference. This allows for an accurate measurement of the system clock rate to be made relative to the SYNC reference frequency. Using this measurement, any drift in the system clock rate can be compensated for in the time-of-flight calculations. A single register can be read which gives a value which can be used to scale the oscillator frequency.

8.2 Integrated high-accuracy calibrated reference oscillator

iND83207 integrates a high-accuracy oscillator which is calibrated at final production test. In the case of systems which have no accurate LIN SYNC field to measure as a clock reference, this integrated clock can be used as an alternative.

A frequency counter function is provided, which enables the frequency of the system clock (nominally 16MHz) to be measured with respect to the high-accuracy reference clock. This measurement can be used, as with the LIN SYNC field measurement, to scale any calculations which use the system clock as their reference.

9 Power management unit

iND83207 implements a power management unit which derives all of the supplies required by iND83207 from the VBAT pin, which is directly connected to the vehicle battery.

iND83207 requires a 3.3V supply for core analog and digital functions, plus a 1.5V supply for the core logic of the microcontroller. These supplies require external decoupling capacitors on the VDD3P3 and VDD1P5 pins.

iND83207 is safe to load-dump transient voltage spikes.

9.1 Supply monitoring

iND83207 incorporates a power-on reset circuit which ensures that the system resets into the same state whenever the battery voltage is connected for the first time. In addition, while in the active state, iND83207 monitors both generated supplies with brown-out reset (BOR) circuits. When these circuits indicate a failing supply, the system can either be fully reset immediately, or an interrupt can be generated which the MCU can use to implement a controlled power-down sequence.

9.2 Watchdog timers

iND83207 incorporates two watchdog timers, one on the analog ASIC and one on the micro-controller itself. When either of these timers expire, they can be used to reset the iND83207 device.

10 Housekeeping ADC

An 8-bit SAR ADC provided on iND83207 serves the purpose of monitoring voltages at many of the device pins, plus junction temperature. An on-board adjustable ADC reference is provided, but this is limited to approximately 2.6V maximum. Any pin voltages which may be above this level are divided down to place them within the ADC input range.

Note that the housekeeping ADC must not be used while the ultrasound receiver (USRX) is in operation.

The full list of ADC channels is

- Supplies: VBAT, VDD3P3, VDD1P5
- PA[4:0]
- PB[2:0]
- LIN1 / LIN2
- VPTAT – voltage proportional to absolute temperature of the silicon

11 Clock Calibration

11.1 CCAL Overview

The Clock Calibration (CCAL) module provides a frequency counter function which enables the frequency of a system clock to be measured with respect to an accurate reference clock. This measurement can then be used to calibrate the system clock or scale calculations which use the system clock as their reference.

11.2 CCAL Operation

CCAL performs its frequency counter function through a reference counter, running off the reference clock, and a calibration counter, running off the system clock.

Shown in Figure 11.1, the initial value of the reference counter is set through the INIT_REF_VAL field on the CCAL register bank. When run is set high by writing to the RUN field this initial reference counter value is decremented by 1 at each positive edge of the strobe reference clock. The calibration counter is initialised to 0 and then incremented by 1 at each positive edge of the system clock.

Beginning of frequency counter operation

When the reference counter is decremented to zero the calibration counter is stopped on the next positive edge of the reference clock, Figure 11.2.

End of frequency counter operation

This final value of the calibration count should be equal to,

$$\frac{clk\_sysfrequency \times (INIT\_REF\_VAL + 1)}{ref\_clkfrequency} - 1.$$

The clk_sys frequency can then be determined from this final calibration counter value to a resolution of,

frac{ref_clk frequency}{clk_sys frequency times (INIT_REF_VAL + 1)}.

Thus, the maximum achievable resolution achieved by the CCAL module is dictated by the width of the ref counter REF_CNT_W. The maximum achievable resolution being,

$$\frac{ref\_clkfrequency}{clk\_sysfrequency \times 2^{REF\_CNT\_W}}.$$

This assumes that the calibration counter width, CAL_CNT_W, is large enough so that,

$$2^{CAL\_CNT\_W} > \frac{ref\_clkfrequency}{clk\_sysfrequency \times 2^{REF\_CNT\_W}}$$

if not, the calibration counter will overflow.

Note it is possible, if the number of overflows that will occur is known, to determine the correct counter value through,

$$2^{CAL\_CNT\_W} \times no.overflows + CAL\_CNTR\_VAL$$

Example. For a clk_sys frequency of ~16MHz, ref_clk frequency of 1MHz and reference counter width of 7bits the CCAL can determine the frequency of the clk_sys to a resolution of,

$$\frac{1MHz}{16MHz \times (2^7)} = 0.000488 = 0.0488\%$$

11.3 CCAL Configuration

To configure the CCAL module for the frequency counting operation the debug key is enabled and the INIT_REF_VAL is set to an initial value.

The CCAL interrupt can also be enabled. The interrupt will be generated at the completion of the frequency counter operation, on the first positive edge of the strobe reference clock while the reference counter value is 0, shown in Figure 11.2.

The frequency counting operation is initialised by writing a 1 to the RUN field. Once RUN is high the reference counter value will be decremented on every positive edge of the strb_ref_clk. The calibration counter is incremented on the positive edge of the clk_sys at the same time as the reference counter is being decremented.

On the first positive edge of the strb_ref_clk while the reference counter is equal to zero the RUN field is cleared and the CCAL interrupt is generated. The frequency of the sys_clk can then be determined from the value of the CALCNTR field.

An example of the initial value being set and the frequency counter operation being run is shown below.

```
SYSCTRLA_SFRS->DEBUG_ACCESS_KEY = 0x1DACCE55;

CCAL_SFRS->IRQ.ENABLE.CCAL      = 1;

CCAL_SFRS->CCAL.REFOSCENA  = 0x1;
CCAL_SFRS->CCAL.INITREFVAL = 0x3F;
CCAL_SFRS->CCAL.RUN        = 0x1;

int retries = 100000;

while(retries--) {
  if(CCAL_SFRS->IRQ.IRQ.CCAL) {
    CCAL_SFRS->IRQ.CLEAR.CCAL = 1;
    return EXIT_SUCCESS;
  }
}
```

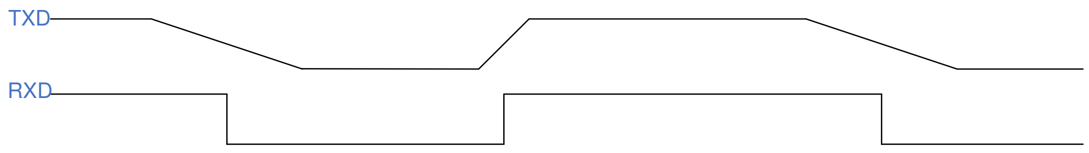
11.4 CCAL Registers

Refer to the ASIC register specification documentation included separately.

12 LIN Transmit Delay Compensation

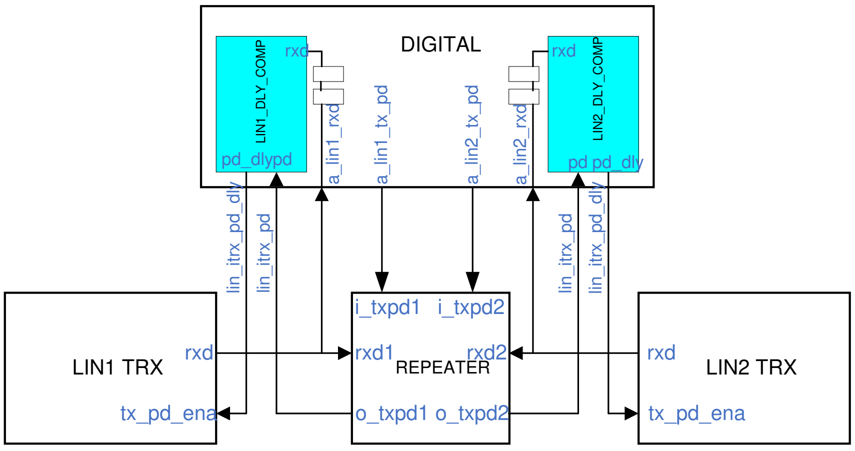
12.1 Overview

The LIN transceiver(itrx_lin) can have unequal delays for rising and falling edges, depending on the load on LIN Bus. As this difference gets significant, it starts to eat up the bit period. In repeater mode these can add up quickly over multiple modules and may end up violating the margin on the minimum bit period. In order to avoid this situation, the rise/fall delays needs to be compensated. For example if the fall delay is more than the rise delay then “0” bit will be shorter and “1” will be longer than the set bit period. In this case we need to delay the application of “1” on the bus to match the difference of fall and rise delays.



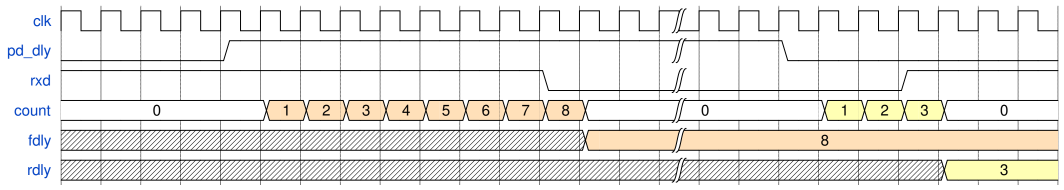
12.2 BLOCK Diagram

The pull down signal at the input of the lin transceiver has been brought in as an input and a delay compensated version of this signal is provided back to the transceiver, for the measurement purpose the retimed rxd signal is used in the lin_dly_comp block. lin_dly_comp is the module responsible for doing the delay compensation. There are 2 instance of this in ioctrla for the 2 LIN transceivers.



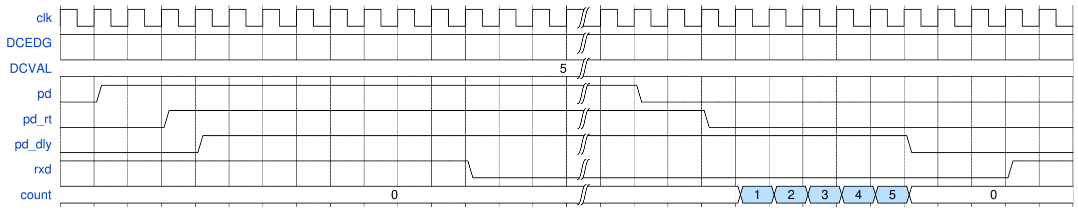
12.3 Measurement

The rise and fall delays are measured interms of number of system clocks. For Rise Delay(RDLY), the delay counter measure the delay from pd_dly signal being de-asserted to rxd going high. For Fall Delay(FDLY), the delay counter measure the delay from pd_dly signal being asserted to rxd going low.



12.4 Compensation

The compensation is done based on register settings DCEDG and DCVAL. DCEDG sets the edge to be delayed for compensation and DCVAL decides the number of clock cycles it needs to be delayed by. following is an example of compensation of the rise edge, which goes along the example in measurement section.



12.5 Modes of Operations

lin_dly_comp block looks for a falling edge of data(rising edge of pd) first, followed by a rising edge of data(falling edge of pd) to complete a cycle. A cycle may consist of only measurement, only compensation or both depending on the configuration. .. table:: functional configurations

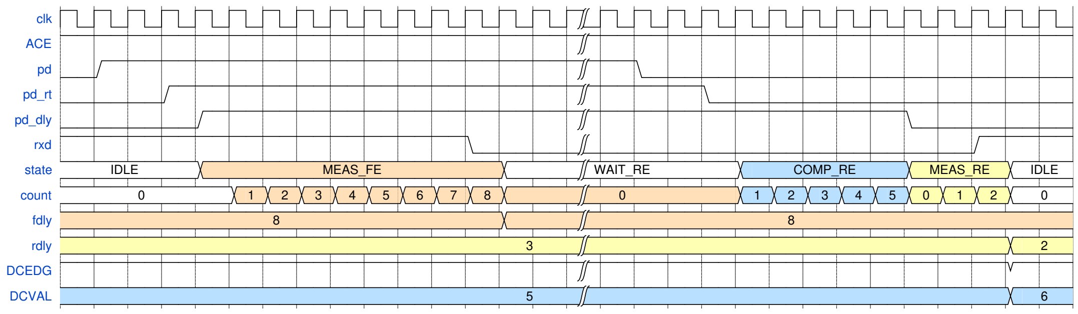
BYP	ACE	DME	DCE	Functionality
0	0	0	0	Don't use
1	x	x	x	bypass mode
0	1	x	x	auto compensation mode
0	0	1	0	Measurement only mode
0	0	0	1	Compensation only mode
0	0	1	1	Measurement and Compensation mode

12.5.1 Bypass Mode

In this mode the pd signal is routed straight to pd_dly, without any synchronous delay. Setting this mode, means the legacy design without any involvement of lin delay compensation block.

12.5.2 Auto Compensation Mode

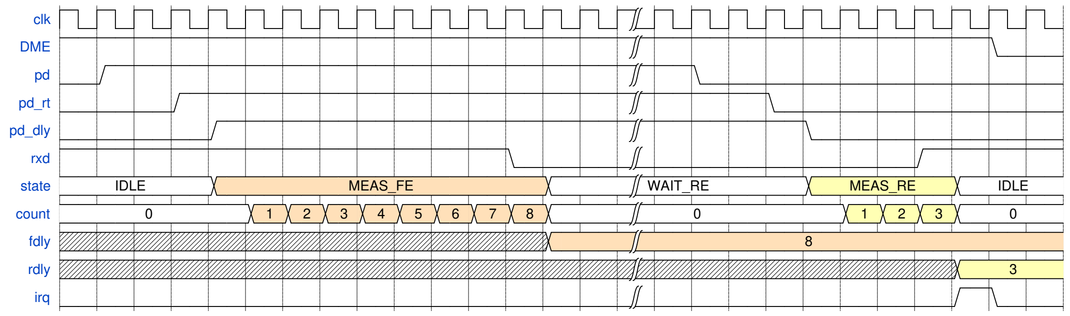
In this mode the lin_dly_comp block measure the rise and fall delays and uses them to compensate the delay in next cycle. Every cycle involves delay compensation of a particular edge depending on the previous measurement and the new delay measurement. The RDLY, FDLY, DCEDG, DCVAL gets updated by the lin_dly_comp block in every cycle.



12.5.3 Measurement Only

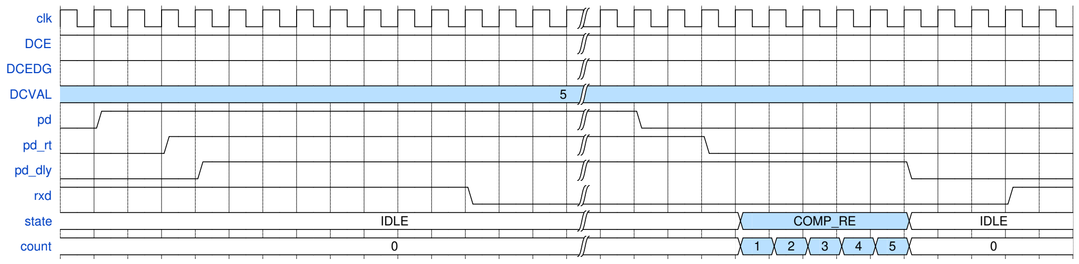
In this mode the lin_dly_comp block measure the rise and fall delays. To execute this mode as shown in table, the DME bit needs to be set, and when the lin_dly_comp block is done with the measurement of rise and fall delays, it will clear the DME, which signifies the measurement is done. At the end of measurement there is also the assertion of lindlycomp interrupt(a single cycle pulse) gets asserted. In the current implementation the interrupts for lin1_dly_comp and lin2_dly_comp are ORed and they are sharing the same vector position with timer3.

lin_dly_comp looks for the falling edge first, to do the FDLY measurement and then the rising edge to do the RDLY measurement to finish the measurement cycle. The firmware can read the FDLY and RDLY and evaluate the DCEDG and DCVAL for compensation.



12.5.4 Compensation Only

In this mode the lin_dly_comp block compensate the rise or fall edge, with the edge selection set by DCEDG and the delay value set by DCVAL. The firmware on the basis of the measurement values can set DCEDG and DCVAL, and setting the DCE enables the compensation feature. If the DCE is deasserted in the middle of a compensation cycle then the lin_dly_comp block finish the current compensation cycle and then goes IDLE.



12.5.5 Measurement and Compensation

In this mode the firmware sets both DME and DCE, It also sets the DCEDG and DCVAL to be used for compensation. The first cycle the lin_dly_comp block does the measurement and the compensation, and clears the DME bit at the end of the cycle, and the next cycle onwards it only does compensation.

12.6 Design Doccumentation

13 USTRX DSP

block diagram...

13.1 Filter Options

Following describes 7 filter options that's available. The gains are not completely normalised, so there needs to be a corresponding correction factor in the firmware.

