D-FLOW

ULTRASONIC TECHNOLOGY PROVIDER

DFLOW UFO2 Specification (Revision 0.13.4)

Ultrasonic Flowmeter ASIC - Customer specification

Features

- Ultrasonic control & measurement interface
- Ultrasonic HF timing capability with 40ps single shot time resolution and sing-around capability
- 5 channel ΣΔ-ADC for temperature sensors
- On-chip temperature sensor
- Battery sensing
- Internal voltage regulators
- Voltage regulator for external circuits
- HF and LF oscillators
- CPU system with MSP430 compatible instruction set
- 64-bit Floating point unit
- Digital interfaces
  - 2 x UART
  - SPI (Master /Slave)
  - I2C (Master / Slave)
  - 2x PWM
  - 12 GPIOs
  - Pulse-Count-Output
- 2-wire JTAG interface
- 32kB Flash and 5kB SRAM on chip
- Ultra low power option with a average of 40µA with one ultrasonic measurements per second

Typical applications

- Ultrasonic flow measurements
- Ultrasonic distance measurements
- Temperature measurements
- Direct differential temperature measurements
- Energy measurements

Brief Functional Description

The UFO2 ASIC is a single chip solution for temperature- and ultrasonic time of flight measurements. It is comprised of an analog measuring part and a digital micro controller part.

Ultrasonic transducers and resistive temperature sensors can be connected directly to the ASIC. The UFO2 supports the use of up to four transducers. It also supports the use of multiple ASIC:s when more transducers are required.

Temperature measurements are sampled by the integrated ΣΔ-ADC. Three external temperature sensors can be connected at the same time to the temperature interface.

An integrated 64-bit FPU as well as an integer multiplier can be used by the micro controller for all calculations. Two separate watchdogs are also included to suit a wide range of applications.

In order to reduce the overall cost of the system, the UFO2 ASIC has everything needed for basic flow meter applications and temperature measurements implemented on chip. At the same time it is a versatile circuit allowing for example the transducer drive stages or the analog filter to be implemented externally. The input/output interfaces of the ASIC also support external circuits as external micro controllers or displays. It also includes a voltage regulator for external circuits.

Suitable application for the ASIC, can be both high performance meters and high volume battery powered meters. Battery powered applications can use different power down features to achieve very low power consumption. Since both temperature and flow can be measured, energy metering capability can for example be realized. There are no practical limitation on the size of the meter where the ASIC can be used. External drivers can be applied if transducer distances are long or if signal attenuation is high.

1 Typical Application

Figure 1 shows external components at the different UFO2 interfaces. In figure 2 all interfaces are connected and different options are shown as well.

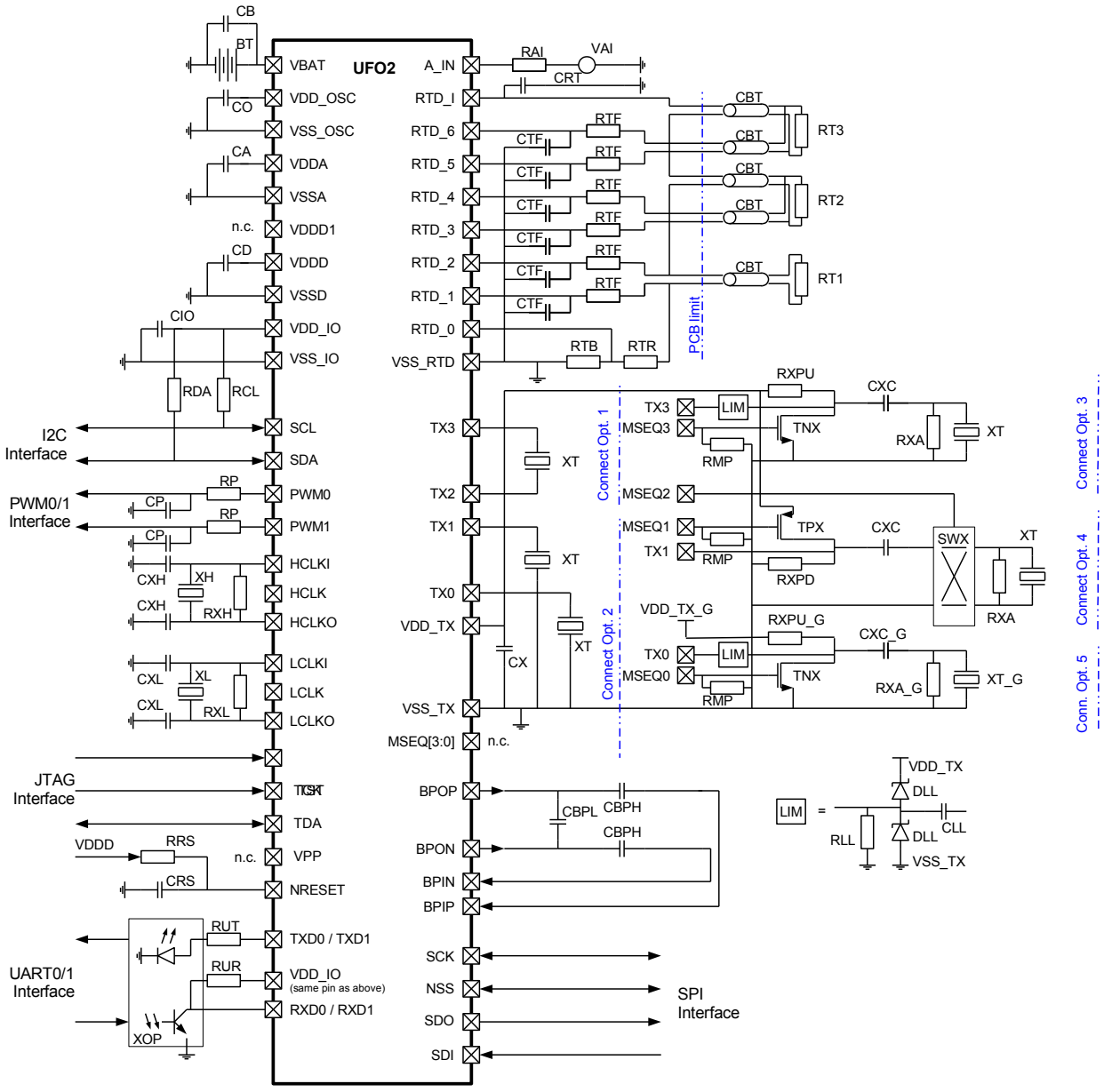


Figure 1: Example application diagram

2 Pin-out

2.1 Pin Description

No.	Name	Type[1,2]	1st Function	2nd Function	3rd Function
1	VDDD	SO	Digital Supply Output (1.8V) Power Domain 0, always on		
2a 2b	VSSD VSS_IO	G NCDB G NCDB	Ground Ground		
3	VDD_IO	SO	Digital Input Output Supply Output (3.0V)		
4	VDDD1	SO	Digital Supply Output (1.8V) Power Domain 1, switched		
5	SDO	DIO	GPIO4	SPI Data Out	
6	SDI	DIO	GPIO5	SPI Data In	
7	NSS	DIO	GPIO6	SPI nSlave Select	
8	SCK	DIO	GPIO7	SPI Clock	
9	SCL	DIO	GPIO8	I2C Clock	
10	SDA	DIO	GPIO9	I2C Data	
11	PWM0	DIO-AIO	GPIO10	PWM 0 Out	LCLK Clock Out
12	PWM1	DIO-AIO	GPIO11	PWM 1 Out	HCLK Clock Out
13	TCK	DIO	GPIO12		JTAG Clock
14	TDA	DIO	GPIO13	PULSE Out	JTAG Data
15	NRESET	DI-OD-PU	System nReset		
16	TST	DI-PD	Test mode		
17	VSS_RTD	G NCDB	Ground Temperature Interface		
18	RTD_0	AIO	Temperature Interface 0		
19	RTD_1	AIO	Temperature Interface 1		
20	RTD_2	AIO	Temperature Interface 2		
21	RTD_3	AIO	Temperature Interface 3		
22	RTD_4	AIO	Temperature Interface 4		
23	RTD_5	AIO	Temperature Interface 5		
24	RTD_6	AIO	Temperature Interface 6		
25	RTD_I	AIO	Temperature Interface Current Output		
26	A_IN	AIO	Analog Input		

1

[2]All Digital I/Os can be configured to be Push-Pull or Open Drain.