13.1.1 Bandwidth

The -3dB filter bandwidth δf of the filters are normalised to the transmitted frequency(f_{tx}) in the table An example for calculating the bandwidth for a transmitted frequency f_{tx} = 50kHz, and if filter 4 is selected then

$$\frac{\delta f}{f_{tx}} \times f_{tx} = 0.05469 \times 50kHz = 2.73kHz.$$

13.1.2 Group Delay

The group delay in the table is in terms of number of samples at f_{tx} An example for calculating the group delay of the filter in sec for a transmitted frequency f_{tx} = 50kHz, and if filter 4 is selected then

$$gd(ms) = groupdelay(samples) \times \frac{1000}{f_{tx}} = 8.042 \times \frac{1}{50} = 0.16084ms.$$

Filter	Coeffs			Gain	δf / f_{tx}	group delay
	a1	a2	b			
1	986	949	1	0.8264	0.02051	18.875
2	996	969	1	0.8264	0.03174	13.875
3	982	942	2	0.8264	0.04150	10.375

Filter	Coeffs			Gain	δf / f	group delay
	a1	a2	b			
4	975	929	4	1.1019	0.05469	8.042
5	946	877	11	1.0101	0.10840	4.208
6	870	750	41	0.9966	0.21680	2.140
7	779	624	109	1.0009	0.38477	1.236

14 Ultrasound Receive FIFOs

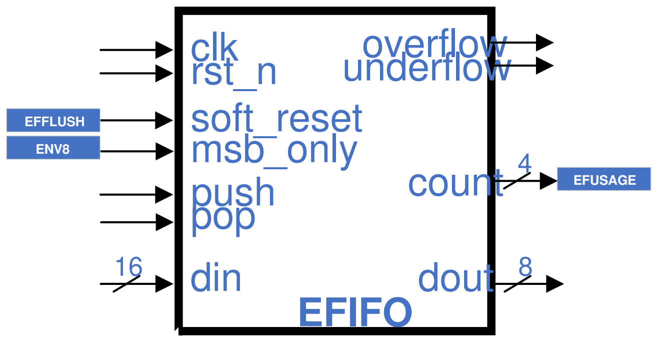
14.1 Overview

There are 2 Data FIFOs dedicated for ultra sound receiver, one for envelope data, and the other one is for frequency data of the received signal. The ustrx core pushes the envelope data(16bit samples) into the envelope FIFO(ENVFIFO), and the firmware can pop data from the ENVFIFO via ESAMPLES register either via WORD accesses or HALFWORD accesses. Similarly the ustrx core pushes frequency counter data(10bit samples) into the envelope FIFO(ENVFIFO), and the firmware can pop data from the FCFIFO via FCSAMPLES register either via WORD accesses or HALFWORD accesses. The 6 MSbits in every half word can be ignored.

NB: one thing to remeber is that the DATA received by the firmware will be in a big-endian format. So incase of a WORD access firmware need to perform a REV instruction, and incase of a HALFWORD access it has to perform a REV16 instruction.

14.2 Details

14.2.1 ENVFIFO



Size & Mode

The size of the FIFO is 8 bytes which means it can hold upto 4 samples of 16bit envelope data, or 8 samples of 8bit data when ENV8 is set. In 8bit mode where only the most significant byte of the envelope sample gets pushed to the FIFO, This mode is selected when the CSR.ENV8 register is set.

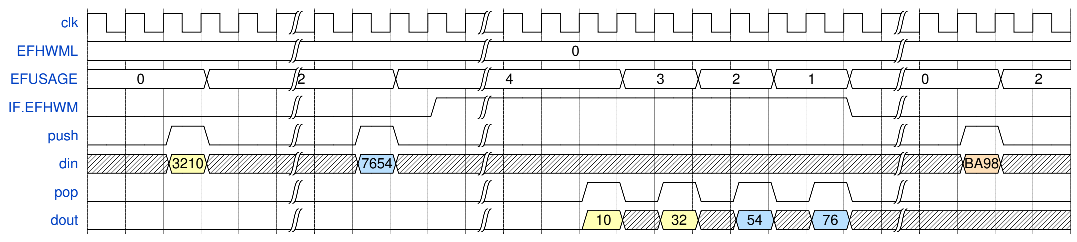
Watermark

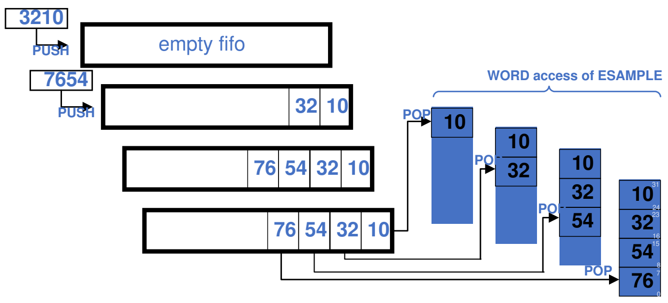
CSR.EFIFO.EFHWML can be used to set a High Watermark Level for the ENVFIFO.
CSR.IF.EFHWM is the flag to indicate the envelope FIFO is at or above the High Watermark Level
CSR.IE.EFHWM can be used to enable the High Watermark Level interrupt for ENVFIFO.

This interrupt is a non sticky interrupt, and doesn't require clear to save firmware load. As the clough interface passes an ievent for rising edge of the interrupt, so the idea is to create an interrupt which has a guaranteed edge on the interrupt everytime the fifo usage crosses the water mark. So that after reading the samples from the fifo while still executing the interrupt handler if (CSR.EFIFO.EFUSAGE >>2) > CSR.EFHWML, then mcu gets another interrupt.

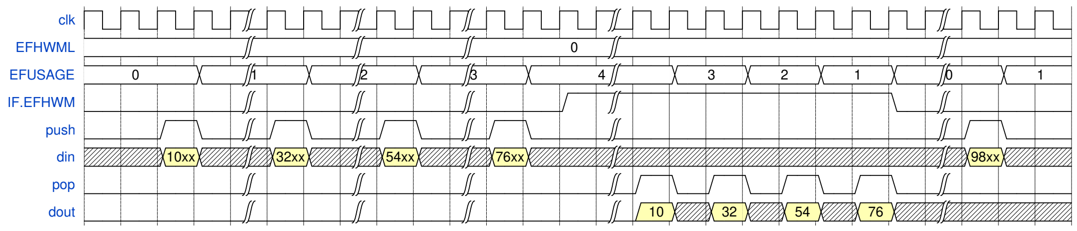
Everytime the fifo usage become equal or above the set water mark, the interrupt goes high and cleared at the last byte read. If the fifo usage is still high then the flag gets set again. it also gets cleared with CSR.EFIFO.EFFLUSH.

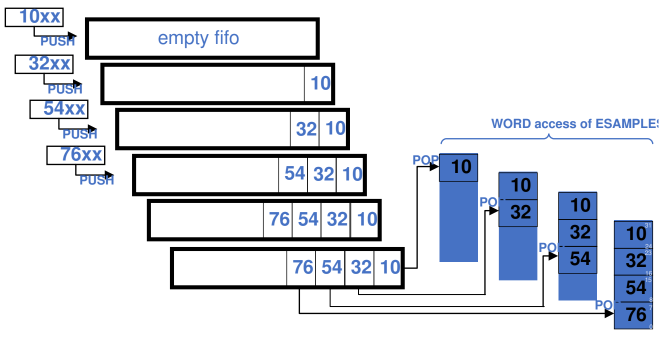
16bit mode(ENV8=0)





8bit mode(ENV8=1)

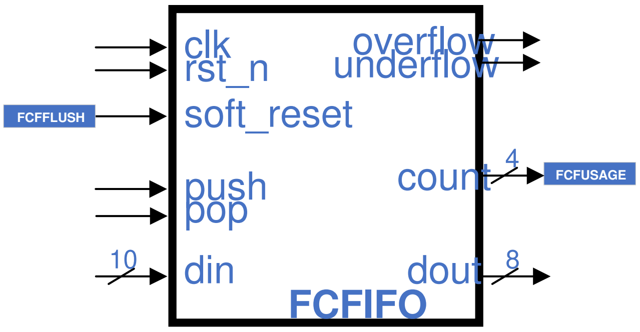




Flow Error

There are interrupts for overflow and underflow conditions of the FIFO. CSR.IF.EFOVF flag gets set in case of an overflow condition. CSR.IF.EFUNF flag gets set in case of an underflow condition.

14.2.2 FCFIFO



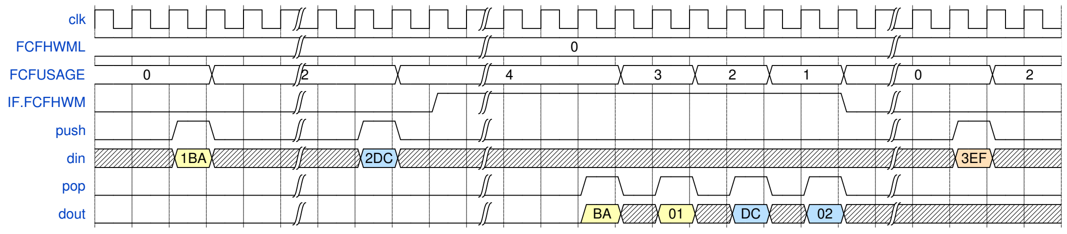
Size & Mode

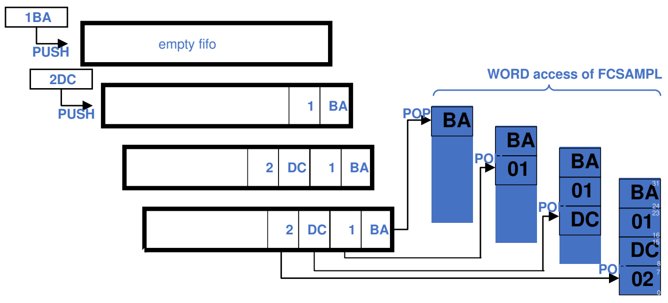
The size of the FIFO is 4x10 bits which means it can hold upto 4 samples of 10bit envelope data.

Watermark

CSR.FCFIFO.FCFHWML can be used to set a High Watermark Level for the FCFIFO. CSR.IF.FCFHWM is the flag to indicate the frequency counter FIFO is at or above the High Watermark Level CSR.IE.FCFHWM can be used to enable the High Watermark Level interrupt for FCFIFO.

This interrupt is a non sticky interrupt, and doesn't require clear to save firmware load. As the clough interface passes an ievent for rising edge of the interrupt, so the idea is to create an interrupt which has a guaranteed edge on the interrupt everytime the fifo usage crosses the water mark. So that after reading the samples from the fifo while still executing the interrupt handler if (CSR.FCFIFO.FCFUSAGE >>2) > CSR.EFHWML, then mcu gets another interrupt. Everytime the fifo usage become equal or above the set water mark, the interrupt goes high and cleared at the last byte read. If the fifo usage is still high then the flag gets set again. it also gets cleared with CSR.FCFIFO.FCFFLUSH.





Flow Error

There are interrupts for overflow and underflow conditions of the FIFO. CSR.IF.FCFOVF flag gets set in case of an overflow condition. CSR.IF.FCFUNF flag gets set in case of an underflow condition.

15 Digital Features

15.1 Watchdog timers

iND83207 incorporates two watchdog timers, one on the analog ASIC and one on the microcontroller itself. When either of these timers expire, they can be used to reset the iND83207 device.

15.2 UART

In addition to the LIN master and slave controllers, HAN will incorporate an additional UART controller.

The UART can be multiplexed to the PA[1:0] pins to operate in full duplex mode, or can be connected to the LIN1 or LIN2 pins to operate in half-duplex mode.

15.3 PWM generators

Han will incorporate a single PWM generator. Firmware programs the frequency and duty cycle, and selects which pins each of the PWM blocks is to control. The PWM blocks can control the following pins:

- LIN1
- LIN2
- PA[4:0]
- PB[2:0]

16 Device traceability

iND83207 is implemented using two silicon die which are bonded together in the package. One die (“the micro-controller die”) implements the microcontroller subsystem (M0 + peripherals, Flash, RAM). The other die (“the iND83207 analog ASIC die”) implements all the other iND83207 functions.

When supplied to the customer, both die will be programmed with unique IDs which will allow full tracing of the manufacturing flow for those individual die.

The micro-controller die will allocate Flash memory locations to store this information. The iND83207 analog ASIC die will incorporate a one-time-programmable (OTP) memory array for this purpose. Programming of this OTP will only be possible at the time of production testing.

17 Register map

17.1 USTRX

Ultrasound TRX Registers

USTRX	**CSR**	0x50000000
	control and status register for interrupt and FIFOs IE: interrupt enable, IF: interrupt flag, EFIFO: envelope FIFO, FCFIFO: frequency counter FIFO	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A				B			C	D				E	F		G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

#	Bit(s)	Field	Description	Type	Reset
A	31:28	FCFUSAGE	*current usage of the frequency counter FIFO* number of BYTE(s) in the frequency counter FIFO	ro	n/a
B	27	FCFFLUSH	*flush the frequency counter FIFO*	wo	n/a
C	24	FCFHWML	*frequency counter FIFO high watermark level* Number of WORDS in FCFIFO which generates FCFHWM interrupt. Interrupt flag asserts when (FCFUSAGE/4) > FCFHWML. ex: FCFHWML=0, can generate FCFHWM interrupt when FCFUSAGE gets to 4	r/w	0
D	23:20	EFUSAGE	*current usage of the envelope FIFO* number of BYTE(s) in the envelope FIFO	ro	n/a
E	19	EFFLUSH	*flush the envelope FIFO*	wo	n/a
F	18	ENV8	*read envelope as 8-bit MSB only* When set, only MSByte of envelope sample is pushed into FIFO, otherwise the whole 16 bit sample gets pushed into the FIFO	r/w	0
G	16	EFHWML	*envelope FIFO high watermark level* Number of words in ENVFIFO which genererates EFHWM interrupt. Interrupt flag asserts when (EFUSAGE/4) > EFHWML. ex: EFHWML=0, can generate EFHWM interrupt when EFUSAGE gets to 4	r/w	0
H	15	TXSEQ	*transmission sequence end interrupt flag* can be read for status, write a ‘1 to clear the flag	dual	0
I	14	TXBRST	*transmission burst end interrupt flag* can be read for status, write a ‘1 to clear the flag	dual	0
J	13	FCFUNF	*frequency counter FIFO underflow interrupt flag* can be read for status, write a ‘1 to clear the flag	dual	0
K	12	EFUNF	*envelope FIFO underflow interrupt flag* can be read for status, write a ‘1 to clear the flag	dual	0
L	11	FCFOVF	*frequency counter FIFO overflow interrupt flag* can be read for status, write a ‘1 to clear the flag	dual	0
M	10	EFOVF	*envelope FIFO overflow interrupt flag* can be read for status, write a ‘1 to clear the flag	dual	0
N	9	FCFHWM	*frequency counter FIFO high water mark interrupt flag* reads 1 if frequency counter FIFO is at or above set water mark	ro	0
O	8	EFHWM	*envelope FIFO high water mark interrupt flag* reads 1 if envelope FIFO is at or above set water mark	ro	0
P	7	TXSEQ	*transmission sequence end interrupt enable*	r/w	0
Q	6	TXBRST	*transmission burst end interrupt enable*	r/w	0
R	5	FCFUNF	*frequency counter FIFO underflow interrupt enable*	r/w	0
S	4	EFUNF	*envelope FIFO underflow interrupt enable*	r/w	0
T	3	FCFOVF	*frequency counter FIFO overflow interrupt enable*	r/w	0
U	2	EFOVF	*envelope FIFO overflow interrupt enable*	r/w	0
V	1	FCFHWM	*frequency counter FIFO high water mark interrupt enable*	r/w	0
W	0	EFHWM	*envelope FIFO high water mark interrupt enable*	r/w	0

USTRX	**TXCTRL**	0x50000004
	transmit controls	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	B			C	D	E	F	G					H				I			J			K			L					

#	Bit(s)	Field	Description	Type	Reset
A	31	LRSWAP	*swap the left and right drive signals*	r/w	0
B	30:28	TX2PD	*TX2 pulldown select* This setting is irrelevant during transmission. TX2PD<2> enables a 25 Ω pull-down. TX2PD<1> enables a 50 Ω pull-down. TX2PD<0> enables a 100 Ω pull-down	r/w	0
C	27	FCENA	*frequency counter enable* Enable for receive period detector	r/w	0
D	26	RXENA	*Envelope receiver enable* Enable for evelope receiver enable	r/w	0
E	25	TXDMY	*Dummy transmit burst flag* If set, transmit burst does not activate drivers	r/w	0
F	24	TXENA	*Transmit enable* Drive burst transmit enable	r/w	0
G	23:19	CLK2	*Increment to frequency division* Increment to frequency division from system clock to ultrasound frequency	r/w	0
H	18:16	CLK1	*Base frequency division* Base frequency division from system clock to ultrasound frequency	r/w	0
I	14:12	SQ3	*Number of resistive drive cycles* Number of resistive drive cycles after last drivelet	r/w	0
J	11:9	SQ2	*Number of table antiphase drive cycles* Number of squelch drivelets to use after antiphase drive	r/w	0
K	8:6	SQ1	*Number of antiphase drive cycles* Number of antiphase drive cycles to execute after burst	r/w	0
L	5:0	TXCYCLES	*Number of drive cycles* Number of drive cycles, 0 - continuous drive. Maximum allowed value is ‘d32	r/w	0

USTRX	**ENVFIFO**	0x50000008
	Envelope detector FIFO	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	ESAMPLES	*Envelope FIFO read Register* envelope FIFO can be read via this register. Received data will be ordered as big-endian. Word(32bit) access is highly recommended for speed. Halfword(16bit) access is also allowed. single Byte access can be misleading due to endianness and not recommended. REV/REV16 instructions can be used on received data to alter the endianness. independent of access type, always keep the base address for reading from the register same i.e the Byte address of the lowest byte	ro	n/a

USTRX	**FCFIFO**	0x5000000C
	frequency counter FIFO	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	FCSAMPLES	*frequency counter FIFO read Register* frequency counter FIFO can be read via this register. Received data will be ordered as big-endian. Word(32bit) access is highly recommended for speed. Halfword(16bit) access is also allowed. single Byte access can be misleading due to endianness and not recommended. REV/REV16 instructions can be used on received data to alter the endianness. independent of access type, always keep the base address for reading from the register same i.e the Byte address of the lowest byte	ro	n/a

USTRX	**CFG**	0x50000010
	Various configs	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A			B			C						D		E		F				G				H			I				

#	Bit(s)	Field	Description	Type	Reset
A	31:29	ENVDV	*Post-rectifier decimation value* Post-rectifier decimation value. Decimation factor is (ENVDV+1). This decimation have a gain factor (ENVDV+1)/2^(ceil(log2(ENVDV+1)))	r/w	0
B	28:27	IQDV	*Pre-rectifier decimation value* Pre-rectifier decimation value. Decimation factor is 2^(IQDV+1)	r/w	0
C	25:24	FCDBNC	*Frequency counter edge debounce* Period detector debouncing control 0x**0** — feature disabled 0x**1** — 4 cycles of system clock 0x**2** — 16 cycles of system clock 0x**3** — 64 cycles of system clock	r/w	0
D	19:18	AUTOFC	*Automaic frequency counter enable* Configuration for automatically enabling the period detector from transmit sequence	r/w	0
E	17:16	AUTORX	*Automaic receiver enable* Configuration for automatically enabling the envelope receiver from transmit sequence	r/w	0
F	15	RXBIAS	*Receiver bias enable* Receiver bias enable	r/w	0
G	11:8	SQ3STR	*Resistive drive strength* Drive strength during post-burst resistive damping	r/w	0
H	7:5	DTIME	*Driver dead time control* Driver dead time control	r/w	0
I	4:0	STR	*Transmit drive strength*	r/w	0

USTRX	**RXDSP**	0x50000014
	Envelope receiver DSP config	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			A							B										C											D

#	Bit(s)	Field	Description	Type	Reset
A	28:22	B	*IIR feedforward coefficient* IIR feedforward coefficient	r/w	0
B	21:12	A2	*Second IIR feedback coefficient* Second IIR feedback coefficient	r/w	0
C	11:2	A1	*First IIR feedback coefficient* First IIR feedback coefficient	r/w	0
D	0	UPREQ	*IIR coefficient update request* Set this bit to request for an update of IIR coefficients. This gets cleared by the core once the coeffs gets absorbed by the DSP. Don't change the IIR coeffs while it's set	r/w	0

USTRX	**RXAFE**	0x50000018
	Receiver analog configuration	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		A			B					C	D			E			F		G			H		I			J		K		

#	Bit(s)	Field	Description	Type	Reset
A	29:27	RGGN	*N side input resistor trim* Sets grounded resistor value for N input	r/w	0
B	26:24	RGGP	*P side input resistor trim* Sets grounded resistor value for P input	r/w	0
C	21	STGBYP	*Stage bypass* Stage bypass for gain reduction	r/w	0
D	20:18	PG4AV	*PG4 gain trim* PG4 gain trim	r/w	0
E	17:15	PG3AV	*PG3 gain trim* PG3 gain trim	r/w	0
F	14:13	PG2RL	*PG2 RL trim* PG2 RL trim	r/w	0
G	12:10	PG2GM	*PG2 GM trim* PG2 GM trim	r/w	0
H	9:8	PG1RL	*PG1 RL trim* PG1 RL trim	r/w	0
I	7:5	PG1GM	*PG1 GM trim* PG1 GM trim	r/w	0
J	4:3	LNARL	*LNA RL trim* LNA RL trim	r/w	0
K	2:0	LNAGM	*LNA GM trim* LNA GM trim	r/w	0

USTRX	**DRVNWVR**	0x5000001C
	Reverse drive amplitude table	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				A																											

#	Bit(s)	Field	Description	Type	Reset
A	27:0	DRVNWV	*Reverse drive amplitude table* Reverse drive amplitude table	r/w	0

USTRX	**CPREGS**	0x50000020
	Transmit charge pump controls	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								A						B		C					D			E	F	G	H				

#	Bit(s)	Field	Description	Type	Reset
A	23	DISVREG	*disable vregcp clocking* setting this to ‘1 will stop the vregcp clock idle to ‘0. if this is not set and ENA is ‘1 then vregcp clock will be clocking at SYSCLK	r/w	0
B	17:16	CLKMODE	*charge pump clocking mode* sets the charge pump clocking mode, if ENA is ‘0 the charge pump clock will be idle at 0 0x**0** — charge pump gets the SYSCLK 0x**1** — charge pump gets the SYSCLK/2 synchronised to rising edge of SYSCLK 0x**2** — charge pump gets the SYSCLK/2 synchronised to falling edge of SYSCLK 0x**3** — no clocks to the charge pump	r/w	0
C	15	ILIM	*Enable for output current limit* Enable for output current limit	r/w	1
D	10:8	VINREG	*Trim for input to charge pump* Trim for input to charge pump	r/w	4
E	7	ACTV	*Indicates charge pump is currently switching* If high then charge pump is currently pumping charge into the reservoir cap	ro	n/a
F	6	ENA	*Transmit charge pump function enable* Transmit charge pump function enable	r/w	0
G	5	DISCHARGE	*discharge enable*	r/w	0
H	4:0	VBOOST	*Transmit supply voltage trim* Transmit supply voltage trim	r/w	6

17.2 HKADC

ADC Control

HKADC	**CONF**	0x50000030
	Configuration settings for the ADC	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A				B								C							D			E				F	G	H	I	J	

#	Bit(s)	Field	Description	Type	Reset
A	31:28	VREFH	*High reference voltage trim* Controls the high ADC reference voltage. The voltage is equal to VBUF * (VREFH / 15) where VBUF is the output of the reference buffer (see VREFGAIN)	r/w	0xF
B	27:24	VREFL	*Low reference voltage trim* Controls the low ADC reference voltage. The voltage is equal to VBUF * (VREFL / 15) where VBUF is the output of the reference buffer (see VREFGAIN)	r/w	0
C	19:16	VREFGAIN	*VREF gain trim* Controls the ADC reference buffer output voltage. The output voltage of the buffer is equal to VBG * (15 / VREFGAIN) where VBG is the bandgap voltage (1.212V)	r/w	0xF
D	12:10	SAMPCYC	*Sample cycle* The number of clock cycles to use to sample the input signal into the ADC input sampling capacitor is equal to (1+SAMP_CYC). For all on-chip sources, setting SAMP_CYC=0 gives a sampling cycle which is long enough to ensure that the input signal will have settled to significantly better than the ADC resolution. For off-chip sources which are being sampled through a GPIO pin, it may be necessary to set SAMP_CYC to a non-zero value if the source impedance is greater than about 200kOhms	r/w	0
E	9:8	CLKSEL	*ADC Clock select* Selects the clock frequency for the ADC clk 0x**0** — SYSCLK/8. For an 16MHz SYSCLK, this is 2MHz 0x**1** — SYSCLK/16. For an 16MHz SYSCLK, this is 1MHz 0x**2** — SYSCLK/32. For an 16MHz SYSCLK, this is 0.5MHz 0x**3** — SYSCLK/64. For an 16MHz SYSCLK, this is 0.25MHz	r/w	0
F	5	CALBUF	*ADC reference buffer calibration feature* 0x**0** — Normal operation 0x**1** — Change resistor tap connection from VREFLO to VADC and connect VREFLO to ground or VDD depending on GNDREF. Debug only	r/w	0
G	4	GNDOFFBUF	*ADC reference ground offset feature* 0x**0** — Bandgap and ADC has common ground 0x**1** — Difference between bandgap and ADC ground is compensated by switched cap	r/w	0
H	3	BUFENA	*Buffer enable* Set to enable the ADC reference buffer. After this bit has been set it is necessary to make a series of conversions on the ADC before the ADC reference comes within specification. Each ADC conversion issues a clock to the ADC reference buffer which is used by its auto-zero function. After each of these the reference comes closer to specification. A series of at least 10 conversions should be run before the output data should be considered valid. If the ADC reference buffer is disabled, or the ADC has been unused for a length of time, this procedure needs to be repeated	r/w	0
I	2	FULLRG	*Full range* Set this bit for ADC reference voltages based on the full range of the supply voltage (rather than VBUF - the buffered bandgap voltage)	r/w	0
J	1:0	MODE	*ADC mode select* Selects the ADC operating mode. Please use the default setting	r/w	2

HKADC	**CNTRL**	0x50000034
	ADC Data Conversion Control Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															A	B					C									D	E

#	Bit(s)	Field	Description	Type	Reset
A	16	GUARD	*GUARD for ADC input Mux Switching* if asserted disables all the ADC input muxing, This needs to be set before changing the INSEL. This sequence is enforced to avoid the shorting of channels during the INSEL change	r/w	1
B	15	OUTENABLE	*PA0 select enabled for bringing out ADC Input* Enables the GPA[0] select to bring out the ADC input selected by INSEL. Debug purpose only *— only accessible when 'debug access' is enabled*	r/w	0
C	12:8	INSEL	*ADC Input Select* Selects the input signal to be measured by the ADC 0x**0** — PA[0] 0x**1** — PA[1] 0x**2** — PA[2] 0x**3** — PA[3] 0x**4** — PA[4] 0x**5** — VBOOST, ignore for zubat 0x**6** — PB[0] 0x**7** — PB[1] 0x**8** — PB[2] 0x**9** — VDD1P5 0x**a** — VBAT 0x**b** — VPTAT 0x**c** — LIN1 0x**d** — LIN2 0x**e** — TX1, ignore for zubat 0x**f** — TX2, ignore for zubat 0x**10** — VDD3P3 *— only accessible when 'guard access' is enabled*	r/w	0x1F
D	1	CONT	*Continuous Conversion Enable* if set enables the continuous conversion mode, else it's a single conversion. this is only checked at the end of current conversion	r/w	0
E	0	CONVERT	*ADC START/STATUS Register* Set to start a conversion, gets cleared at the end of single conversion. If CONT is set then this doesn't get cleared at the end of conversion. This can be read to check the current status of ADC conversion	r/w	0

HKADC	**DATA**	0x50000038

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											A				

#	Bit(s)	Field	Description	Type	Reset
A	7:0	DATA	The result of the last ADC conversion	ro	0

HKADC	**ADCIRQ**	0x5000003C
	ADC_CTRL interrupts Contains the enable, clear, status and active flags for the ADC_CTRL interrupt sources.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							A								B								C								D

#	Bit(s)	Field	Description	Type	Reset
A	24	DATA_RDY	data ready inpterrupt active	ro	0
B	16	DATA_RDY	data ready interrupt status	ro	0
C	8	DATA_RDY	data ready interrupt clear *— cleared automatically after each write*	wo	0
D	0	DATA_RDY	data ready interrupt enable	r/w	0

17.3 TDAC

test dac control memory

TDAC	**DACCTRL**	0x50000040
	DAC Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	B			C																D											

#	Bit(s)	Field	Description	Type	Reset
A	31	ENABLE	*test DAC enable*	r/w	0
B	30	AUTOPUSH	*auto push enable*	r/w	0
C	27:16	NEXTDCODE	*Next DAC Code*	r/w	0
D	11:0	DCODE	*Current DAC Code*	dual	0

17.4 LINMB

LINM_ATHENS

LINMB	**DATABYTE03**	0x50000050
	Data Bytes 0 to 3	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A								B								C								D							

#	Bit(s)	Field	Description	Type	Reset
A	31:24	DATABYTE3	*Data Byte 3* 4th byte of the 8-byte Data Byte	r/w	0
B	23:16	DATABYTE2	*Data Byte 2* 3rd byte of the 8-byte Data Byte	r/w	0
C	15:8	DATABYTE1	*Data Byte 1* 2nd byte of the 8-byte Data Byte	r/w	0
D	7:0	DATABYTE0	*Data Byte 0* 1st byte of the 8-byte Data Byte	r/w	0

LINMB	**DATABYTE47**	0x50000054
	Data Bytes 4 to 7	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A								B								C								D							

#	Bit(s)	Field	Description	Type	Reset
A	31:24	DATABYTE7	*Data Byte 7* 8th byte of the 8-byte Data Byte	r/w	0
B	23:16	DATABYTE6	*Data Byte 6* 7th byte of the 8-byte Data Byte	r/w	0
C	15:8	DATABYTE5	*Data Byte 5* 6th byte of the 8-byte Data Byte	r/w	0
D	7:0	DATABYTE4	*Data Byte 4* 5th byte of the 8-byte Data Byte	r/w	0

LINMB	**CSR**	0x50000058
	Control Status Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A				B									C	D	E	F				G	H	I	J		K	L		M	N	O	P

#	Bit(s)	Field	Description	Type	Reset
A	31	ENHCHK	*Enhancement Check* The host controller has to set the checksum type used in the current frame by adjusting this register 0x**0** — for classic checksum 0x**1** — for enhanced checksum	r/w	0
B	27:24	LENGTH	*Data Length* The host controller has to define the length of the data field of the current LIN frame by adjusting this register. If this is loaded with the value 4'b1111 the length of the data field is decoded from Bit 5 and 4 of ID resgister	r/w	0
C	18	TIMEOUT	*Timeout Error* There are several reason that can cause a timeout error: The master detects a timeout error if it is expecting data from the bus but no slave does respond. If the slave responds to late and the frame is not finished within the maximum frame length TFRAME_MAX a timeout error will be detected too. The slave detects a timeout error if it is requesting a data acknowledge to the host controller (for selecting receive or transmit, data length and loading data), and the host controller does not set CTRL.DATA_ACK or CTRL.STOP register until the end of the reception of the first byte after the identifier. The slave detects a timeout error if it has transmitted a wakeup signal and it detects no sync field (from the master) within 150 ms. Note: The slave does not perform an exact check of the frame length TFRAME_MAX but a timeout is detected after 200 bit times, if the slave is in receive mode and there are missing data fields or a missing ID field from the master	ro	0
D	17	CHK	*Checksum Error* Checksum Error	ro	0
E	16	BITMON	*Bit Error* During Transmit: this gets set when the monitored state on the bus is different from the bit transmitted. During Receive: this gets set when detect a Byte Field Framing Error if the ninth bit after a valid start bit is dominant	ro	0
F	15	ACTIVE	*Lin Bus Active* The bit indicates whether the LIN bus is active or not. Note: For the LIN slave, this bit is set after the detection of a correct SYNC BREAK / SYNC FIELD sequence and it is reset at the end of the transmission or if the processing of the current frame is stopped by the host controller 0x**0** — no Lin bus activity 0x**1** — transmission on the LIN bus is active	ro	0
G	11	INTR	*Interupt Request* The LIN core sets the bit when it requests an interrupt to the host controller. It has the same value as the interrupt output INTR. The bit has to be reset by the host controller by setting the bit CTRL.RST_INT register	ro	0
H	10	ERROR	*Lin Error* The LIN core sets the bit if an error has been detected (compare error register). The bit has to be reset by the host controller by setting the bit CTRL.RST_ERR register	ro	0

#	Bit(s)	Field	Description	Type	Reset
I	9	`WAKEUP`	The bit is set when the LIN core is transmitting a Wakeup signal or when the LIN core has received a Wakeup signal	ro	0
J	8	`COMPLETE`	The LIN core will set the bit after a transmission has been successfully finished and it will reset it at the start of a transmission	ro	0
K	6	`SLEEP`	*Sleep Request* The bit is used by the LIN core to determine whether the LIN bus is in Sleep Mode or not. The host controller has to set the bit after sending or receiving a Sleep Mode frame or if a bus idle timeout interrupt is requested. The bit will be reset by the LIN core when a wakeup signal is detected	r/w	0
L	5	`TRANSMIT`	*Transmit Operation* The bit determines whether the current frame is a transmit frame or a receive frame for the LIN node. It has to be set by the host controller 0x**0** — receive operation 0x**1** — transmit operation	r/w	0
M	3	`RSTINT`	*Reset interrupt* The host controller has to set this bit to reset the STATUS.INTR register and the interrupt request output of the LIN core. A read access to this bit delivers always the value 0	wo	0
N	2	`RSTERR`	*Reset Error* The host controller has to set this bit to reset the error bits in status register and error register. A read access to this bit delivers always the value 0	wo	0
O	1	`WAKEUPREQ`	*WakeUp Request* The bit has to be set by the host controller to terminate the Sleep Mode of the LIN bus by sending a Wakeup signal. The bit will be reset by the LIN core	r/w	0
P	0	`STARTREQ`	*START Request* The bit has to be set by the host controller of a LIN master to start the LIN transmission after loading Identifier, data length and data buffer. The LIN core will reset the bit after the transmission is finished or an error is occurred	r/w	0