No.	Name	Type	1st Function	2nd Function	3rd Function
27a 27b 27c	VSSA	G NCDB	Ground		
28a 28b	VDDA	SO	Analog Supply Output (3.0V)		
29	BPON	AIO	Band Pass (for external Components)		
30	BPOP	AIO	Band Pass (for external Components)		
31	BPIN	AIO	Band Pass (for external Components)		ATBUS[0]
32	BPIP	AIO	Band Pass (for external Components)		ATBUS[1]
33	VDD_OSC	SO	HF Frequency Oscillator Supply Output (1.8V)		
34a 34b	VSS_OSC PSUBA	G NCDB G NCDB	Ground Substrate Contact		
35	VPP	AIO-HV	FLASH test pad		
36	VBAT	SI	Power Supply		
37	TX0	AIO	Transducer Driver 0		
38	TX1	AIO	Transducer Driver 1		
39	VDD_TX	SO	Transducer Driver Supply Output (3.0V)		
40	VSS_TX	G NCDB	Ground for Transducer Driver		
41	TX2	AIO	Transducer Driver 2		
42	TX3	AIO	Transducer Driver 3		
43	MSEQ0	DIO	Digital Output for External Transistors 0		
44	MSEQ1	DIO	Digital Output for External Transistors 1		
45	MSEQ2	DIO	Digital Output for External Transistors 2		
46	MSEQ3	DIO	Digital Output for External Transistors 3		
47	LCLKI	AI	Input of LF Clock Amplifier		
48	LCLKO	AO	Output of LF Clock Amplifier		
49	LCLK	DIO	GPIO14	LCLK Clock Out	PULSE Out
50	HCLKI	AI	Input of HF Clock Amplifier		
51	HCLKO	AO	Output of HF Clock Amplifier		
52	HCLK	DIO	GPIO15	HCLK Clock Out	PULSE Out
53	TXD0	DIO	GPIO0	UART 0 Transmit	

No.	Name	Type	1st Function	2nd Function	3rd Function
54	RXD0	DIO	GPIO1	UART 0 Receive	
55	TXD1	DIO	GPIO2	UART 1 Transmit	
56	RXD1	DIO	GPIO3	UART 1 Receive	

Notes:

D=Digital, A=Analog, S=Supply, G=Ground, I=Input, O=Output, T=Test Pin, OD=Open Drain, HV=High Voltage.

NCDB = Not Connected, Internal Down Bond to the Die Attach Area

All Digital I/Os can be configured to be Push-Pull or Open Drain

2.2 Pin-Out Overview

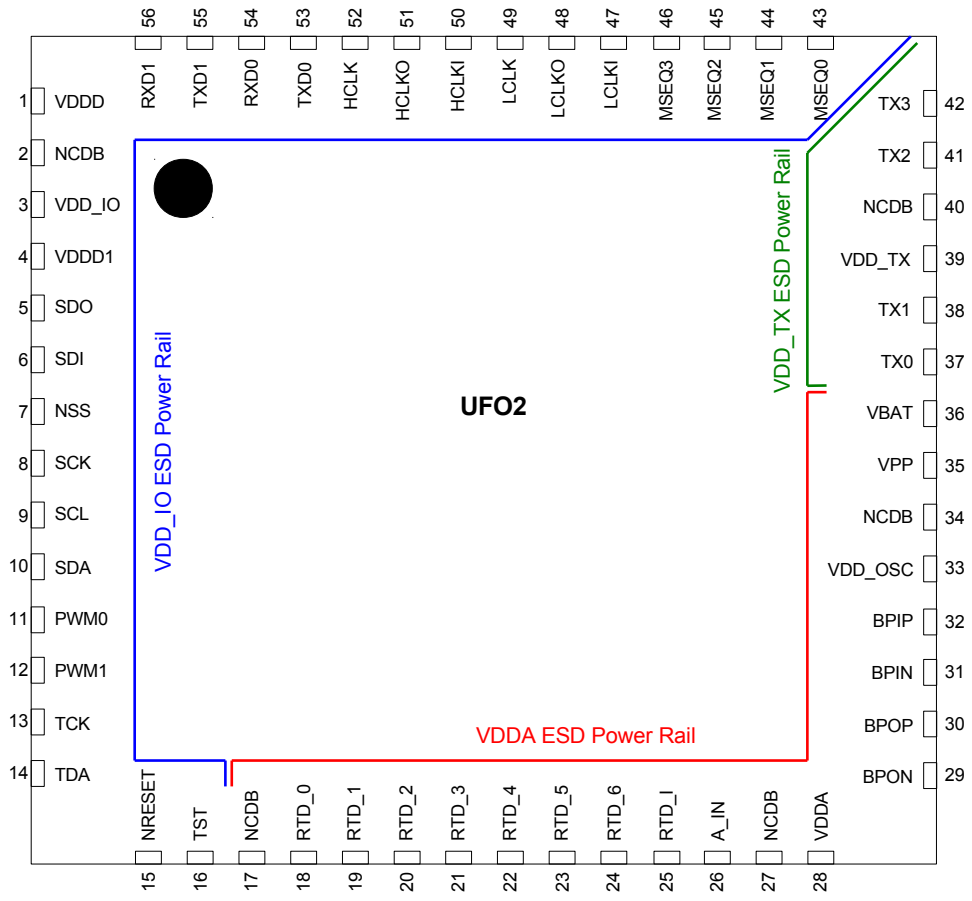


Figure 2: Pin Out Overview

3 System Description

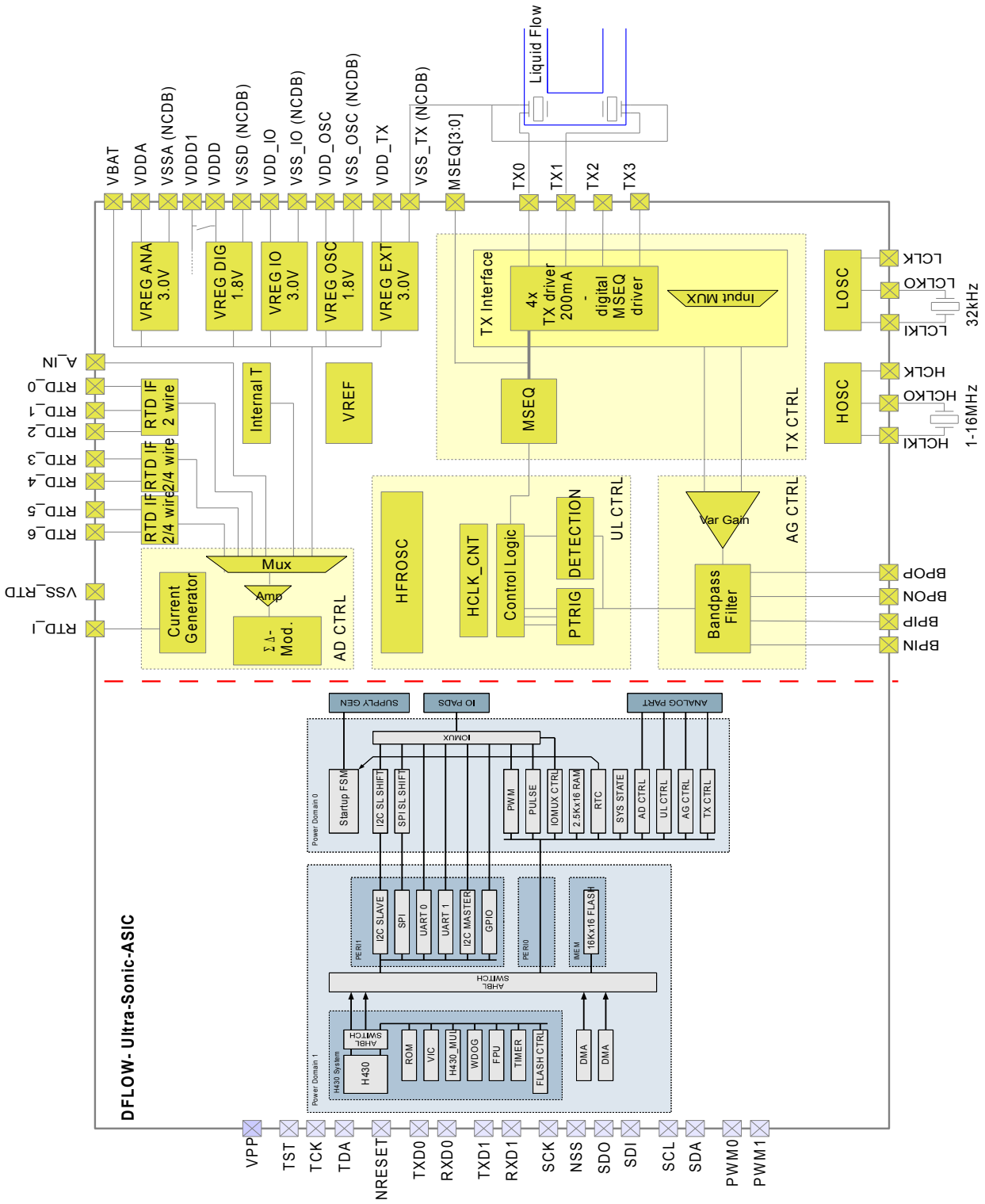


Figure 3: System block diagram

3.1 General Description

The UFO2 ASIC described in this specification is a mixed mode low power circuit for mainly ultrasonic flow measurements, absolute temperature measurements and differential temperature measurements. It contains an analog measuring part and a digital MCU part, compare with the system block diagram in figure Error: Reference source not found.

The UFO2 ASIC contains all electronic functions needed for an ultrasonic flow and energy meter as shown in the system block diagram. 0 to 4 ultrasonic transducers and 0 to 3 temperature sensors can be connected to the ASIC. In short the UFO2 ASIC is able of the following:

- The ASIC measures acoustic traveling times between 1 to 4 ultrasonic transducers. This is done with the purpose of calculating distance or flow. In the case of a distance measurement one transducer serves as both the sender of an acoustic pulse and the receiver of the reflected same pulse. In the case of flow measurements the acoustic traveling time between one sending and another receiving transducer is measured.
- The ASIC measures the absolute temperature at 1 to 3 resistive platinum sensors for control and compensation purposes.
- The ASIC measures the differential temperature between 2 resistive platinum sensors for energy calculation purposes. Preferably the two measurements are performed simultaneously.
- The ASIC contains different power down features in order to ensure a very low power performance.
- The MCU part calculates distance, flow velocity, flow rate or energy. This will involve numerical calculations and table look up operations. The precision in calculations is double precision (64-bit floating point).
- The 64-bit FPU improves the efficiency in the calculations.
- The ASIC is programmed, debugged and communicated with by using a digital communication interface.
- There are different communication interfaces.
- In other more complex applications the UFO2 ASIC is capable of working together with for example external microprocessors, display drives and additional UFO2 circuits. The ASIC can work both as “master” and “slave” in synchronized multiple ASICs configurations.
- It is also possible to realize external transducer drive stages and filters. In for example gas measurement applications the excitation voltage needs to be a lot higher and the center frequency of the transducers a lot lower.

3.2 Measurements

3.2.1 Ultrasonic time/flow measurements

A transit time ultrasonic flow meter uses ultrasonic transducers that can both send and receive sound. The sound is transmitted between the transducers through the fluid that pass through the flow meter. The transducers are organized so that the sound velocity will interact with the flow velocity. The sound propagation time between the transducers is measured in both directions. If there is no fluid motion the two times are ideally equal but if there is fluid velocity present the interaction with the sound velocity will cause one time, the

downstream one, to decrease and the other time, the upstream one, to increase. These two times are used to calculate the flow velocity. The UFO2 ASIC can both measure the ultrasonic transit times and perform the necessary calculations.

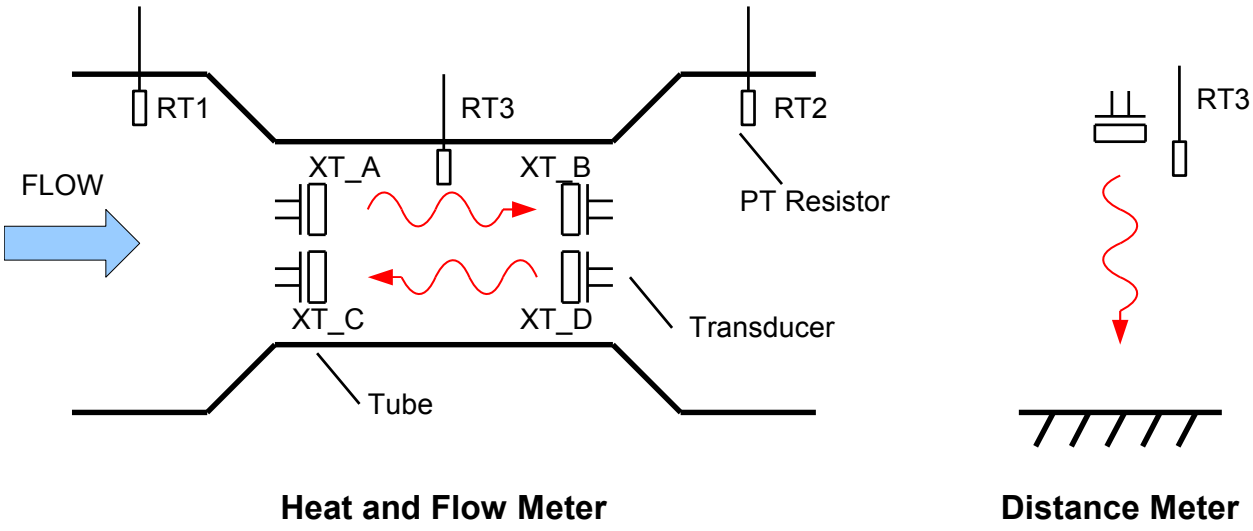


Figure 4: General Meter Outline

In its simplest form a flow meter consists of only two transducers and no temperature sensor. If comparing with figure 4 where the flow velocity is from left to right, the transit time measured when transducer TX_A is sending and transducer TX_B is receiving would form the downstream time t_{down}. The time measurement in the other direction from TX_B to TX_A gives the time t_{up}.