LINMB	**CONFIG**	0x5000005C
	Configuration Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										A						B		C					D								

#	Bit(s)	Field	Description	Type	Reset
A	21:16	`ID`	FRAME ID	r/w	0
B	15:14	`PRESCL`	*Prescaler* Prescaler Setting	r/w	3
C	13:9	`BTMULT`	*Bt Mult* Bit time Multiplier	r/w	0x1F
D	8:0	`BTDIV`	*Bt Div* Bit time divider [8:0]	r/w	0x1FF

17.5 LINSB

LINS_ATHENS

LINSB	DATABYTE03	0x50000060
	Data Bytes 0 to 3	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A								B								C								D							

#	Bit(s)	Field	Description	Type	Reset
A	31:24	DATABYTE3	*Data Byte 4* 4th byte of the 8-byte Data Byte	r/w	0
B	23:16	DATABYTE2	*Data Byte 3* 3rd byte of the 8-byte Data Byte	r/w	0
C	15:8	DATABYTE1	*Data Byte 2* 2nd byte of the 8-byte Data Byte	r/w	0
D	7:0	DATABYTE0	*Data Byte 1* 1st byte of the 8-byte Data Byte	r/w	0

LINSB	DATABYTE47	0x50000064
	Data Bytes 4 to 7	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A								B								C								D							

#	Bit(s)	Field	Description	Type	Reset
A	31:24	DATABYTE7	*Data Byte 7* 8th byte of the 8-byte Data Byte	r/w	0
B	23:16	DATABYTE6	*Data Byte 6* 7th byte of the 8-byte Data Byte	r/w	0
C	15:8	DATABYTE5	*Data Byte 5* 6th byte of the 8-byte Data Byte	r/w	0
D	7:0	DATABYTE4	*Data Byte 4* 5th byte of the 8-byte Data Byte	r/w	0

LINSB	CSR	0x50000068
	Control Status Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A				B								C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	

#	Bit(s)	Field	Description	Type	Reset
A	31	ENHCHK	*Enhancement Check* The host controller has to set the checksum type used in the current frame by adjusting this field 0x**0** — for classic checksum 0x**1** — for enhanced checksum	r/w	0
B	27:24	LENGTH	*Data Length* The host controller has to define the length of the data field of the current LIN frame by adjusting this field. If this is loaded with the value 4'b1111 the length of the data field is decoded from Bit 5 and 4 of ID field	r/w	0
C	19	PARITY	*Parity Error* Identifier parity error	ro	0

#	Bit(s)	Field	Description	Type	Reset
D	18	TIMEOUT	*Timeout Error* There are several reason that can cause a timeout error: The master detects a timeout error if it is expecting data from the bus but no slave does respond. If the slave responds to late and the frame is not finished within the maximum frame length TFRAME_MAX a timeout error will be detected too. The slave detects a timeout error if it is requesting a data acknowledge to the host controller (for selecting receive or transmit, data length and loading data), and the host controller does not set CTRL.DATA_ACK or CTRL.STOP register until the end of the reception of the first byte after the identifier. The slave detects a timeout error if it has transmitted a wakeup signal and it detects no sync field (from the master) within 150 ms. Note: The slave does not perform an exact check of the frame length TFRAME_MAX but a timeout is detected after 200 bit times, if the slave is in receive mode and there are missing data fields or a missing ID field from the master	ro	0
E	17	CHK	*Checksum Error* Checksum Error	ro	0
F	16	BITMON	*Bit Error* During Transmit: this gets set when the monitored state on the bus is different from the bit transmitted. During Receive: this gets set when detect a Byte Field Framing Error if the ninth bit after a valid start bit is dominant	ro	0
G	15	ACTIVE	*Lin Bus Active* The bit indicates whether the LIN bus is active or not. Note: For the LIN slave, this bit is set after the detection of a correct SYNC BREAK / SYNC FIELD sequence and it is reset at the end of the transmission or if the processing of the current frame is stopped by the host controller 0x**0** — no Lin bus activity 0x**1** — transmission on the LIN bus is active	ro	0
H	14	BUSIDLETIMEOUT	*BUS Idle Timeout* This bit is set by the LIN core if CTRL.SLEEP register is not set and no bus activity is detected for 4s. In addition, an interrupt request to the host controller is generated in that case. After that, the host controller may assume that the LIN bus is in sleep mode and it has to set CTRL.SLEEP register of the LIN core	ro	0
I	13	ABORTED	This bit is set by the LIN core slave if a transmission is aborted after the beginning of the data field due to a timeout or bit error (caused e. g. by a new sync break after missing data bytes). The bit is also set if the processing of the current frame has been stopped by setting CTRL.STOP register. The bit is cleared by the LIN core after receiving a correct SYNC BREAK / SYNC FIELD sequence	ro	0
J	12	DATAREQ	*Data Request* The LIN core slave sets the bit after receiving the Identifier and requests an interrupt to the host controller. The host controller has to decode the Identifier to decide whether the current frame is a transmit or a receive operation. It has to adjust CTRL.TRANSMIT register and to load the data length. For transmit operations it has to load the data buffer too. After that the host controller has to set CTRL.DATA_ACK register	ro	0
K	11	INTR	*Interupt Request* The LIN core sets the bit when it requests an interrupt to the host controller. It has the same value as the interrupt output INTR. The bit has to be reset by the host controller by setting the bit CTRL.RST_INT register	ro	0
L	10	ERROR	*Lin Error* The LIN core sets the bit if an error has been detected (compare error register). The bit has to be reset by the host controller by setting the bit CTRL.RST_ERR register	ro	0
M	9	WAKEUP	The bit is set when the LIN core is transmitting a Wakeup signal or when the LIN core has received a Wakeup signal	ro	0
N	8	COMPLETE	The LIN core will set the bit after a transmission has been successfully finished and it will reset it at the start of a transmission	ro	0
O	7	STOP	The host controller of the LIN slave has set this register if it handles a data request interrupt and can not make use of the frame content with the received identifier(e.g. extended identifiers). For that case the LIN slave stops the processing of the LIN communication until the next SYNC BREAK is detected. A read access to this bit delivers always the value 0	wo	0
P	6	SLEEP	*Sleep Request* The bit is used by the LIN core to determine whether the LIN bus is in Sleep Mode or not. The host controller has to set the bit after sending or receiving a Sleep Mode frame or if a bus idle timeout interrupt is requested. The bit will be reset by the LIN core when a wakeup signal is detected	r/w	0
Q	5	TRANSMIT	*Transmit Operation* The bit determines whether the current frame is a transmit frame or a receive frame for the LIN node. It has to be set by the host controller 0x**0** — receive operation 0x**1** — transmit operation	r/w	0

#	Bit(s)	Field	Description	Type	Reset
R	4	DATAACK	*Data Acknowledgement* The bit has to be set by the host controller of a LIN slave after handling a data request interrupt (compare STATUS.DATA_REQ register). The bit will be reset by the LIN core	r/w	0
S	3	RSTINT	*Reset interrupt* The host controller has to set this bit to reset the STATUS.INTR register and the interrupt request output of the LIN core. A read access to this bit delivers always the value 0	wo	0
T	2	RSTERR	*Reset Error* The host controller has to set this bit to reset the error bits in status register and error register. A read access to this bit delivers always the value 0	wo	0
U	1	WAKEUPREQ	*WakeUp Request* The bit has to be set by the host controller to terminate the Sleep Mode of the LIN bus by sending a Wakeup signal. The bit will be reset by the LIN core	r/w	0

LINSB	**CONFIG**	0x5000006C
	Configuration Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				A		B				C						D							E								

#	Bit(s)	Field	Description	Type	Reset
A	27:26	BUSINACTIVE	*Bus Inactivity Time*	r/w	0
B	25:24	WUPREPEAT	*wakeup repeat time*	r/w	0
C	21:16	ID	Frame ID	r/w	0
D	15:14	PRESCL	*Prescaler* Prescaler Setting	r/w	3
E	8:0	BTDIV	*Bt Div* Bit time divider	r/w	0x1FF

LINSB	**SYNCCOUNT**	0x50000070
	sync counter Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														A																	

#	Bit(s)	Field	Description	Type	Reset
A	17:0	SYNCCOUNT	*sync counter* number of clk_sys per sync pattern = sync count * (2^(PRESCL+1))	ro	0x3FFFF

17.6 BTE

BTE	**BTE_CTRL**	0x50000080
	BTE Control Register This register can only be written if there is no ongoing transfer. If the BTE is transferring data, any writes to this register will be ignore	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				A	B	C	D	E								F															

#	Bit(s)	Field	Description	Type	Reset
A	27	START	An operation is initiated when this bit is set. The bit auto-clears when the block is complete	r/w	0
B	26	BLOCKING	If set then the block transfer has priority over the MCU. If the MCU tries to use the bus, it will stall until the block transfer is complete. If not set, then the MCU waits only until the remainder of the current word completes and then waits until the bus is idle again before continuing	r/w	0
C	25	TX_DIR	Transfer direction. If set then SRAM->ASIC otherwise ASIC->SRAM	r/w	0
D	24	INC_ADDR	if set then ASIC die address increments at the end of each transfer. Set to zero if the peripheral is a FIFO	r/w	0
E	23:16	BXNUM	*Number of 32-bit words to transfer*	r/w	0
F	15:0	BXADD	*Address of the ASIC die (LSB)* This is the lower 16 bits of the ASIC die address. The MSBs are 0x5001	r/w	0

BTE	**BTE_SRAM_ADDR**	0x50000084
	BTE SRAM Address Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																A															

#	Bit(s)	Field	Description	Type	Reset
A	15:0	BXSRAMADDR	*Address of the SRAM (LSB)* This is the lower 16 bits of the SRAM address. The MSBs are 0x2000	r/w	0

17.7 SYSCTRLA

System control

SYSCTRLA	**NAME**	0x50010100
	ASIC name	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	NAME	*ASIC name* A read from this register will return the ASIC name	ro	n/a

SYSCTRLA	**REV**	0x50010104
	Silicon Revision	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																A															

#	Bit(s)	Field	Description	Type	Reset
A	15:0	REV	*Silicon Revision* A read from this register will return the silicon revision (e.g. A0)	ro	n/a

SYSCTRLA	**RETAIN**	0x50010108
	Retained data	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	RETAIN	*Firmware scratch register* Only reset at power-on (e.g contents retained in Hibernate mode and retained despite any hard or soft resets)	r/w	0

SYSCTRLA	**DEBUG_ACCESS_KEY**	0x5001010C
	Debug access key	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	DEBUG_ACCESS_KEY	Write the value 0x1dacce55 to this register to enable debug options. Write any other value to disable the debug options	r/w	0

SYSCTRLA	**TRIM_ACCESS_KEY**	0x50010110
	Trim access key	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	TRIM_ACCESS_KEY	Write the value 0x1d155afe to this register to enable ‘trim access’ (which allows write access to various trim settings and production test options). Write any other value to disable trim access. [31:29] is used as tx1_pd_ena	r/w	0

SYSCTRLA	**TRIMCTRL1**	0x50010114
	trim registers for bandgaps, regulators, oscillators	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A											B						C			D					E				F		

#	Bit(s)	Field	Description	Type	Reset
A	31:24	HFOSC	*16MHz RC oscillator frequency trim* *— read-only, unless 'trim access' is enabled*	r/w	0xFF
B	20:16	LFOSC	*10KHz RC oscillator frequency trim* *— read-only, unless 'trim access' is enabled*	r/w	0x1F
C	14:12	VCORE_MCU	*VCORE MCU regulator voltage output trim* *— read-only, unless 'trim access' is enabled*	r/w	2
D	11:8	VCORE_MCU_CZ	*VCORE MCU regulator compensation capacitor trim* *— read-only, unless 'trim access' is enabled*	r/w	6
E	6:4	VBGVREG	*vreg bandgap trim* Trim for the vreg bandgap *— read-only, unless 'trim access' is enabled*	r/w	2
F	2:0	VBG	*bandgap trim* Trim for the main bandgap *— read-only, unless 'trim access' is enabled*	r/w	4

SYSCTRLA	**TRIMCTRL2**	0x50010118
	trim registers for speakers, IZTC	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				A							B								C					D				E			

#	Bit(s)	Field	Description	Type	Reset
A	27:24	PGA12LPF	*trim value for PGA1&2 LPF* *— read-only, unless 'trim access' is enabled*	r/w	0
B	20:16	IZTCNEG	*trim value for IZTC neg* *— read-only, unless 'trim access' is enabled*	r/w	0x13
C	12:8	IZTCPOS	*trim value for IZTC pos* *— read-only, unless 'trim access' is enabled*	r/w	0x11
D	7:4	SPKR2	*speaker2 trim* speaker2 trim *— read-only, unless 'trim access' is enabled*	r/w	4
E	3:0	SPKR1	*speaker1 trim* speaker1 trim *— read-only, unless 'trim access' is enabled*	r/w	4

SYSCTRLA	**USTRXDFT**	0x5001011C
	ultrasound transceiver driver direct control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							A				B	C			

#	Bit(s)	Field	Description	Type	Reset
A	8	USTRXADCOUT	*Output ADC words* When selected, envelope FIFO is loaded with ADC results *— read-only, unless 'debug access' is enabled*	r/w	0
B	4	USTRXFRCENA	*Enable for direct drive control* Enable for direct drive control *— read-only, unless 'debug access' is enabled*	r/w	0
C	3:0	USTRXFRCDRV	*Drive levels to apply when transmit direct drive enabled* Drive levels to apply when transmit direct drive enabled. [3]: TX1P, [2]: TX2P, [1]: TX1N, [0]: TX2N *— read-only, unless 'debug access' is enabled*	r/w	0

SYSCTRLA	ANATEST	0x50010120
	various testmodes to test analog frontend	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																			A	B		C	D	E		F		G	H	I	

#	Bit(s)	Field	Description	Type	Reset
A	12	VBGBUF	*Buffered VBG test enable* *— read-only, unless 'debug access' is enabled*	r/w	0
B	11:10	USTXTESTENA	*kelvin sensing tx pin test bits* test bits for kelvin sensing on TX pin *— read-only, unless 'debug access' is enabled*	r/w	0
C	9	GPADCCON	*connect all GPIOs to ADCIN net* forces all the GPIOs(PAs & PBs) to connect to the ADC input. can be used for VIL/VIH testing *— read-only, unless 'debug access' is enabled*	r/w	0
D	8	REFOSC	*reference oscillator test enable* *— read-only, unless 'debug access' is enabled*	r/w	0
E	7:6	USRXTESTENA	*drives the usrx_test_ena* *— read-only, unless 'debug access' is enabled*	r/w	0
F	5:4	TXCP	*TX charge-pump test enable* TODO *— read-only, unless 'debug access' is enabled*	r/w	0
G	3	VBG	*vbg test enable* *— read-only, unless 'debug access' is enabled*	r/w	0
H	2	VBGVREG	*vbg vref test enable* *— read-only, unless 'debug access' is enabled*	r/w	0
I	1:0	IZTC	*iztc test mode selection* Selects the mode to be tested 0x**0** — Not in testmode, pos_ena and neg_ena controlled by core 0x**1** — only neg_ena set 0x**2** — only pos_ena set 0x**3** — force pos_ena and neg_ena '1 *— read-only, unless 'debug access' is enabled*	r/w	0

17.8 CRGA

Clock & reset generator

CRGA	CLOCKSRC	0x50010200
	Low frequency clock control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																						A	B							C	D

#	Bit(s)	Field	Description	Type	Reset
A	9	ACTIVESYSCLK	*active system clock* tells abouts the clock being used as system clock 0x**0** — Slow clock 10kHz 0x**1** — Fast clock 16MHz	ro	n/a
B	8	SYSCLKSEL	*System clock select* selects the clock between fast and slow system clocks 0x**0** — Slow clock (10kHz) 0x**1** — Fast clock (16MHz)	r/w	0
C	1	HFRCSTS	*Fast RC oscillator status* Will be high if the 16MHz RC oscillator is enabled	ro	0
D	0	HFRCENA	*Fast RC oscillator enable* Setting this bit enables the 16MHz RC oscillator	r/w	0

CRGA	**RESETCTRL**	0x50010204
	Reset control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							A								B				C	D		E	F				G	H		I	J