In order to increase the time resolution of these time measurements the ASIC can also utilize a sing-around method. The UFO2 design has the capability to start, maintain, and terminate a sing-around loop. Still referring to figure 4 using transducers TX_A and TX_B, the sing-around method basically works like this: An ultrasound pulse is transmitted from transducer TX_A to transducer TX_B. When the arrival of the sound pulse at transducer TX_B is detected by the ASIC it immediately transmits a new sound pulse from transducer TX_A. When this sound pulse is again detected at transducer TX_B a new pulse is transmitted from transducer TX_A and so on. A sing-around loop is now in operation. This sing-around loop is repeated a predetermined number ***N*** times. The time t_{down} that is measured by the UFO2 ASIC is the total time of these ***N*** sing-around loops. The same procedure is repeated in the reversed sound direction in order to get the time t_{up}.

Compared to a single shot time measurement the advantage of the sing-around method is that the time resolution is improved by a factor equal to the number of sing-around loops ***N***. The transducer distance is virtually increased ***N*** times. The number of loops can be set in the LOOPC register from 1 to 65535. This is an important advantage since the time resolution is often the main limiting factor for the meter performance in the low flow region.

This is especially true in the case of small flow meters for liquids where the transducer distance is short and the speed of sound is high.

The transit times t_{up} and t_{down} can then be used to calculate for example speed of sound, flow velocity or flow rate. These calculations can be done by the UFO2 micro controller with FPU assistance.

The use of the four analog measuring modules (AD CTRL, UL CTRL, AG CTRL, TX CTRL) of the UFO2 ASIC is made easy to use by a software library interface. The UL CTRL, the AG CTRL and the TX CTRL

modules concerns the ultrasonic measurements.

The TX CTRL module contains the transducer interface. The software library interface supports four different ultrasonic measuring modes:

Measuring mode1: This mode is suitable in applications with large transducer distances, gas applications and applications where near zero flow performance is less critical. This mode uses two ultrasonic transducers. The LOOPC register is preferably set to 1 (single shot) or 2.

Measuring mode 2: This mode is intended to be used in symmetrical 2-path meters with four transducers. Measurements are performed interlinked in the two paths – two measurement loops in the first path, the next two loops in the second path, the next two back in the first and so on. Equally many sing-around loops are performed in each path and the resulting time measurement is an average of the two paths. The LOOPC register is preferably set to multiples of 4.

Measuring mode 3: This mode is used for LOOPC higher than 2 in 1-path (2 transducers) applications. The LOOPC register is preferably set to n*8+4 (n≥0).

Measuring mode 4: This mode is used for distance measurements using only one transducer that simultaneously sends and receives sound . This mode is under development.

The AG CTRL module performs signal conditioning. By adjusting the gain settings different transducer distances, different fluids or other application specific properties that influence the signal amplitude can be compensated for. The software library interface supports settings of the two stage automatic gain control:

R1: Stage one of the gain control is settable and can perform automatic gain adjustments. The R1 register defines the gain in stage one together with gm1. If an automatic adjustment of the automatic gain control is issued R1 will likely get a new value. This first stage will compensate the signal amplitude for changing properties like temperature and flow velocity.

gm1: Stage one of the gain control is settable and can perform automatic gain adjustments. The gm1 register defines the gain in stage one together with R1. If an automatic adjustment of the automatic gain control is issued gm1 will change but only if R1 reach an end value.

R2: Stage two of the gain control is settable and fixed. The R2 register defines the gain in stage two. If an automatic adjustment of the automatic gain control is issued R2 still stays fixed. This second stage provides the coarse adjustment of the signal amplitude for application specific properties like transducer distance and fluid type.

PTRIG: The PTRIG register defines the threshold value defining the arrival of the signal. When PTRIG is exceeded the signal detection and time measurement functions are armed. PTRIG should be set higher than the signal noise.

PTRIG_LOW: The PTRIG_LOW register defines a threshold value used by the automatic gain control. The automatic gain control strives to exceed PTRIG_LOW. PTRIG_LOW should be set higher than PTRIG. The PTRIG_LOW status is also included in a status report.

PTRIG_HIGH: The PTRIG_HIGH register defines a last threshold value. The status of PTRIG_HIGH is included in the status report but it is not used by the automatic gain control. PTRIG_HIGH is preferably set higher than PTRIG_LOW.

The UL CTRL module controls the sing-around loop and performs the time measurements. The software library interface supports different settings:

LOOPC: The LOOPC register defines the number of sing-around loops. Depending on the measuring mode LOOPC can be set any value between 1 and 65535. Normally LOOPC would be set to between 1 and 50.

SMARK: The SMARK register defines the beginning of the time window in which the received signal should appear. SMARK starts counting at the moment of excitation on the sending transducer. In power down mode the analog modules are turned off until SMARK.

SSPACE: The SSPACE register defines the end of the time window in which the received signal should appear. SSPACE starts counting at the moment of SMARK. If SSPACE elapse without a signal being detected a time-out is issued.

3.2.2 Temperature measurements

Temperature measurements are made using a ΣΔ-ADC. The ADC can be connected with up to three external resistive sensors, an on-chip sensor, a DC voltage supplied on the analog input pin and the battery input pin.

The AD CTRL module is fully featured for external resistive sensors with a current generator settable between 32µA and 2mA. The only external components required besides the sensors are two resistors, one reference resistor (RTR type ≤0.1%) and one bias resistor. It is preferable if value of these resistor is in the same order as the sensors, for example 1kΩ if PT1000 sensor/s are used. If sensor cables are long an external filter is also preferable.

The external sensors are preferable of the PT type, especially if multiple sensors are used. PT100, PT500, PT1000 sensors are all possible to use. If multiple sensors are used they are preferable of the same type.

Two of the interfaces support 4-wire sensors. These interfaces can be used both for absolute temperature measurements and differential measurements. One interface supports 2-wire sensors and can only be used for absolute measurements.

- The temperature resolution is up to 0.01°C.
- The absolute temperature accuracy of the RTD interfaces is up to ±0.2°C . In order to reach the highest accuracy an external 0.1% or better reference resistor needs to be connected.
- The differential temperature between two sensors can also be directly measured. The accuracy of the differential temperature measurement is up to ±0.1°C.
- The on-chip temperature sensor measures the internal chip temperature with a ±5°C accuracy by default. Calibration will improve the accuracy.
- DC voltage supplied on the analog input pin A_IN can be measured.
- The battery voltage can also be measured.

The use of the complete AD CTRL module is made easy to use by the software library interface. This part of the software is under development.

4 Operating Conditions

Currents flowing into the ASIC are specified as positive values. Currents flowing out of the ASIC are specified as negative values.

4.1 Absolute Maximum Ratings

Note: The values below denote absolute maximum ratings. These ratings are stress ratings only. Functional operation of the device at conditions between maximum operating conditions and absolute maximum ratings is not implied. Exposure to these conditions for extended periods may affect device reliability (e.g. hot carrier degradation, oxide breakdown). Applying conditions above absolute maximum ratings may be destructive to the devices.

The reference voltages are the ground pins VSSx.