#	Bit(s)	Field	Description	Type	Reset
A	24	REQ	*Hard reset request* Set to trigger a hard reset of Brock	wo	0
B	16	REQ	*Soft reset request* Set to trigger a soft reset of Brock — *cleared automatically after each write*	wo	0
C	12	WDTA	*WDT flag clear* Set to clear the WDT flag	wo	0
D	11	BORVCORE_MCU	*BOR VCORE_MCU clear* Set to clear the 1.8V brownout detected flag	wo	0
E	9	BORVDDA	*BOR VDDA clear* Set to clear the VDDA brownout detected flag	wo	0
F	8	POR	*POR flag clear* Set to clear the POR flag	wo	0
G	4	WDTA	*Watchdog bark flag* Set by the hardware when the watchdog barks	ro	n/a
H	3	BORVCORE_MCU	*BOR VCORE_MCU flag* Set by the hardware when a brownout of the 1.5V supply is detected	ro	0
I	1	BORVDDA	*BOR VDDA flag* Set by the hardware when a brownout of the VDDA supply is detected	ro	0
J	0	POR	*Power on reset flag* Set by the hardware during power-on reset	ro	n/a

CRGA	**BORACTION**	0x50010208
	BOR action	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										A				B	

#	Bit(s)	Field	Description	Type	Reset
A	5:4	VCORE_MCU	*BOR VCORE_MCU action* Defines the consequences of brown-out condition on the VCORE_MCU supply being detected by the hardware 0x**0** — No action 0x**1** — IRQ generated 0x**2** — Hard reset generated	r/w	2
B	1:0	VDDA	*BOR VDDA action* Defines the consequences of brown-out condition on the VDDA supply being detected by the hardware 0x**0** — No action 0x**1** — IRQ generated 0x**2** — Hard reset generated	r/w	2

CRGA	**BORCONFIG**	0x5001020C
	BOR configuration	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							A					B																C			

#	Bit(s)	Field	Description	Type	Reset
A	24	BORTESTENA	*BOR test enable* *— only accessible when 'debug access' is enabled*	r/w	0
B	19:16	BORVCOREMCUTHRESH	*BOR VCORE_MCU threshold* Select the BOR threshold voltage level for the VCORE_MCU regulator 0x**0** — 1.212V 0x**1** — 1.227V 0x**2** — 1.243V 0x**3** — 1.259V 0x**4** — 1.275V 0x**5** — 1.292V 0x**6** — 1.310V 0x**7** — 1.328V 0x**8** — 1.346V 0x**9** — 1.365V 0x**a** — 1.385V 0x**b** — 1.405V 0x**c** — 1.426V 0x**d** — 1.447V 0x**e** — 1.469V 0x**f** — 1.491V	r/w	8
C	3:0	BORVDDATHRESH	*BOR VDDA threshold* Select the BOR threshold voltage level for the VDDA regulator 0x**0** — 1.61V. DVT data disagrees. TBC 0x**1** — 1.65V. DVT data disagrees. TBC 0x**2** — 1.69V. DVT data disagrees. TBC 0x**3** — 1.73V. DVT data disagrees. TBC 0x**4** — 1.77V. DVT data disagrees. TBC 0x**5** — 1.82V. DVT data disagrees. TBC 0x**6** — 1.87V. DVT data disagrees. TBC 0x**7** — 1.92V. DVT data disagrees. TBC 0x**8** — 1.98V. DVT data disagrees. TBC 0x**9** — 2.04V. DVT data disagrees. TBC 0x**a** — 2.10V. DVT data disagrees. TBC 0x**b** — 2.17V. DVT data disagrees. TBC 0x**c** — 2.24V. DVT data disagrees. TBC 0x**d** — 2.40V. DVT data disagrees. TBC 0x**e** — 2.58V. DVT data disagrees. TBC 0x**f** — 2.79V. DVT data disagrees. TBC	r/w	0xC

CRGA	**WDTACTION**	0x50010210
	WDT action	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															A

#	Bit(s)	Field	Description	Type	Reset
A	0	WDTACTION	*WDT action* Defines the action to be taken in the event of WDT bark 0x**0** — IRQ generated 0x**1** — Hard reset generated	r/w	1

17.9 PMUA

Power management unit

PMUA	**CTRL**	0x50010300
	Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													A	B	C

#	Bit(s)	Field	Description	Type	Reset
A	2	FASTSHUTDOWN	*Fast shutdown* When set, the PMU will force de-selection of the fast oscillator and disabling of the fast oscillator during power de-escalation when entering the Hibernate state. Having the fast oscillator in use and this option selected when requesting hibernate is the route to the fastest shutdown possible	r/w	1
B	1	FASTBOOT	*Fast boot* Set to enable used of the fast clock during subsequent power-up sequences (including the portion consumed by the Clough boot sequence). The default value brings the system up with the slow clock to make the initial boot and any boot after a hard reset (e.g. after a brownout) as safe as possible	r/w	1
C	0	HIBERNATE	Set to put Brock into HIBERATE mode. Before setting this bit, ensure that wake interrupt controller HOLD bit has been set (and that a corresponding Lullaby interrupt has been received)	wo	0

PMUA	**DEBUG**	0x50010304

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															A

#	Bit(s)	Field	Description	Type	Reset
A	0	IGNORE_CIFS	*Ignore QACKs* Setting a bit in this register prevents PMUA from waiting for the assertion of the corresponding 'Quiescent State Acknowlege' signal when before transitioning towards the Hibernate state	r/w	0

17.10 EVTHOLD

Event hold

EVTHOLD	**HOLD**	0x50010400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															A

#	Bit(s)	Field	Description	Type	Reset
A	0	HOLD	Set to prevent serialisation of new non-wakeup events in prepartion for hibernate mode. At the point of becoming set, a request to send the lullaby interrupt is automatically generated. The lullaby handler can then safely assert the PMUA->CTRL.HIBERNATE bit in order to put the device into hibernate mode	r/w	0

17.11 CCAL

CCAL	**CCAL**	0x50010500
	Clock Calibration	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					A											B	C														D

#	Bit(s)	Field	Description	Type	Reset
A	26:16	CALCNTR	*Calibration Counter Value* Current value of the calibration counter	ro	0
B	15	REFOSCENA	*Reference oscilator enable*	r/w	0
C	14:8	INITREFVAL	*Initial Reference Counter Value* no. of ref_clk cycles the clock frequency is counted over - 1 (i.e to count for 16 cycles, 15 is written)	dual	0
D	0	RUN	Start frequency counter operation	r/w	0

CCAL	**IRQ**	0x50010504
	CCAL interrupts Contains the enable, clear, status and active flags for the CCAL interrupt sources.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							A								B								C								D

#	Bit(s)	Field	Description	Type	Reset
A	24	CCAL	calibration completed inpterrupt active	ro	0
B	16	CCAL	calibration completed interrupt status	ro	0
C	8	CCAL	calibration completed interrupt clear — *cleared automatically after each write*	wo	0
D	0	CCAL	calibration completed interrupt enable	r/w	0

17.12 IOCTRLA

I/O control

IOCTRLA	GPAPL	0x50010600
	GPIO Port A Lower Nibble Pin Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			A	B	C		D				E	F	G	H					I	J	K	L					M	N	O	P	

#	Bit(s)	Field	Description	Type	Reset
A	28	PDENA3	*Pin 3 pulldown enable* Enable Pulldown on GPA[3]	r/w	0
B	27	PUENA3	*Pin 3 pullup enable* Enable Pullup on GPA[3]	r/w	0
C	26	RDENA3	*Pin 3 read enable* Enable Read path on GPA[3]	r/w	0
D	24	HWMODE3	*Pin 3 hardware mode* Enable Hardware mode on GPA[3] 0x**0** — PA3 controlled by GPIOA barium settings 0x**1** — PA3 used to output PWM	r/w	0
E	20	PDENA2	*Pin 2 pulldown enable* Enable Pulldown on GPA[2]	r/w	0
F	19	PUENA2	*Pin 2 pullup enable* Enable Pullup on GPA[2]	r/w	0
G	18	RDENA2	*Pin 2 read enable* Enable Read path on GPA[2]	r/w	0
H	17:16	HWMODE2	*Pin 2 hardware mode* Enable Hardware mode on GPA[2] 0x**0** — PA2 controlled by GPIOA barium settings 0x**1** — PA2 used as I2S Data out 0x**2** — PA2 used for UART RXD 0x**3** — PA2 used for test mux1 output	r/w	0
I	12	PDENA1	*Pin 1 pulldown_ enable* Enable Pulldown on GPA[1]	r/w	0
J	11	PUENA1	*Pin 1 pullup enable* Enable Pullup on GPA[1]	r/w	0
K	10	RDENA1	*Pin 1 read enable* Enable Read path on GPA[1]	r/w	0
L	9:8	HWMODE1	*Pin 1 hardware mode* Enable Hardware mode on GPA[1] 0x**0** — PA1 controlled by GPIOA barium settings 0x**1** — PA1 used as I2S SCK out 0x**2** — PA1 used for PWM 0x**3** — PA1 used for test mux0 output	r/w	0
M	4	PDENA0	*Pin 0 pulldown enable* Enable Pulldown on GPA[0]	r/w	0
N	3	PUENA0	*Pin 0 pullup enable* Enable Pullup on GPA[0]	r/w	0
O	2	RDENA0	*Pin 0 read enable* Enable Read path on GPA[0]	r/w	0
P	1:0	HWMODE0	*Pin 0 hardware mode* Enable Hardware mode on GPA[0] 0x**0** — PA0 controlled by GPIOA barium settings 0x**1** — PA0 used as I2S WS out 0x**2** — PA0 used for UART TXD 0x**3** — PA0 used for UART RXD	r/w	0

IOCTRLA	**GPAPU**	0x50010604
	GPIO Port A Upper Nibble Pins Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											A	B	C		D

#	Bit(s)	Field	Description	Type	Reset
A	4	PDENA4	*Pin 4 pulldown enable* Enable Pulldown on GPA[4]	r/w	0
B	3	PUENA4	*Pin 4 pullup enable* Enable Pullup on GPA[4]	r/w	0
C	2	RDENA4	*Pin 4 read enable* Enable Read path on GPA[4]	r/w	0
D	0	HWMODE4	*Pin 4 hardware mode* Enable Hardware mode on GPA[4] 0x**0** — PA4 controlled by GPIOA barium settings 0x**1** — PA4 used for UART TXD	r/w	0

IOCTRLA	**GPBPL**	0x50010608
	GPIO Port B Lower Nibble Pins Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								A	B		C	D			E	F	G		H	I			J	K	L		M	N			O

#	Bit(s)	Field	Description	Type	Reset
A	23	LVL2	*Pin 2 level*	r/w	0
B	22	STR2	*Pin 2 strength*	r/w	0
C	20	PDENA2	*Pin 2 pulldown enable*	r/w	0
D	19	PUENA2	*Pin 2 pullup enable*	r/w	0
E	16	HWMODE2	*Pin 0 hardware mode* Enable Hardware mode on GPB[2] 0x**0** — PB2 controlled by GPIOA barium settings 0x**1** — PB2 used to output PWM	r/w	0
F	15	LVL1	*Pin 1 level*	r/w	0
G	14	STR1	*Pin 1 strength*	r/w	0
H	12	PDENA1	*Pin 1 pulldown enable*	r/w	0
I	11	PUENA1	*Pin 1 pullup enable*	r/w	0
J	8	HWMODE1	*Pin 0 hardware mode* Enable Hardware mode on GPB[1] 0x**0** — PB1 controlled by GPIOA barium settings 0x**1** — PB1 used to output PWM	r/w	0
K	7	LVL0	*Pin 0 level*	r/w	0
L	6	STR0	*Pin 0 strength*	r/w	0
M	4	PDENA0	*Pin 0 pulldown enable*	r/w	0
N	3	PUENA0	*Pin 0 pullup enable*	r/w	0
O	0	HWMODE0	*Pin 0 hardware mode* Enable Hardware mode on GPB[0] 0x**0** — PB0 controlled by GPIOA barium settings 0x**1** — PB0 used to output PWM	r/w	0

IOCTRLA	**DEBUGENA**	0x5001060C
	Debug Mode Enables	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							A								B

#	Bit(s)	Field	Description	Type	Reset
A	8	EFUSEEXTENA	*enable driving the fuse macro externally* this enables fuse macro driven via following pins. PB0: resetn, PA0: clk1, PA1: clk2, PA2: pw *— only accessible when 'debug access' is enabled*	r/w	0
B	0	EXTCLKENA	*external clock enable* This needs to be set if PA1 needs to be used as an external clock source to replace the high frequency RC, This should only be set if GPAPL.HWMODE1 is 0, and GPAPL.RDENA1 is set *— only accessible when 'debug access' is enabled*	r/w	0

IOCTRLA	**LINMUX**	0x50010610
	LIN Bypass Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							A								B

#	Bit(s)	Field	Description	Type	Reset
A	8	RPTRENA	*LIN repeater enable* TODO	r/w	0
B	0	M1S2	*LIN Master/Slave Port Select* Default '0 keeps LIN1 port as Slave and LIN2 port as Master. if set to '1 this will assume the LIN1 port as Master and LIN2 port as Slave	r/w	0

IOCTRLA	**GPLIN**	0x50010614
	LIN Pin Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A			B	C		D		E	F	G	H	I	J	K		L			M	N		O		P	Q	R	S	T	U	V	

#	Bit(s)	Field	Description	Type	Reset
A	31:30	SPKRISEL1	*LIN2 spkr isel*	r/w	0
B	28	DIAGPD1	*LIN2 diagnostic pulldown enable*	r/w	0
C	27:26	TXIPD1	*LIN2 Transmitter pull-down strength* *— read-only, unless 'trim access' is enabled*	r/w	0
D	25:24	TXSLEW1	*LIN2 Transmitter slew select* 0x**0** — ~5 us 0x**1** — ~0.6 us 0x**2** — ~50 ns 0x**3** — ~3 ns *— read-only, unless 'trim access' is enabled*	r/w	0
E	23	TXPOL1	*LIN2 transmit polarity*	r/w	0
F	22	RXPOL1	*LIN2 receive polarity*	r/w	0
G	21	MPU1K1	*LIN2 Master Pullup Enable* LIN2 Master Pullup(1K) enable	r/w	0
H	20	SPU30K1	*LIN2 Slave Pullup Enable* LIN2 Slave Pullup(30K) enable	r/w	0
I	19	BIASENA1	*LIN2 Bias enable* enables the Bias for the LIN2 transceiver	r/w	0

#	Bit(s)	Field	Description	Type	Reset
J	18	RDENA1	*LIN2 receive enable* Enable receive path on LIN2, this is independent of HWMODE1, make sure this is set when expecting input data from LIN1	r/w	0
K	17:16	HWMODE1	*LIN2 hardware mode* select the HW mode on LIN2 0x**0** — LIN2 DATA/TX_ENA controlled by GPIOA barium settings 0x**1** — LIN2 used to connect the lin controller 0x**2** — LIN2 used for uart, lin2_rxd conneects to uart_rxd and uart_txd connects to lin2_txd. if HWMODE0 is set lin2_rxd conneects to uart_rxd only 0x**3** — LIN2 used to output PWM	r/w	1
L	15:14	SPKRISEL0	*LIN1 spkr isel*	r/w	0
M	12	DIAGPD0	*LIN1 diagnostic pulldown enable*	r/w	0
N	11:10	TXIPD0	*LIN1 Transmitter pull-down strength* *— read-only, unless 'trim access' is enabled*	r/w	0
O	9:8	TXSLEW0	*LIN1 Transmitter slew select* 0x**0** — ~5 us 0x**1** — ~0.6 us 0x**2** — ~50 ns 0x**3** — ~3 ns *— read-only, unless 'trim access' is enabled*	r/w	0
P	7	TXPOL0	*LIN1 transmit polarity*	r/w	0
Q	6	RXPOL0	*LIN1 receive polarity*	r/w	0
R	5	MPU1K0	*LIN1 Master Pullup Enable* LIN1 Master Pullup(1K) enable	r/w	0
S	4	SPU30K0	*LIN1 Slave Pullup Enable* LIN1 Slave Pullup(30K) enable	r/w	0
T	3	BIASENA0	*LIN1 Bias enable* enables the Bias for the LIN1 transceiver	r/w	0
U	2	RDENA0	*LIN1 receive enable* Enable receive path on LIN1, this is independent of HWMODE0, make sure this is set when expecting input data from LIN1, for example when HWMODE0 is set to 2, still you need to set this bit to enable the receive path	r/w	0
V	1:0	HWMODE0	*LIN1 hardware mode* select the HW mode on LIN1 0x**0** — LIN1 DATA/TX_ENA controlled by GPIOA barium settings 0x**1** — LIN1 used to connect the lin controller 0x**2** — LIN1 used for uart, lin1_rxd conneects to uart_rxd and uart_txd connects to lin1_txd. if HWMODE1 is set uart_txd connects to lin1_txd only 0x**3** — LIN1 used to output PWM	r/w	1

IOCTRLA	LIN1DLYCOMP	0x50010618
	LIN1 Delay compensation control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	B								C								D											E	F	G	H