No.	Parameter Voltage referred to SGND	Condition	Symbol	Min.	Max.	Unit
1	Supply Voltage VBAT		VBAT_MAX	-0.3	4.0	V
2	Supply Voltage VDD_IO and VDD_TX		VDD3_MAX	-0.3	3.6	V
3	Supply Voltage VDDA, VDDD, VDD_OSC		VDD1P8_MAX	-0.3	1.98	V
4	Voltage at Digital Pins GPIOs, TST, TCK, TDA, NRESET, TXx, MSEQx		VIO_D_MAX	-0.3	VDD3_MAX	V
5	Voltage at Analog Pins HCLKx, LCLKx, BPxx, RTD_x		VIO_A_MAX	-0.3	VDD1P8_MAX	V
6	Voltage for FLASH Test Pin		VHVPAD_MAX	-0.3	13 (tbd.)	V
7	Storage Temperature		TSTORE_MAX	-40	125	°C
8	Junction Temperature		TJ_MAX	-40	125	°C
9	Power Dissipation		PDIS_MAX			mW

4.2 Recommended Operating Conditions

4.2.1 General Parameters

No.	Parameter	Condition	Symbol	Min.	Typ	Max.	Unit
1	Battery Supply Voltage		VBAT	3.1		3.7	V
2	Low Voltage at Digital Pins GPIOs, TST, TCK, TDA, NRESET, TXx, MSEQx		VIO_DH			0.3	V

No.	Parameter	Condition	Symbol	Min.	Typ	Max.	Unit
3	High Voltage at Digital Pins GPIOs, TST, TCK, TDA, NRESET, TXx, MSEQx		VIO_DL	VDDIO-0.3 2)			V
4	Voltage at Pin RTD_x		VRTD_X	150		1550	mV
5	Voltage at Pin A_IN		VAI	150		1550	mV
6	Transducer Exciting Resistance		TX_REXC =f(TX_RPD) 1)		5		Ω
7	Transducer Attenuation Resistance		TX_RATT =f(TX_RPD) 1)		70		Ω
8	Thermal Resistance of QFN package	Junction Ambient JEDEC EIA/JESD51-2	$R_{\theta JA}$		70 (tbd.)		K/W
9	Thermal Resistance of QFN package	Junction Board JEDEC EIA/JESD51-8	$R_{\theta JB}$		7		K/W
10	Junction Temperature		TJ	-40		125	°C
11	Power Dissipation		PDIS		tbd.		mW

1) Recommended resistor configuration by register settings with maximum US energy, minimum electrical radiation and maximum attenuation after excitation.

2) The VDDIO voltage is described in Chapter “Detailed Electrical Parameters -> Power Modules”.

4.2.2 External Components

No.	Device	Description	Symbol / Cond.	Min.	Typ	Max.	Unit
1	BT	Battery Voltage Battery Capacity Battery Resistance	VBT CBT RBT	3.1 (tbd.)	3.3 7 10	3.6	V Ah Ω
2	CB1 CB2	Battery Decoupling Capacitance in parallel	ESR = tbd.		100 10		nF µF
3	CO1 CO2	Buffer Capacitance for Ring-Oscillator Supply (Ceramic 1.8V typ.) in parallel	ESR = tbd.		100 4.7		nF µF
4	CA1 CA2	Buffer Capacitance for Analog Supply (Ceramic 1.8V typ.) in parallel	ESR = tbd.		100 4.7		nF µF
5	CD1 CD2	Buffer Capacitance for Digital Supply (Ceramic 1.8V typ.) in parallel	ESR = tbd.		100 4.7		nF µF
5	CIO1 CIO2	Buffer Capacitance for IO Supply (Ceramic 3V typ.) in parallel	ESR = tbd.		100 4.7		nF µF

No.	Device	Description	Symbol / Cond.	Min.	Typ	Max.	Unit
6	CX1 CX2	Buffer Capacitance for Excitation Supply (Ceramic 3V typ.) in parallel	ESR = tbd.		100 4.7		nF µF
7	RDA	Pull Up Resistance at I2C Interface Data Pin		3 (tbd.)	30	50 (tbd.)	kΩ
8	RCL	Pull Up Resistance at I2C Interface Clock Pin		3 (tbd.)	30	50 (tbd.)	kΩ
9	RP	PWM Filter Resistor (Tau=3.9ms)			39		kΩ
10	CP	PWM Filter Capacitor (Tau=3.9ms)			100		nF
11	RRS	Reset Filter Resistor (optional)			39 (tbd.)		kΩ
12	CRS	Reset Filter Capacitance (optional)			100 (tbd.)		nF
13	RUT	UART Transmitter Resistor ((3V-0.7V)/5mA)			470		Ω
14	RUR	UART Receiver Resistor Pull Up to VDD_IO		1 (tbd.)	3.3 (tbd.)	12 (tbd.)	kΩ
15	RTF	Temperature Interface Filter Resistance			2.7		kΩ
16	CTF	Temperature Interface Filter Capacitance			1		nF
17	CRT	Decoupling Capacitance at pin RTI	ESR = tbd.		10		nF
18	CBT	Cable to Temperature Sensor Wire Length Wire Capacitance Wire Inductance Wire Resistance	 LEW CW LW RW		 tbd. 2 6 0.4	 tbd. 4 20 1.4	 mm nF µH Ω
19	RT1, RT2, RT3	Platinum Thermal Resistances: PT100 TC1 Drift Accuracy Settling Time Operating Time at I=8mA Temperature Range	 RPT100 @ 0°C TC1_100 DR_100 AC_100 TS_100 TOP_100 TR_RT100		 100 38.5 +-0.2 -40	 0.2 +-0.5 200 tbd 1	 Ω %/100°C °C/year Ω ms ms 190
20	RT1, RT2, RT3	Platinum Thermal Resistances: PT500 TC1 Drift Accuracy Settling Time Operating Time at I=8mA Temperature Range	 RPT500 @ 0°C TC1_500 DR_500 AC_500 TS_500 TOP_500 TR_RT500		 500 38.5 tbd. tbd. tbd. -40	 1	 Ω %/100°C °C/year Ω ms ms 190

No.	Device	Description	Symbol / Cond.	Min.	Typ	Max.	Unit
21	RT1, RT2, RT3	Platinum Thermal Resistances: PT1000 TC1 Drift Accuracy Settling Time Operating Time at I=8mA Temperature Range	 RPT1000 @ 0°C TC1_1000 DR_1000 AC_1000 TS_1000 TOP_1000 TR_RT1000		 1000 38.5 tbd. tbd. tbd. -40	 1 	 Ω %/100°C °C/year Ω ms ms 190
22	RTR	Precision Thin Film MELF Reference Resistor MMA 0204 (should be the same value as RT1, RT2 and RT3.) Resistance TC Accuracy	 RTR TC_RTR AC_RTR	 -0.15	 RT1,2,3 0.1	 +0.15	 Ω ppm/°C %
23	RTB	Voltage Shift Resistance at the Bottom at the resister chain. Should be in the order of the temperature sense resistances	 RTB AC_RTR		 Approx. as Rtx 5		 Ω %
24	XT	Transducer Element for Water Applications: Frequency Inductance Capacitance Resistance Parallel Capacitance Quality Factor	 F_W L_W C_W R_W CP_W Q_W	 1	 3.72 10.37 180 14.15 640 18	 4.2	 MHz µH pF Ω pF
25	XT_G	Transducer Element for Gas Applications: Frequency Inductance Capacitance Resistance Parallel Capacitance Quality Factor	 F_G L_G C_G R_G CP_G Q_G	 30	 250 2889 125.8 405 1100 13	 1000	 kHz µH pF Ω pF
26	RXPU	Pull Up Resistance to 3V to charge the Coupling Capacitance CXC for Water Applications			1		kΩ
27	RXPU_G	Pull Up Resistance to tbd. V to charge the Coupling Capacitance CXC_E for Gas Applications			10		kΩ
28	RXPD	Pull Down Resistance to VSS to discharge the Coupling Capacitance CXC for Water Applications			1		kΩ
29	CXC	Coupling Capacitance CXC for Water Applications		1.0	1.8	4.7	nF