#	Bit(s)	Field	Description	Type	Reset
A	31	DCEDG	*LIN1 delay compensation edge*	dual	0
B	30:24	DCVAL	*LIN1 delay compensation value*	dual	0
C	22:16	DMF	*LIN1 measured fall edge delay*	ro	0
D	14:8	DMR	*LIN1 measured rise edge delay*	ro	0
E	3	DCE	*Enable LIN1 Delay compensation*	r/w	0
F	2	DME	*Enable LIN1 Delay Measurement*	r/w	0
G	1	ACE	*Enable LIN1 AUTO Delay compensation* measures delays and updates the comp registers. ignores the DME and DCE settings. updates DCEDG and DCVAL automatically after every measurement	r/w	0
H	0	BYP	*Enable bypass* pass through the drive without any synchronous delay	r/w	0

IOCTRLA	LIN2DLYCOMP	0x5001061C
	LIN1 Delay compensation control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	B								C								D											E	F	G	H

#	Bit(s)	Field	Description	Type	Reset
A	31	DCEDG	*LIN2 delay compensation edge*	dual	0
B	30:24	DCVAL	*LIN2 delay compensation value*	dual	0
C	22:16	DMF	*LIN2 measured fall edge delay*	ro	0
D	14:8	DMR	*LIN2 measured rise edge delay*	ro	0
E	3	DCE	*Enable LIN2 Delay compensation*	r/w	0
F	2	DME	*Enable LIN2 Delay Measurement*	r/w	0
G	1	ACE	*Enable LIN2 AUTO Delay compensation* measures delays and updates the comp registers. ignores the DME and DCE settings. updates DCEDG and DCVAL automatically after every measurement	r/w	0
H	0	BYP	*Enable bypass* pass through the drive without any synchronous delay	r/w	0

IOCTRLA	**LINTMODE**	0x50010620
	LIN Test Mode Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																			A				B				C				D

#	Bit(s)	Field	Description	Type	Reset
A	12	GPACON1	*LIN2 test via GPA* If set to ‘1 it connects the lin2_txd to PA[0] and lin2_rxd to PA[1]. This only works if GPAPL.HWMOD0 and GPAPL.HWMOD1 is set to 0. GPLIN.HWMODE1 should be set to ‘0, The LIN2 receive path needs to be set via GPLIN.RDENA1, The rest of the port directions gets automatically set *— only accessible when ‘debug access’ is enabled*	r/w	0
B	8	DIGINTEST1	*Enable LIN2 Digital Test Input* If set to ‘1 it enables the digital test input path. GPLIN.BIASENA1 needs to be set to allow this debug feature *— only accessible when ‘debug access’ is enabled*	r/w	0
C	4	GPACON0	*LIN1 test via GPA* If set to ‘1 it connects the lin1_txd to PA[0] and lin1_rxd to PA[1]. This only works if GPAPL.HWMOD0 and GPAPL.HWMOD1 is set to 0. GPLIN.HWMODE0 should be set to ‘0, The LIN1 receive path needs to be set via GPLIN.RDENA0, The rest of the port directions gets automatically set *— only accessible when ‘debug access’ is enabled*	r/w	0
D	0	DIGINTEST0	*Enable LIN1 Digital Test Input* If set to ‘1 it enables the digital test input path. GPLIN.BIASENA0 needs to be set to allow this debug feature *— only accessible when ‘debug access’ is enabled*	r/w	0

IOCTRLA	**DIGTESTMUX**	0x50010624
	Digital test mux select	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																		A								B					

#	Bit(s)	Field	Description	Type	Reset
A	13:8	TESTMUXSEL1	Selects debug signal to output on port A pin 2. Please see description of Test Mux Sel 0 *— only accessible when ‘debug access’ is enabled*	r/w	0

#	Bit(s)	Field	Description	Type	Reset
B	5:0	TESTMUXSEL0	Selects debug signal to output on port A pin 1 0x**0** — ustrx: dftmux_cmp 0x**1** — ioctrla: a_lin1_itrx_pd 0x**2** — ioctrla: a_lin2_itrx_pd 0x**3** — ioctrla: a_lin1_itrx_pd_dly 0x**4** — ioctrla: a_lin2_itrx_pd_dly 0x**5** — ioctrla: a_lin1_dig_in 0x**6** — ioctrla: a_lin2_dig_in 0x**7** — hkadc: adc_clk 0x**8** — hkadc: state[0] 0x**9** — hkadc: state[1] 0x**a** — hkadc: state[2] 0x**b** — ccal: ref_clk 0x**c** — efuse: pw 0x**d** — efuse: reset_n 0x**e** — efuse: clk1 0x**f** — efuse: clk2 0x**10** — efuse: state[0] 0x**11** — efuse: state[1] 0x**12** — efuse: state[2] 0x**13** — efuse: state[3] 0x**14** — efuse: 1’b0 0x**15** — efuse: bit_count[0] 0x**16** — efuse: bit_count[1] 0x**17** — efuse: bit_count[2] 0x**18** — efuse: bit_count[3] 0x**19** — efuse: bit_count[4] 0x**1a** — efuse: bit_count[5] 0x**1b** — efuse: end_wait_pulse 0x**1c** — linm: rxd 0x**1d** — linm: txd 0x**1e** — lins: rxd 0x**1f** — lins: txd 0x**20** — crga: clk_lf_rc 0x**21** — crga: clk_hf_rc 0x**22** — crga: clk_lf_ext (external clk only observed on GPA2) 0x**23** — crga: clk_sys_gated 0x**24** — crga: a_por_n 0x**25** — crga: bor_vdda_n 0x**26** — crga: bor_vcore_mcu_n 0x**27** — crga: wdt_bark 0x**28** — pmua: cstate[0] 0x**29** — pmua: cstate[1] 0x**2a** — pmua: cstate[2] 0x**2b** — pmua: cstate[3] 0x**2c** — pmua: counter[0] 0x**2d** — pmua: counter[1] 0x**2e** — pmua: counter[2] 0x**2f** — pmua: counter[3] 0x**30** — pmua: quacks 0x**31** — pmua: daffodil 0x**32** — pmua: snowflake — *only accessible when ‘debug access’ is enabled*	r/w	0

#	Bit(s)	Field	Description	Type	Reset

17.13 WDTA

Watchdog Timer (barium)

WDTA	**CTRL**	0x50010700
	Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							A						B		C							D								E	

#	Bit(s)	Field	Description	Type	Reset
A	24	WUPTOUT	*wakeup via timer status* A flag that indicates wakeup timeout event has happened. write a '1 to clear this field	r/w	0
B	18:17	WUPTSEL	*wakeup timeout select* Defines the wakeup timeout period (the time the chip stays in hibernate) 0x**0** — 2^9 * Strobe Period (51.2ms for 10kHz Strobe) 0x**1** — 2^{12} * Strobe Period (409.6ms for 10kHz Strobe) 0x**2** — 2^{15} * Strobe Period (3.28ms for 10kHz Strobe) 0x**3** — 2^{17} * Strobe Period (13.1ms for 10kHz Strobe)	r/w	3
C	16	WUPTENA	*enable the wakeup feature* This enables the wakeup from hibernate via timeout feature	r/w	0
D	9:8	TIMEOUTSEL	*Timeout select* Defines the watchdog timeout period (the time between a clear operation and the next timeout) 0x**0** — 2^8 * Strobe Period (25.6ms for 10kHz Strobe) 0x**1** — 2^{10} * Strobe Period (102.4ms for 10kHz Strobe) 0x**2** — 2^{13} * Strobe Period (819.2ms for 10kHz Strobe) 0x**3** — 2^{17} * Strobe Period (13.1ms for 10kHz Strobe)	r/w	3
E	1	RUNNING	*Running status* A flag that indicates when the watchdog timer is enabled 0x**0** — Watchdog timer is stopped and cleared 0x**1** — Watchdog timer is running	ro	0

WDTA	**STOP**	0x50010704

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	STOP	Write the *stop* code (0x6da475c3) to this register to reset the timer and disable the watchdog (e.g. during debug). If any other value is written to this register the watchdog will be enabled	wo	0

WDTA	**CLEAR**	0x50010708

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	CLEAR	Write the values 0x3c570001 and 0x007f4ad6 (in sequence, with no intervening accesses) to reset the watchdog timer. Periodically performing this action is the expected method of preventing the watchdog from timing out (and resetting the MCU)	wo	0

WDTA	CNTVAL	0x5001070C
	Counter value	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	CNTVAL	*Counter value* The instantaneous value of watchdog timeout counter	ro	0

17.14 EFUSE_CTRL_HELIUM

eFUSE Controller

EFUSE_CTRL_HELIUM	CONF	0x50010800
	eFUSE configure register This register defines the timing of the signals used to program and read the efuse. These values needs to be adjusted to match the efuse signal spec. PROG_WIDTH defines the PW pulse width. T1_T2 defines the hold/setup delays between CLK1 and PW. The Tclk period of both CLK1 and CLK2 is: Tclk = (PROG_WIDTH + (2 x T1_T2)).	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								A								B															

#	Bit(s)	Field	Description	Type	Reset
A	23:16	T1_T2	Define the number of system clock cycles the T1 (hold time between CLK1 and PW) delay will be. The same register also is used to define the T2 delay (setup time between PW and CLK1). This register will allow the user to change the system clock frequency and keep being able to program the eFUSE macro. The number in this field multiply by the period of the system clock MUST be greater than 1us. The reset value is set to work with the 16Mhz RC clock which is the default system clock	r/w	0x10
B	15:0	PROG_WIDTH	Define the number of system clock cycles the PROG pulse width will be. This register will allow the user to change the system clock frequency and keep being able to program the eFUSE macro. The number in this field multiply by the period of the system clock MUST be equal (or close to) to 10us. The actual window defined in the eFUSE macro is from 8us to 12us. The reset value is set to work with the 16Mhz RC clock which is the default system clock	r/w	0xA0

EFUSE_CTRL_HELIUM	CTRL	0x50010804
	Control of the opperations of the eFUSE Macro	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							A								B

#	Bit(s)	Field	Description	Type	Reset
A	8	READ	Writting a one to this bit will start the read of the eFUSE macro data into the 32-bit DATA register(s). When the read operation is done, the controller will clear that bit to let FW knows that data are available in the DATA register(s)	r/w	0
B	0	WRITE	Writting a one to this bit will start the program of the 32-bit DATA register(s) into the eFUSE macro. When the program operation is done, the controller will clear that bit to let FW knows that a new program operation can be done. The programming can only be done to set a bit to one. The programming can only be done during manufacturing. The DATA register(s) need to be written before this bit is set	r/w	0

EFUSE_CTRL_HELIUM	**DATA**	0x50010808
	set 0 of 32-bit Data of the eFUSE Macro	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	DATA	On read, this register return the value of the specific 32-bit of the eFUSE macro. On write, the value written is used to program the specific 32-bit of the eFUSE macro. The programming can only be done to set a bit to one. The programming can only be done during manufacturing. This register becomes a read only register once the chip used in the application/field	r/w	0

EFUSE_CTRL_HELIUM	**DEBUG**	0x5001080C
	eFUSE debug register This register should only be used for debug. The Quasi programming feature is not implemented in HW but it can be done by FW through this debug register.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																						A		B	C	D	E			F	

#	Bit(s)	Field	Description	Type	Reset
A	9:8	RD_THRES	set the read threshold for margin tests 0x**0** — threshold set for nominal read 0x**1** — threshold set for margin0 read 0x**2** — threshold set for margin1 read	r/w	0
B	7	PW_FW	When set, Firmware has control over the PW pin of the macro	r/w	0
C	6	CLK2_FW	When set, Firmware has control over the CLK2 pin of the macro	r/w	0
D	5	CLK1_FW	When set, Firmware has control over the CLK1 pin of the macro	r/w	0
E	4	RESETN_FW	When set, Firmware has control over the RESETN pin of the macro	r/w	0
F	1:0	DEBUG_MODE	Control the RESETN signal by Firmware when RESETN_FW_SEL is set 0x**0** — The controller FSM drives the efuse signals 0x**1** — The FW controls the efuse signals through the xxx_FW register fields 0x**2** — External pins controls the efuse signals	r/w	0

EFUSE_CTRL_HELIUM	**DEBUGDATA**	0x50010810
	Debug eFUSE data. The data are directly the outputs of the fuse macro	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	DEBUGDATA	This register is only used for Debug to have access to the output of the fuse macro	ro	0

17.15 LINWICA

LIN wakeUp interrupt controller

LINWICA	**IRQ**	0x50010900
	Wakeup interrupts Contains the the enable, clear, status and active flag for the Wakeup interrupt sources.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						A	B							C	D							E	F							G	H

#	Bit(s)	Field	Description	Type	Reset
A	25	LIN2	LIN2 wakeup interrupt active	ro	0
B	24	LIN1	LIN1 wakeup interrupt active	ro	0
C	17	LIN2	LIN2 wakeup interrupt status	ro	0
D	16	LIN1	LIN1 wakeup interrupt status	ro	0
E	9	LIN2	LIN2 wakeup interrupt clear — *cleared automatically after each write*	wo	0
F	8	LIN1	LIN1 wakeup interrupt clear — *cleared automatically after each write*	wo	0
G	1	LIN2	LIN2 wakeup interrupt enable	r/w	0
H	0	LIN1	LIN1 wakeup interrupt enable	r/w	0

17.16 PWM

Pulse width modulation waveform generator

PWM	**BASE**	0x50010A00
	Base functions	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														A	

#	Bit(s)	Field	Description	Type	Reset
A	1:0	PRESCALESEL	*Prescaler select* Defines the ratio between the system clock and the clock used for the waveform generator 0x**0** — Divide by 1 0x**1** — Divide by 2 0x**2** — Divide by 4 0x**3** — Divide by 8	r/w	0

PWM	**CTRL**	0x50010A04
	PWM control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							A								B								C								D

#	Bit(s)	Field	Description	Type	Reset
A	24	UPDATE	Set to trigger consumption of new PULSE parameters (period and pulse width). The flag is automatically cleared by the hardware when the settings are consumed, so reading a high value indicates that an update is still pending	dual	0
B	16	INVERT	Set to invert the output waveform	r/w	0
C	8	ENASTS	*Enable status* Status of enable in the waveform generator	ro	0
D	0	ENAREQ	*Enable request* Set to enable the waveform generator	r/w	0

PWM	**PULSE**	0x50010A08
	PWM pulse setup	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																B															

#	Bit(s)	Field	Description	Type	Reset
A	31:16	PERIOD	Specifies the period of the output waveform in terms of a number of prescaler output cycles	r/w	0
B	15:0	PWIDTH	*Pulse Width* Specifies the pulse width of the output waveform in terms of a number of prescaler output cycles. This control normally determines the high time of the output waverform, however, if INVERT is set it determines the low time	r/w	0

PWM	**INTEDGECTRL**	0x50010A0C
	PWM interrupt control Contains the the enable and clear for the PWM edge interrupt sources.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														A																B	

#	Bit(s)	Field	Description	Type	Reset
A	17:16	CLEAR	*Interrupt clear*	wo	0
B	1:0	ENABLE	*Interrupt enable*	r/w	0

PWM	**INTEDGESTATUS**	0x50010A10
	PWM interrupt status Contains the the status for the PWM edge interrupt sources.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														A																B	

#	Bit(s)	Field	Description	Type	Reset
A	17:16	IRQ	*Interrupt active* Each channel of PWM respresents 2 bits in edge triggered interrupt. LSB representing rising edge interrupt and MSB representing falling edge interrupt	ro	0
B	1:0	STATUS	*Interrupt status* Each channel of PWM represents 2 bits in edge status. LSB representing rising edge and MSB representing falling edge. Note: 0x2 by default since PWM always triggers the falling edge when PWM starts off disabled	ro	2

PWM	**INTUPDATED**	0x50010A14
	PWM interrupt control Contains the enable, clear, status and active for the PWM updated interrupt sources.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							A								B								C								D

#	Bit(s)	Field	Description	Type	Reset
A	24	IRQ	*Interrupt active*	ro	0
B	16	STATUS	*Interrupt status*	ro	0
C	8	CLEAR	*Interrupt clear*	wo	0
D	0	ENABLE	*Interrupt enable*	r/w	0

17.17 I2S

I2S Master Transmitter register

I2S	**CONF**	0x50010B00
	Configuration Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				A								B		C					D	E							F				G

#	Bit(s)	Field	Description	Type	Reset
A	27:24	BITSTS	*I2S bits counter* Represent the remaining bits to shift	ro	0
B	19:18	SCKHT	*SCK high time* Defines SCK high period in terms of number of system clock periods	r/w	0
C	17:16	SCKLT	*SCK low time* Defines SCK low period in terms of number of system clock periods	r/w	0
D	12	EDGESEL	*I2S shift edge select* If 0 the I2S slave is expected to capture SD on the rising edge of SCK. If 1 the I2S slave is expected to capture SD on the falling edge of SCK	r/w	0
E	11:8	WORDSIZE	*I2S data width* Select the number of bits to transmit. Ex:To transmit 16 bits set this to 15	r/w	0xF
F	4	SOURCESEL	*source select*	r/w	0
G	0	ENABLE	*I2S strobe enable* enable I2S master transmitter	r/w	0

17.18 UARTA

UART

UARTA	**DATA**	0x50010D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								A							

#	Bit(s)	Field	Description	Type	Reset
A	7:0	DATA	Used for both received data and data that is to be transmitted	dual	0

UARTA	**UARTDATARECEIVESTATUS**	0x50010D04
	Data Receive Status This register contains the receive status associated with the current byte read from the UART_DATA register. The Data Receive Status register is updated only after a read from the UART_DATA register. Example- Read UART_DATA byte first. Then read UART_DATA_RECEIVE_STATUS register second.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													A	B	C

#	Bit(s)	Field	Description	Type	Reset
A	2	BREAKERROR	*Break Error* This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity, and stop bits)	ro	0
B	1	PARITYERROR	*Parity Error* When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the UARTLCR_H register	ro	0
C	0	FRAMEERROR	*Framing Error* When this bit is set to 1, it indicates that the received byte did not have a valid stop bit (a valid stop bit is 1)	ro	0

UARTA	**MSGCTRL**	0x50010D08
	Message control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								A	B	C	D	E	F	G															H	I	J

#	Bit(s)	Field	Description	Type	Reset
A	23	LOOPENA	*Loopback enable* Set to enable loopback	r/w	0
B	22	BREAKENA	*Break enable* Set to force transmission of zero (for a break condition)	r/w	0
C	21	STICKENA	*Sticky partiy enable* Set to enable sticky parity	r/w	0
D	20	PARODD	*Odd parity* Set for odd parity (see also PARENA)	r/w	0
E	19	PARENA	*Parity enable* Set to enable parity (see PARODD for odd/even)	r/w	0
F	18	STOP	*Stop bit control* 0x**0** — One stop bit 0x**1** — If a 5-bit transmission it selects 1.5 stop bits, otherwise 2 stop bits (6, 7 & 8 bits)	r/w	0
G	17:16	SIZE	*Transmission word size* 0x**0** — 5-bit 0x**1** — 6-bit 0x**2** — 7-bit 0x**3** — 8-bit	r/w	3
H	2	UFIFOSOFTRESET	*FIFO SOFT RESET* Resets FIFO pointers to zero and initializes FIFO contents to zero — *cleared automatically after each write*	wo	0
I	1	ENABLE_STS	*Enable status* Status of UART enable	ro	0
J	0	ENABLE	Set to enable the UART	r/w	0

UARTA	**UARTINT**	0x50010D0C
	UART Interrupts Contains the the enable, status and clear for the UART interrupt sources.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		A	B	C	D	E	F			G	H	I	J	K	L			M	N	O	P	Q	R			S	T	U	V	W	X

#	Bit(s)	Field	Description	Type	Reset
A	29	TXDONE	*Transmission done Interrupt* Set by the UART when the transmission is done	ro	0
B	28	BREAKKERR	*Break Error Interrupt* Set by the UART when a break error occurs	ro	0
C	27	PRTYERR	*Parity Error Interrupt* Set by the UART when a parity error occurs	ro	0
D	26	FRMERR	*Framing error Interrupt* Set by the UART when a framing error occurs	ro	0
E	25	OVRUNERR	*Overrun error Interrupt* Set by the UART when an overrun error occurs	ro	0
F	24	RXDONE	*Rx Data ready Interrupt* Set by the UART when Rx data is ready	ro	0
G	21	TXDONE	*Transmission is done* Set by the UART when the transmit is done	ro	0
H	20	BREAKKERR	*Break IRQ* Set by the UART when a break error occurs	ro	0
I	19	PRTYERR	*Parity Error* Set by the UART when a parity error occurs	ro	0
J	18	FRMERR	*Framing error* Set by the UART when a framing error occurs	ro	0
K	17	OVRUNERR	*Overrun error* Set by the UART when an overrun error occurs	ro	0
L	16	RXDONE	*Rx Data ready* Set by the UART when Rx data is ready	ro	0
M	13	TXDONE	*Transmission done Interrupt Clear* *— cleared automatically after each write*	wo	0
N	12	BREAKKERR	*Break Error Interrupt Clear* *— cleared automatically after each write*	wo	0
O	11	PRTYERR	*Parity Error Interrupt Clear* *— cleared automatically after each write*	wo	0
P	10	FRMERR	*Framing error Interrupt Clear* *— cleared automatically after each write*	wo	0
Q	9	OVRUNERR	*Overrun error Interrupt Clear* *— cleared automatically after each write*	wo	0
R	8	RXDONE	*Rx Data ready Interrupt Clear* *— cleared automatically after each write*	wo	0
S	5	TXDONE	*Transmission done Interrupt Enable*	r/w	0
T	4	BREAKKERR	*Break Error Interrupt Enable*	r/w	0
U	3	PRTYERR	*Parity Error Interrupt Enable*	r/w	0
V	2	FRMERR	*Framing error Interrupt Enable*	r/w	0
W	1	OVRUNERR	*Overrun error Interrupt Enable*	r/w	0
X	0	RXDONE	*Rx Data ready Interrupt Enable*	r/w	0

UARTA	**UARTBAUD**	0x50010D20
	UART Baud rate The controls in this register define the relationship between the system clock and the UART baud rate. The baud rate is given by the following equation. baud_rate = Fclk/(OSR*(BAUDDIV+1)) where Fclk is the frequency of the system clock, and OSR and BAUDDIV are the fields of this register.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			A				B				C					D															

#	Bit(s)	Field	Description	Type	Reset
A	28	URETARD	*Retard Register* Retards the sample window by 1 cycle. For Debug	r/w	0
B	24	UADVANCE	*Advance Register* Advances the sample window by 1 cycle. For Debug	r/w	0
C	20:16	OSR	*Over-sampling ratio* Valid OSR Range: 6 to 16	r/w	0x10
D	15:0	BAUDDIV	*Baud rate divider*	r/w	0

17.19 GPIOA

ASIC GPIO registers

GPIOA	**GPADATA**	0x50014000
	GPIO Port A Data	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											A				

#	Bit(s)	Field	Description	Type	Reset
A	4:0	GPADATA<1024>	*Port A data* Data written to or read from Port A Note: the address shown is the base address for access. The memory space actually covers 1024 bytes. The 8 bits of data written/read are masked by bits 9:2 of the address used. This allows access of specifically enabled bits in the data field. For direct access of the full GPIO port, the address of the port, must be incremented by 0x3FC. This has the effect of setting address[9:0] to 0b11111111_00 which enables read and write access of all bits in the port.	dual	0

GPIOA	**GPBDATA**	0x50014400
	GPIO Port B Data	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													A		

#	Bit(s)	Field	Description	Type	Reset
A	2:0	GPBDATA<1024>	*Port B data* Please see description for Port A	dual	0

GPIOA	GPLDATA	0x50014800
	GPIO Port LIN Data	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														A	

#	Bit(s)	Field	Description	Type	Reset
A	1:0	GPLDATA<1024>	*LIN port data* <0>: LIN1, <1>: LIN2	dual	0

GPIOA	GPENA	0x50014C00
	GPIO Port Enables	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													A	B	C

#	Bit(s)	Field	Description	Type	Reset
A	2	GPLENA	*GPL Enable*	r/w	1
B	1	GPBENA	*GPB Enable*	r/w	1
C	0	GPAENA	*GPA Enable*	r/w	1

GPIOA	GPAPL	0x50014C04
	GPIO Port A Lower Nibble Pins Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		A	B	C	D	E	F			G	H	I	J	K	L			M	N	O	P	Q	R			S	T	U	V	W	X

#	Bit(s)	Field	Description	Type	Reset
A	29	GPAACTDET[3]	*Pin 3 activity interrupt*	ro	n/a
B	28	GPACLR[3]	*Pin 3 interrupt clear — cleared automatically after each write*	wo	0
C	27	GPAFE[3]	*Pin 3 falling edge enable*	r/w	0
D	26	GPARE[3]	*Pin 3 rising edge enable*	r/w	0
E	25	GPAIE[3]	*Pin 3 interrupt mask*	r/w	0
F	24	GPADIR[3]	*Pin 3 output enable*	r/w	0
G	21	GPAACTDET[2]	*Pin 2 activity interrupt*	ro	n/a
H	20	GPACLR[2]	*Pin 2 interrupt clear — cleared automatically after each write*	wo	0
I	19	GPAFE[2]	*Pin 2 falling edge enable*	r/w	0
J	18	GPARE[2]	*Pin 2 rising edge enable*	r/w	0
K	17	GPAIE[2]	*Pin 2 interrupt mask*	r/w	0
L	16	GPADIR[2]	*Pin 2 output enable*	r/w	0
M	13	GPAACTDET[1]	*Pin 1 activity interrupt*	ro	n/a
N	12	GPACLR[1]	*Pin 1 interrupt clear — cleared automatically after each write*	wo	0
O	11	GPAFE[1]	*Pin 1 falling edge enable*	r/w	0
P	10	GPARE[1]	*Pin 1 rising edge enable*	r/w	0
Q	9	GPAIE[1]	*Pin 1 interrupt mask*	r/w	0
R	8	GPADIR[1]	*Pin 1 output enable*	r/w	0
S	5	GPAACTDET[0]	*Pin 0 activity interrupt*	ro	n/a
T	4	GPACLR[0]	*Pin 0 interrupt clear — cleared automatically after each write*	wo	0
U	3	GPAFE[0]	*Pin 0 falling edge enable*	r/w	0
V	2	GPARE[0]	*Pin 0 rising edge enable*	r/w	0
W	1	GPAIE[0]	*Pin 0 interrupt mask*	r/w	0
X	0	GPADIR[0]	*Pin 0 output enable*	r/w	0

GPIOA	GPAPU	0x50014C08
	GPIO Port A Upper Nibble Pins Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										A	B	C	D	E	F

#	Bit(s)	Field	Description	Type	Reset
A	5	GPAACTDET[4]	*Pin 4 activity interrupt*	ro	n/a
B	4	GPACLR[4]	*Pin 4 interrupt clear — cleared automatically after each write*	wo	0
C	3	GPAFE[4]	*Pin 4 falling edge enable*	r/w	0
D	2	GPARE[4]	*Pin 4 rising edge enable*	r/w	0
E	1	GPAIE[4]	*Pin 4 interrupt mask*	r/w	0
F	0	GPADIR[4]	*Pin 4 output enable*	r/w	0

GPIOA	GPBPL	0x50014C0C
	GPIO Port B Pin Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										A	B	C	D	E	F			G	H	I	J	K	L			M	N	O	P	Q	R

#	Bit(s)	Field	Description	Type	Reset
A	21	GPBACTDET[2]	*Pin 2 activity interrupt*	ro	n/a
B	20	GPBCLR[2]	*Pin 2 interrupt clear* *— cleared automatically after each write*	wo	0
C	19	GPBFE[2]	*Pin 2 falling edge enable*	r/w	0
D	18	GPBRE[2]	*Pin 2 rising edge enable*	r/w	0
E	17	GPBIE[2]	*Pin 2 interrupt mask*	r/w	0
F	16	GPBDIR[2]	*Pin 2 output enable*	r/w	0
G	13	GPBACTDET[1]	*Pin 1 activity interrupt*	ro	n/a
H	12	GPBCLR[1]	*Pin 1 interrupt clear* *— cleared automatically after each write*	wo	0
I	11	GPBFE[1]	*Pin 1 falling edge enable*	r/w	0
J	10	GPBRE[1]	*Pin 1 rising edge enable*	r/w	0
K	9	GPBIE[1]	*Pin 1 interrupt mask*	r/w	0
L	8	GPBDIR[1]	*Pin 1 output enable*	r/w	0
M	5	GPBACTDET[0]	*Pin 0 activity interrupt*	ro	n/a
N	4	GPBCLR[0]	*Pin 0 interrupt clear* *— cleared automatically after each write*	wo	0
O	3	GPBFE[0]	*Pin 0 falling edge enable*	r/w	0
P	2	GPBRE[0]	*Pin 0 rising edge enable*	r/w	0
Q	1	GPBIE[0]	*Pin 0 interrupt mask*	r/w	0
R	0	GPBDIR[0]	*Pin 0 output enable*	r/w	0

GPIOA	GPLPL	0x50014C14
	GPIO Port LIN Pin Control	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																		A	B	C	D	E	F			G	H	I	J	K	L

#	Bit(s)	Field	Description	Type	Reset
A	13	GPLACTDET[1]	*LIN2 activity interrupt*	ro	n/a
B	12	GPLCLR[1]	*LIN2 interrupt clear* *— cleared automatically after each write*	wo	0
C	11	GPLFE[1]	*LIN2 falling edge enable*	r/w	0
D	10	GPLRE[1]	*LIN2 rising edge enable*	r/w	0
E	9	GPLIE[1]	*LIN2 interrupt mask*	r/w	0
F	8	GPLDIR[1]	*LIN2 output enable*	r/w	0
G	5	GPLACTDET[0]	*LIN1 activity interrupt*	ro	n/a
H	4	GPLCLR[0]	*LIN1 interrupt clear* *— cleared automatically after each write*	wo	0
I	3	GPLFE[0]	*LIN1 falling edge enable*	r/w	0
J	2	GPLRE[0]	*LIN1 rising edge enable*	r/w	0
K	1	GPLIE[0]	*LIN1 interrupt mask*	r/w	0
L	0	GPLDIR[0]	*LIN1 output enable*	r/w	0

17.20 TIMER0

TIMER0	**COUNT**	0x50020000
	Timer Counter Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	COUNT	Initial counter value. The timer will count from this value to 0xFFFFFFFF and roll over to 0x00000000. At this point it will generate an interrupt if enabled. The interrupt routine is responsible for reloading the value if needed as this timer does not auto-reload the original content	r/w	0

TIMER0	**CFG**	0x50020004
	Timer Control Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															A

#	Bit(s)	Field	Description	Type	Reset
A	0	ENA	*Enable* This bit starts/stops the timer: < br> 1 = Timer Running < br> 0 = Timer Inactive	r/w	0

17.21 TIMER1

TIMER1	**COUNT**	0x50020008
	Timer Counter Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	COUNT	Initial counter value. The timer will count from this value to 0xFFFFFFFF and roll over to 0x00000000. At this point it will generate an interrupt if enabled. The interrupt routine is responsible for reloading the value if needed as this timer does not auto-reload the original content	r/w	0

TIMER1	**CFG**	0x5002000C
	Timer Control Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															A

#	Bit(s)	Field	Description	Type	Reset
A	0	ENA	*Enable* This bit starts/stops the timer: < br> 1 = Timer Running < br> 0 = Timer Inactive	r/w	0

17.22 TIMER2

TIMER2	COUNT	0x50020010
	Timer Counter Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	COUNT	Initial counter value. The timer will count from this value to 0xFFFFFFFF and roll over to 0x00000000. At this point it will generate an interrupt if enabled. The interrupt routine is responsible for reloading the value if needed as this timer does not auto-reload the original content	r/w	0

TIMER2	CFG	0x50020014
	Timer Control Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															A

#	Bit(s)	Field	Description	Type	Reset
A	0	ENA	*Enable* This bit starts/stops the timer: < br> 1 = Timer Running < br> 0 = Timer Inactive	r/w	0

17.23 WDT1

Watchdog timer

WDT1	**CFG**	0x50020018
	Config	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																											A		B	C	D

#	Bit(s)	Field	Description	Type	Reset
A	4:3	PRESET	Defines the watchdog timeout period. It means that the WDT internal counter will count from 0 to the prescaler value at the system clock speed and trigger if not cleared. For instance, a system running from a 30MHz Crystal with WDTPRES[1I0] = 10 will trigger the WDT after approximately 0.14 seconds if not cleared properly and in time by the application 0x**0** — 2^{13} / System Clock 0x**1** — 2^{19} / System Clock 0x**2** — 2^{22} / System Clock 0x**3** — 2^{32} / System Clock	r/w	0
B	2	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 cleared by the application	r/w	0
C	1	RSTEN	*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	r/w	0
D	0	ENA	*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	r/w	0

WDT1	**KEY**	0x5002001C
	Writing the sequence KEY0, KEY1 to this register will clear (pet) the watchdog - preventing it from timing out and resetting the system.	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	KEY	To clear the WDT counting the following words must be written in this order and without any other instruction between then: KEY0: 0x3C570001. KEY1: 0x007F4AD6	r/w	0

17.24 FLASH

FLASH	**FLADDR**	0x50020020
	Destination address for flash write / erase operation	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															A																

#	Bit(s)	Field	Description	Type	Reset
A	16:0	ADDR	Target address for write/erase operation. In byte writes, this is the read address of the flash to be written to. In erase modes, it is a read address inside the sector to be erased. This register must be written in the correct sequence or the operation will fail	r/w	0xFFFF

FLASH	FLWRDT	0x50020024
	Flash data to be written	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	DATA	Content to be written into the targeted address. This register must be written in the correct sequence or the operation will fail	r/w	0

FLASH	UNLBWR	0x50020028
	Flash data unlock register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	UNLOCK_WRITE	Control register to unlock write. A value of 0x55555555 must be written to this address at the correct point in the write sequence or the operation will fail	r/w	0

FLASH	BWRSTRT	0x5002002C
	Flash write start register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	WRITE_START	Control register to start a write. A value of 0xAAAAAAAA must be written to this address at the correct point in the write sequence or the operation will fail	r/w	0