No.	Device	Description	Symbol / Cond.	Min.	Typ	Max.	Unit
30	CXC_G	Coupling Capacitance CXC for Gas Applications		10	15	22	nF
31	RXA	Damping resistance to damp the oscillator after the excitation for water application			70		Ω
32	RXA_G	Damping resistance to damp the oscillator after the excitation for gas application			50		kΩ
33	TNX	NMOS Driver Transistor for external excitation for water application FDV301N	VDS,max RDS_on Vth Ciss	- - 0.7 -	- 3.8 0.85 9.5	25 5 1.06	V Ω V pF
34	TNX_G	NMOS Driver Transistor for external excitation for gas application BSS123LT1-D (tbd.)	VDS,max RDS_on Vth Ciss	- - 0.8 -	- 5 - 20	100 6 2.8	V Ω V pF
35	TPX	PMOS Driver Transistor for external Excitation for water application FDV302P	VDS,max RDS_on Vth Ciss	-25 - -1.5 -	- 10.6 -1 11	- 13 -0.65	V Ω V pF
36	VDD_TX_G	Power Supply for external excitation in gas application			70		V
37	RMP	MSEQ Pull Resistance to keep the MSEQ level defined during start up, when the MSEQ pins are high ohmic			100		kΩ
38	CLL	Limiter Capacitance (f_HP=16kHz)	water gas		100 100		pF pF
39	RLL	Limiter Resistor (f_HP= 16kHz)	water gas		20 50		kΩ kΩ
40	DLL	Limiting Schottky-Diode tbd.	Vd @ 100mA		0.4 (tbd.)		V
41	CBPL	Low pass capacitor of the fully differential band pass for 250kHz Crystal			20		pF
42	CBPH	High pass capacitor of the fully differential band pass for 250kHz Crystal			90		pF
43	CXL	Working Capacitance for LCLK oscillator			20 (tbd.)		pF
44	RXL	Operating Point Resistor for LCLK oscillator			10 (tbd.)		MΩ
45	XL	Crystal Frequency for LCLK	f_LCLK		32768		Hz
46	CXH	Working Capacitance for HCLK oscillator			20 (tbd.)		pF

No.	Device	Description	Symbol / Cond.	Min.	Typ	Max.	Unit
47	RXH	Operating Point Resistor for HCLK oscillator			1 (tbd.)		MΩ
48	XH	Crystal Frequency for HCLK	f_HCLK	15.4	16	16.6	MHz
49	RAI	Allowed internal resistance of the external analog signal source				5	kΩ

6 Functional Description of Measurement Modules

6.1 UFO Loop Controller (UL_CTRL)

The UFO loop controller controls the measurement sequence of the ultrasonic measurement. The sequence is controlled by registers, which are accessed by the UFO2 library software. The US measurements can by triggered by software, but can also be triggered by the START_FSM state machine.

6.1.1 Measurement Procedure for Flow Measurements

The following steps **exemplify** what happens during an ultrasonic measurement using measurement mode 1 where the two transducers are connected at TX0 and TX3. This is also illustrated in figure 5.

1. The UFO2 is in maximum power down state.
2. The START-FSM (triggered by the real time clock (RTC)) can be configured to start a measurement. So the CPU decides in advance by software, what the START-FSM has to do: a) temperature measurement, b) US measurement, c) Calibration of AGC or HF_CLK. In case of US measurement the following procedure is done.
3. Option: A calibration of the AG_CTRL is normally done before every measurement.
4. Option: A calibration of the HF_CLK is normally done before every measurement.
5. The transducer on TX0 is excited and the time measurement is started.
6. The US pulse is passing through the flow tube. The receiving AG_CTRL module with the amplifier and band pass filter is powered off until the time SMARK.
7. At SMARK the AG_CTRL module is powered on and after another 3µs the amplifier and the filter are settled.
8. The receiving transducer at TX3 is waiting until the time SSPACE for the arrival of the ultrasonic signal.
9. If the received ultrasonic signal passes the level of PTRIG the signal detection is armed. If the signal does exceed PTRIG until SSPACE expire the time measurement is stopped with a time out.
10. The signal is detected by the module and:

    a) If the number of excitations has not reached the LOOPC value the next excitation at the transducer on TX0 is done. The time measurement is not stopped.

    b) If the number of excitations has reached LOOPC the time measurement is stopped. Another excitation at the transducer on TX0 is not issued.
11. Steps 5 to 10 are repeated but in the other direction with the transducer on TX3 as the sending transducer.
12. The measured transit times are accessible through library functions in double precision. The status from the AG_CTRL calibration and HF_CLK calibration are also accessible.

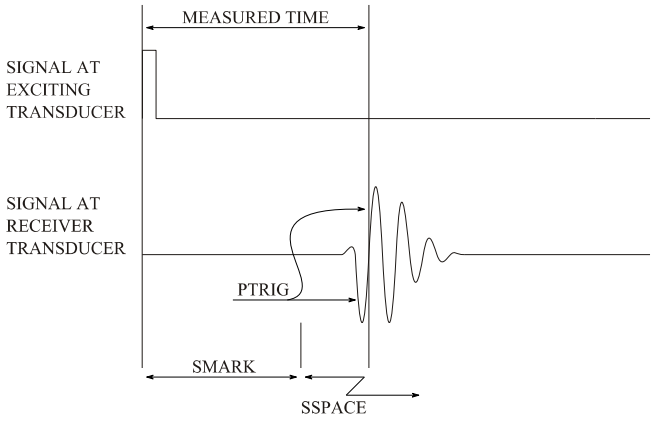


Figure 5: Measurement mode 1 – example with LOOPC = 1

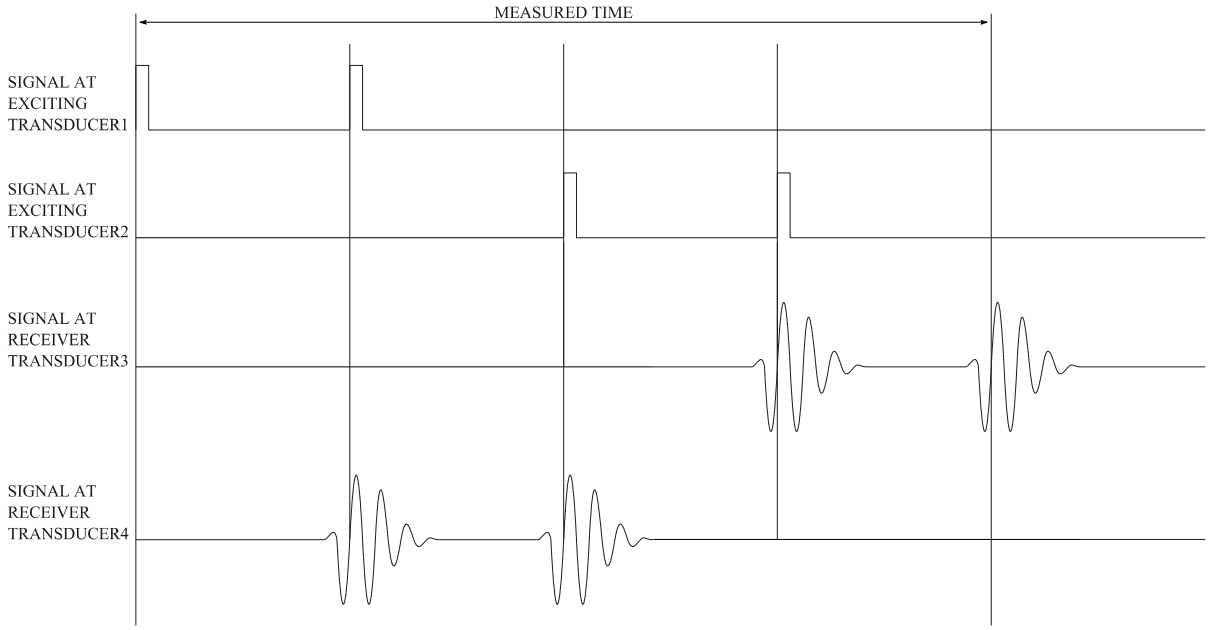


Figure 6: Measurement mode 2 – example with LOOPC = 4

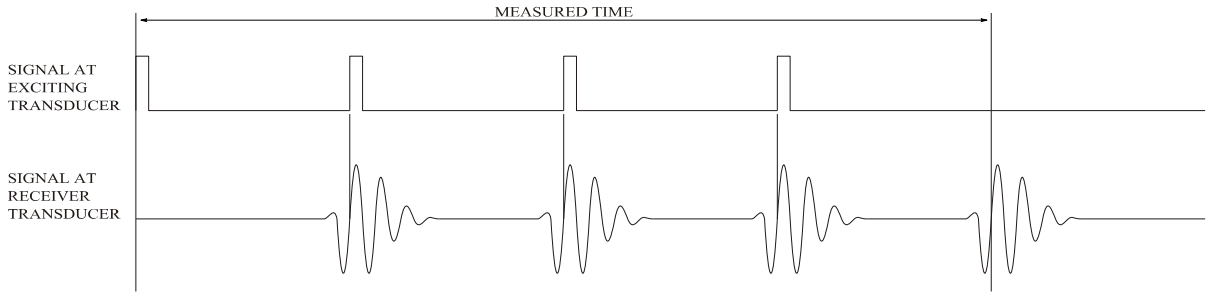


Figure 7: Measurement mode 3 – example with LOOPC = 4

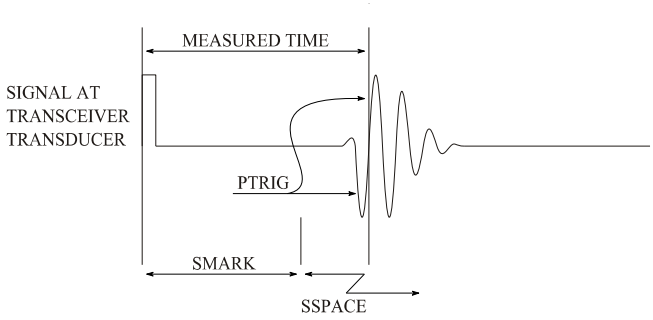


Figure 8: Measurement mode 4 – both excitation and receiving at the same transducer

6.2 Analog to Digital Controller (AD_CTRL)

6.2.1 Temperature Interface RTD

The UFO2 interface supports three external resistive temperature sensors, preferably PT 100, PT500 or PT1000. There are two 4-wire interfaces and one 2-wire interfaces. The 4-wire interfaces can be configured for direct differential measurements or as separate absolute measurements. The 2-wire interface is for absolute measurements. If multiple sensors are used they are preferably of the same type.

To measure the temperature the reference resistance RTR and the bias resistor RTB are necessary. The RTR should be 0.1% or better in order to achieve the highest performance.

6.2.2 Current Generator

The current generator delivers the bias current for the external temperature resistance and reference resistor. It can be controlled between 32µA and 2mA.

6.2.3 Internal Temperature Sensor

An internal temperature sensor delivers the chip temperature information to the ADC.

6.2.4 Battery Sensing

In the ADC the battery voltage can also be measured.

6.2.5 Analog Input

Via the A_IN pins an external analog DC voltage can also be measured by the ADC.

6.3 Automatic Gain Control (AG_CTRL)

6.3.1 Variable Gain Amplifier

The gain of the signal chain is controlled by a two stage amplifier. Stage one is settable, readable and can be calibrated at any chosen moment. The gain of stage one is controlled by the ***R1*** and ***gm1*** registers. Stage two is settable, readable and controlled by the ***R2*** register.

The automatic gain control module sets the gain of stage one in a calibration procedure to fit best the required signal amplitude by using the feedback of the ***PRTIG*** modules. The calibration results are accessible through the library functions.

Gain Stage 1 (TYP)

R1 Setting Code	gm1 [A/V] 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.76	2.12	2.47	2.82	3.17	3.53	3.88	4.23	4.59	4.94	5.29	5.64	6.00	6.35	6.70	7.05
1	2.42	2.90	3.38	3.86	4.35	4.83	5.31	5.80	6.28	6.76	7.25	7.73	8.21	8.70	9.18	9.66
2	3.21	3.86	4.50	5.14	5.78	6.43	7.07	7.71	8.36	9.00	9.64	10.28	10.93	11.57	12.21	12.85
3	3.87	4.64	5.41	6.18	6.96	7.73	8.50	9.28	10.05	10.82	11.60	12.37	13.14	13.92	14.69	15.46
4	4.84	5.80	6.77	7.74	8.71	9.67	10.64	11.61	12.57	13.54	14.51	15.48	16.44	17.41	18.38	19.34
5	5.49	6.59	7.68	8.78	9.88	10.98	12.07	13.17	14.27	15.37	16.46	17.56	18.66	19.76	20.85	21.95
6	6.29	7.54	8.80	10.06	11.31	12.57	13.83	15.09	16.34	17.60	18.86	20.12	21.37	22.63	23.89	25.14
7	6.94	8.33	9.71	11.10	12.49	13.88	15.26	16.65	18.04	19.43	20.81	22.20	23.59	24.98	26.36	27.75
8	8.10	9.72	11.34	12.96	14.58	16.19	17.81	19.43	21.05	22.67	24.29	25.91	27.53	29.15	30.77	32.39
9	8.75	10.50	12.25	14.00	15.75	17.50	19.25	21.00	22.75	24.50	26.25	28.00	29.75	31.50	33.25	35.00
10	9.55	11.46	13.37	15.28	17.18	19.09	21.00	22.91	24.82	26.73	28.64	30.55	32.46	34.37	36.28	38.19
11	10.20	12.24	14.28	16.32	18.36	20.40	22.44	24.48	26.52	28.56	30.60	32.64	34.68	36.72	38.76	40.79
12	11.17	13.40	15.64	17.87	20.11	22.34	24.57	26.81	29.04	31.28	33.51	35.74	37.98	40.21	42.45	44.68
13	11.82	14.19	16.55	18.91	21.28	23.64	26.01	28.37	30.74	33.10	35.46	37.83	40.19	42.56	44.92	47.29
14	12.62	15.14	17.67	20.19	22.72	25.24	27.76	30.29	32.81	35.33	37.86	40.38	42.91	45.43	47.95	50.48
15	13.27	15.93	18.58	21.23	23.89	26.54	29.20	31.85	34.50	37.16	39.81	42.47	45.12	47.78	50.43	53.08
16	14.63	17.56	20.48	23.41	26.33	29.26	32.19	35.11	38.04	40.96	43.89	46.82	49.74	52.67	55.59	58.52
17	15.28	18.34	21.39	24.45	27.51	30.56	33.62	36.67	39.73	42.79	45.84	48.90	51.96	55.01	58.07	61.12
18	16.08	19.29	22.51	25.73	28.94	32.16	35.37	38.59	41.81	45.02	48.24	51.45	54.67	57.88	61.10	64.32
19	16.73	20.08	23.42	26.77	30.11	33.46	36.81	40.15	43.50	46.84	50.19	53.54	56.88	60.23	63.58	66.92
20	17.70	21.24	24.78	28.32	31.86	35.40	38.94	42.48	46.03	49.57	53.11	56.65	60.19	63.73	67.27	70.81
21	18.35	22.02	25.69	29.36	33.04	36.71	40.38	44.05	47.72	51.39	55.06	58.73	62.40	66.07	69.74	73.41
22	19.15	22.98	26.81	30.64	34.47	38.30	42.13	45.96	49.79	53.62	57.45	61.28	65.11	68.94	72.77	76.60
23	19.80	23.76	27.72	31.68	35.64	39.60	43.56	47.53	51.49	55.45	59.41	63.37	67.33	71.29	75.25	79.21
24	20.96	25.16	29.35	33.54	37.74	41.93	46.12	50.31	54.51	58.70	62.89	67.09	71.28	75.47	79.66	83.86
25	21.61	25.94	30.26	34.58	38.91	43.23	47.55	51.88	56.20	60.52	64.84	69.17	73.49	77.81	82.14	86.46
26	22.41	26.90	31.38	35.86	40.34	44.83	49.31	53.79	58.27	62.76	67.24	71.72	76.20	80.69	85.17	89.65
27	23.06	27.68	32.29	36.90	41.51	46.13	50.74	55.35	59.97	64.58	69.19	73.80	78.42	83.03	87.64	92.25
28	24.04	28.84	33.65	38.46	43.26	48.07	52.88	57.69	62.49	67.30	72.11	76.91	81.72	86.53	91.34	96.14
29	24.69	29.62	34.56	39.50	44.44	49.37	54.31	59.25	64.18	69.12	74.06	79.00	83.93	88.87	93.81	98.75
30	25.48	30.58	35.68	40.78	45.87	50.97	56.07	61.16	66.26	71.36	76.45	81.55	86.65	91.74	96.84	101.94
31	26.13	31.36	36.59	41.82	47.04	52.27	57.50	62.72	67.95	73.18	78.40	83.63	88.86	94.09	99.31	104.54