FLASH	UNLSER	0x50020030
	Flash sector erase unlock register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	UNLOCK_ERASE	Control register to unlock a sector erase. A value of 0x66666666 must be written to this address at the correct point in the sector erase sequence or the operation will fail	r/w	0

FLASH	SERSTRT	0x50020034
	Flash sector erase start register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	ERASE_START	Control register to commit a sector erase. A value of 0x99999999 must be written to this address at the correct point in the sector erase sequence or the operation will fail	r/w	0

FLASH	**FLSCTRL**	0x50020040
	Flash control register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														A	

#	Bit(s)	Field	Description	Type	Reset
A	1:0	CTRL	Number of wait states used in the reading process. Each read from flash memory will take number of cycles equal to 1+RWC to complete	r/w	1

FLASH	**FLSCP**	0x50020044
	Flash code protection register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	CODE_PROT	Code Protection / SerialWire Lockout Control Code protection control register. Write a value of 0xF2E11047 to disable the SerialWire interface. Write 0x00000000 to enable it. This allows the user program to disable the SerialWire interface to prevent unauthorized debug access to the part. NOTE1: This register does not lock the Flash Memory against read/write/erase by the applications program. Instead what it does is to disable all communications with the debug interface, therefore preventing any external attack. The application code is still able to modify the flash content. NOTE2: Upon Power-On Reset or Normal Reset the system disables the communication for a small time interval (8192 clock cycles). If the application needs to be protected it is mandatory to set this register with the appropriate code in the beginning of the initialization process and before the internal hardware enable the debug communication	r/w	0

FLASH	**FLS_UNLOCK_CTRL_OP**	0x50020050
	Flash Unlock Control Operation Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A																															

#	Bit(s)	Field	Description	Type	Reset
A	31:0	UNLOCK_CTRL_OP	*Flash Control Operation Register Unlock value* 0xACDC_1972 needs to be written in this register to unlock the Control Operation Register access. When this register is read, it returns the state of the lock: 0: The Control Operation Register is locked. The Control Operation Register (FLASH_CTRL_OP) cannot be written. 1: The Control Operation Register is unlocked. The Control Operation Register (FLASH_CTRL_OP) can be written. Note: After each write to the FLASH_CTRL_OP register, the state of the lock is cleared and the pattern needs to be written again to allow a new configuration of the register	r/w	0

FLASH	**CTRL_OP**	0x50020054
	Flash Control Operation Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													A		B

#	Bit(s)	Field	Description	Type	Reset
A	2:1	SIZE	*SIZE of the write operation* Refer to data sheet for more information of the use of this field	r/w	0
B	0	CHIP	*CHIP bit* This bit is only used during the Erase operation. It allows the system to erase more than one sector. 0: The Erase operation will only erase the sector selected by the FLASH_ADDR register value. 1: The Erase operation will erase the full main array of the flash	r/w	0

FLASH	TRIM	0x50020058
	Flash Trim Register	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														A	B	C															

#	Bit(s)	Field	Description	Type	Reset
A	17	SLEEPDEEP_CFG	*Deep Sleep VDD_IO configuration* This register will be automatically populated with the value stored in the NVR sector 1 (@0001_0000). When set, the system will NOT be reset if VDD_IO is going away during Deep Sleep mode. Otherwise (0), the system is reset if VDD_IO is removed	r/w	0
B	16	SDIO_TIMING_CFG	*SDIO interface timing configuration* This register will be automatically populated with the value stored in the NVR sector 1 (@0001_0000). When set, the SDIO/INT signals are captured on the rising edge of CLK. When cleared, these data are captured on the falling edge of CLK	r/w	1
C	15:0	OSC_TRIM	*Oscillator Trim Value* This register will be automatically populated with the value stored in the NVR sector 1 (@0001_0000)	r/w	0x86

18 Interrupts

There are 22 interrupt channels - 16 used by the ASIC die and 6 used by the Verne MCU..

Interrupt #	Interrupt name	Block	Comment
0	FrequencyCounterFIFO_IRQn	US_TRX	
1	Brownout_IRQn	CRGA	Brown-out detect / reset
2	Watchdog_A_IRQn	WDT	Watchdog timer interrupt
3	GPIO_IRQn	GPIO	
4	EnvelopeFIFO_IRQn	US_TRX	
5	LIN_Master_IRQn	LIN	LIN Master
6	LIN_Wakeup_IRQn	LIN	Wakeup from LIN message
7	LIN_Slave_IRQn	LIN	LIN Slave
8	PWM_0_IRQn	PWM	
9	LINDLYCOMP_IRQn	LIN	
10	USTRXError_IRQn	US_TRX	
11	USTRXStatus_IRQn	US_TRX	
12	UART_A_IRQn	UART	ASIC UART interrupt
13	ADC_A_IRQn	ADC_CTRL	
14	ClockCal_IRQn	CCAL	
15	Lullaby_IRQn		Indicates when the ASIC can safely enter hibernate mode
16	Timer0_IRQn	TMR_0	MCU Timer 0
17	Timer1_IRQn	TMR_1	MCU Timer 1
18	Timer2_IRQn	TMR_2	MCU Timer 2
19	Watchdog_IRQn	WDT	
20	BTE_IRQn	BTE	Block Transfer Engine
21	SDIO_IRQn		Serial Data IO

18.1 Interrupt support on GPIOs

Interrupts are generated based on the transitional value of a pin (e.g. button press). At power-on reset, the GPA lines are HighZ and the GPB lines default to inputs.

The GPIO interrupts are synchronous to the system clock (clk_sys).

18.1.1 GPIO Interrupt Handling

The iND83207 GPIO shares a single hardware interrupt for all 3 GPIO ports (A, B & L). To ensure that there is no possibility of missing an interrupt request from any of the ports that occurs whilst you are handling a request from the GPIO, it is best practise to continue to record and clear flags that are set in the individual interrupt status registers (for each interrupt-enabled bit of each GPIO port) until a value of zero is read back from all of them.

The following code implements the start of a GPIO irq handler that uses such an approach.

```c
#define GPIOA_ACT_DET_FLAG_mask 0x20202020
#define GPIOA_ACT_DET_FLAG_TO_CLEAR_rshft 1
#define N_GPIO_FLAG_BYTES 4

void GPIO_Handler (void)
{
    // to avoid any possibility that a newly detected edge that we don't handle during this
call of the
    // GPIO interrupt handler (e.g. one that is detected at any point during the execution
of the handler)
    // we need to make sure that all the individual interrupt request lines (from each
interrupt-enabled,
    // bit of each GPIO port) are clear (thus any new edge will lead to a positive edge on
the overall
    // request line and therefore a new interrupt event at the input to the NVIC)
    //
    // we do this by starting the handler with a short piece of code that records (and
clears) all the
    // asserted edge detection flags of the GPIO peripheral - continuing to do so until it
has observed
    // no new flags on any port since it last cleared a flag...

    uint32_t all_flags_found = 0; // a variable in which we build up a record of all the
SFR 'activity detec(ed)'
    //  flags that we find, all bits for all three ports will end up in this register -
laid out as follows...
    //
    //  31 30 29 28 27 26 25 24  23 22 21 20 19 18 17 16  15 14 13 12 11 10  9  8   7  6  5
4  3  2  1  0
    //                                                                L1 L0    B2 B1 B0
A4 A3 A2 A1 A0
    //
    uint32_t flags_etc; // temporary storage for the contents of a GPIO ctrl/status word
    uint32_t new_flags; // a record of any new flags read from the current byte we are
processing
    uint32_t *p_flags; // a pointer to the relevant SFR control/status byte
    uint8_t i_flag_word = 0; // which of the SFR words we are currently processing
    int8_t leftshift; // a shift used to align the least significant SFR flag with 'all
flags found'
    uint32_t found_flag_nibble_mask; // a mask used when copying flags into 'all flags
found'
    uint8_t reads_since_new_flag = 0; // how many times we have read flag SFR bytes since
finding any flags
    uint8_t i_bit, n_bits;
    const uint8_t n_bits_per_word[] = {4, 1, 3, 2}; // the number of GPIO bit controlled by
each SFR word

    while (reads_since_new_flag < N_GPIO_FLAG_BYTES) {

      // set up some variables that are dependent on the loop counter (which SFR byte we
are dealing with)...
      p_flags = (volatile uint32_t *)(0x50014C04 + ((i_flag_word == 3)?
16:(4*i_flag_word)));
      leftshift = (i_flag_word * 4) - 5; // word->nibble bit_count minus the shift required
for the first flag
      n_bits = n_bits_per_word[i_flag_word]; // the number of bits in the GPIO port (8 in
'A', 4 in 'B' & 'L')
      found_flag_nibble_mask = 0x0F << (4 * i_flag_word); // isolates bits of the
appropriate nibble for this byte
```

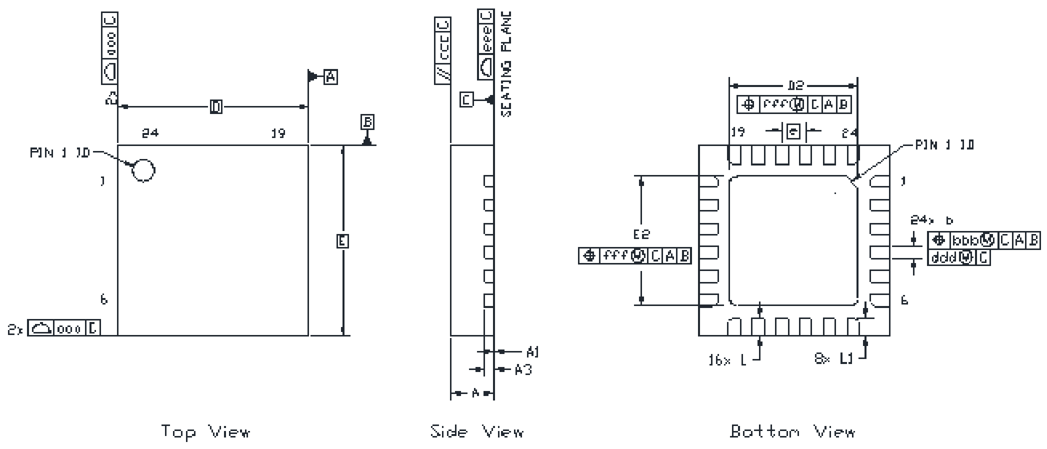
```
      // start by checking for any new 'activity detect(ed)' flags in this peripheral
control/status SFR byte
      // (keeping a copy of the entire word so that we can avoid a read-modify-write when
we write to it later)...
      flags_etc = *p_flags;
      new_flags = flags_etc & GPIOA_ACT_DET_FLAG_mask;
      while (new_flags) {
        // we found 'new' flags so we need to zero our run length 'without finding new
flags' counter
        reads_since_new_flag = 0;
        // and clear all the new flags that we found in the peripheral control/status
register...
        *p_flags = flags_etc | (new_flags >> GPIOA_ACT_DET_FLAG_TO_CLEAR_rshft);
        // we now loop over the individual flags of this byte, recording which ones we have
found...
        new_flags = (leftshift > 0)? (new_flags << leftshift) : (new_flags >> leftshift); //
moves the least significant new flag to the appropriate bit position in the overall flags
register
        for (i_bit=0; i_bit<n_bits; i_bit++) {
          all_flags_found |= new_flags & found_flag_nibble_mask; // record the new flag
          new_flags >>= 7; // moves the next most significant flag to the appropriate bit
position for the 'all flags' variable
          if (new_flags == 0) break; // no need to continue if there are no flags left
        }
        // re-read the SFR to see if any new flags have been set (since we cleared it)...
        flags_etc = *p_flags;
        new_flags = flags_etc & GPIOA_ACT_DET_FLAG_mask;
      }
      reads_since_new_flag++; // we only bypass or leave the loop when we read 'no new
flags'
      if (++i_flag_word >= N_GPIO_FLAG_BYTES) i_flag_word = 0; // increment the flag word
counter
  }

}
```

19 Package Drawing

QFN 4x4 24pins (0.5mm pitch)

Package Outline Drawing



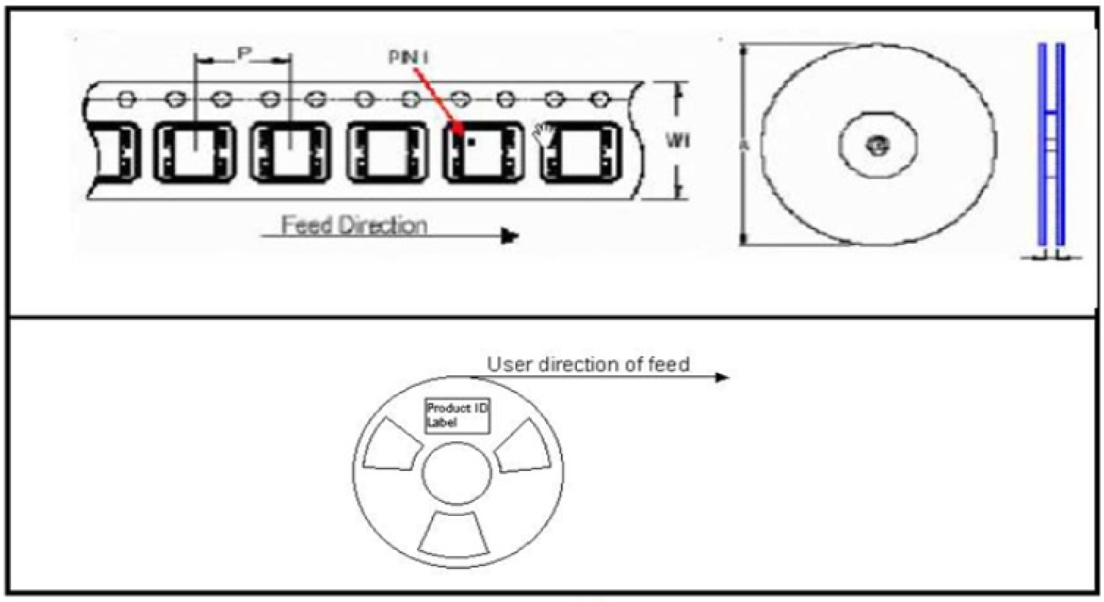
Dimensions and Tolerances

DESCRIPTION		SYMBOL	MIN	NOM	MAX
total thickness		A	0.8	0.9	1.0
stand-off		A1	0.00	0.02	0.05
L/F thickness		A3	.203 REF		
lead width		b	0.18	0.25	0.30
body size	X	D	4.0 BSC		
	Y	E	4.0 BSC		
lead pitch		e	0.5 BSC		
EP size	X	D2	2.50	2.65	2.80
	Y	E2	2.50	2.65	2.80
lead length		L	0.30	0.40	0.45
		L1	0.30	0.40	0.45
package edge tolerance		aaa	0.15		
lead array offset		bbb	0.10		
mold flatness		ccc	0.10		
lead offset		ddd	0.05		
lead coplanarity		eee	0.08		
exposed pad offset		fff	0.10		

Notes:

- Controlling dimensions are in mm
- Dimensioning and tolerancing per ASME Y14.5M
- This drawing conforms to JEDEC outline MO-220, variation VGGD-6 with exception of feature L, L1 which are per supplier designation.

19.1 Tape & Reel packaging



Device orientation in Tape & Reel packaging

20 Environmental

All indie Semiconductor devices meet the proposed RoHS components for cadmium, mercury, hexavalent chromium, polybrominated biphenyls (PBBs), and polybrominated diphenyl ethers (PBDEs). The composition of indie Semiconductor devices do not contain halogens (including bromine) and inorganic (red) phosphorous as an alternative flame-retardant system.

21 Qualification

Product qualification is performed according to the automotive standard AEC-Q100 described in QM N0. 07PL0254

22 References

Indie semiconductor ARM CORTEX M0 core programming guide.

23 Contacts

United States

32 Journey

Aliso Viejo, California 92656, USA

Tel: +1 949-608-0854

sales@indiesemi.com

China

Zhuang Jian

Room 1403, #25 Zhizhubandao

Lane 333, Dongchuan Rd.

Minhang Distrcit

Shanghai, China

sales@indiesemi.com

Scotland

4th Floor, Hobart House

80 Hanover Street

Edinburgh EH2 1EL, Scotland

sales@indiesemi.com

24 Appendix 1: Errata

24.1 LIN Compliance

The LIN master in iND83207 fails to meet the following parts of the LIN Specification Package (Revision 2.2A):

- Param 21 VSerDiode should have a voltage drop at the serial diodes of
    - 0.4V min ; 0.7V typ ; 1.0V max
    - iND83207 has diode functionality with a typical voltage drop of 1.6V
- Param 25 Rmaster should have a resistance of
    - 900 Ohm min ; 1000 Ohm typ ; 1100 Ohm max
    - iND83207 has a typical resistance which meets the spec
    - Over process & temperature the spread will be wider than the specified range

When operating as a LIN slave, iND83207 is fully compliant with the LIN Specification Package (Revision 2.2A).

Document history

This document was generated on Monday 31 January 2022 at 08:40 AM (GMT).