above 8.4 MHz BW
below 8.4 MHz BW
below 4.2 MHz BW

example valid values during GM1 search (R1min=2 & GM1=11 & GMmin=0)

Figure 9: Stage 1 gain settings

Gain Stage 2 (TYP)

R2 Setting Code	gm2 [A/V] 0	1	2	3	4	5	6	7
0	0.64	1.28	1.92	2.56	3.20	3.84	4.49	5.13
1	0.83	1.66	2.49	3.33	4.16	4.99	5.82	6.65
2	1.08	2.15	3.23	4.31	5.38	6.46	7.54	8.62
3	1.27	2.54	3.80	5.07	6.34	7.61	8.87	10.14
4	1.58	3.16	4.74	6.32	7.90	9.49	11.07	12.65
5	1.77	3.54	5.32	7.09	8.86	10.63	12.40	14.17
6	2.02	4.03	6.05	8.07	10.09	12.10	14.12	16.14
7	2.21	4.42	6.62	8.83	11.04	13.25	15.46	17.67
8	2.60	5.19	7.79	10.38	12.98	15.58	18.17	20.77
9	2.79	5.57	8.36	11.15	13.94	16.72	19.51	22.30
10	3.03	6.07	9.10	12.13	15.16	18.20	21.23	24.26
11	3.22	6.45	9.67	12.90	16.12	19.35	22.57	25.79
12	3.54	7.07	10.61	14.14	17.68	21.22	24.75	28.29
13	3.73	7.46	11.18	14.91	18.64	22.37	26.10	29.82
14	3.97	7.95	11.92	15.89	19.87	23.84	27.81	31.79
15	4.17	8.33	12.50	16.66	20.83	24.99	29.16	33.33
16	4.63	9.25	13.88	18.50	23.13	27.75	32.38	37.00
17	4.82	9.64	14.45	19.27	24.09	28.91	33.73	38.55
18	5.06	10.13	15.19	20.26	25.32	30.38	35.45	40.51
19	5.26	10.51	15.77	21.03	26.29	31.54	36.80	42.06
20	5.57	11.13	16.70	22.27	27.83	33.40	38.97	44.54
21	5.76	11.52	17.28	23.04	28.80	34.57	40.33	46.09
22	6.01	12.01	18.02	24.03	30.03	36.04	42.05	48.06
23	6.20	12.40	18.60	24.81	31.01	37.21	43.41	49.61
24	6.58	13.16	19.74	26.31	32.89	39.47	46.05	52.63
25	6.77	13.55	20.32	27.09	33.87	40.64	47.41	54.19
26	7.02	14.04	21.06	28.08	35.10	42.12	49.14	56.16
27	7.22	14.43	21.65	28.86	36.08	43.29	50.51	57.72
28	7.52	15.05	22.57	30.09	37.62	45.14	52.66	60.19
29	7.72	15.44	23.16	30.88	38.60	46.32	54.03	61.75
30	7.97	15.93	23.90	31.86	39.83	47.80	55.76	63.73
31	8.16	16.32	24.49	32.65	40.81	48.97	57.14	65.30

above 8.4 MHz BW
below 8.4 MHz BW
below 4.2 MHz BW

Figure 10: Stage 2 gain settings

6.3.2 Bandpass

There is an internal band pass filter on chip. It is optimized for the D-Flow 4 MHz transducers.

An external RCR filter can be connected via the four Bpxx pins. When using an external filter the signal path through the internal filter can be disabled.

6.3.3 PTRIG

UFO2 features a three level pulse trigger with the ***PTRIG***, ***PTRIG_LOW*** and ***PTRIG_HIGH*** stages. They are all settable and accessible through the library functions.

PTRIG is used to detect the arrival of a signal and arms the signal detection module. It should be set higher than the signal noise and lower than ***PTRIG_LOW***.

PTRIG_LOW is used during the calibration of the variable gain amplifier. During the gain calibration procedure ***R1*** and ***gm1*** of stage one is set so that ***PTRIG_LOW*** is just exceeded. ***PTRIG_LOW*** is set higher than ***PTRIG***. The ***PTRIG_LOW*** status register is updated for every gain calibration and every measurement.

The ***PTRIG_HIGH*** status is included in the status register but it is not used during the gain calibration. The status of ***PTRIG_HIGH*** could for example be used by software to determine if a new gain calibration should be issued. ***PTRIG_HIGH*** is preferably set higher that ***PTRIG_LOW***.

6.4 Transducer Control (TX_CTRL)

As shown in the typical application diagram, the external transducers can be connected with several options to the pins TX0, TX1, TX2, TX3 and MSEQ0, MSEQ1, MSEQ2 and MSEQ3.

Each transducer pin TX0 to TX3 has a driving capability and a sensing capability. Sensing and exciting is not done at the same time but can be done at the same pin, such in for example distance measurements using only one transducer. In flow meter applications one TX channel is sensing and another one exciting.

Connect Option 1:
The transducer pins TX0 to TX3 are driver and send pins and two transducers can be connected in a fully differential way to the UFO2 ASIC. This corresponds to **measurement mode 3**.

Connect Option 2:
Up to four transducers can be connected to the four transducer driver pins TX0 to TX3. The driver pins are sensing input and exciting output. This corresponds to **measurement modes 1, 2 and 4**.

Connect Option 3:
Up to four transducers can be driven with external transistor TNX. A coupling capacitance CXC and an attenuation resistor RXA is needed. The transistor is driven by the MSEQ pin and the sensing input is the corresponding TX pin.

Connect Option 4:
The same as option 3, but here two transducers can be stimulated in a fully differential way by the switch SXW. The switch is controlled by the pins MSEQ2 and MSEQ3. A PMOS driver transistor needs to be controlled by the MSEQ pins, because the switch is not allowed to have negative voltages. The polarity of the pins MSEQ0 and MSEQ1 is inverted, due to the PMOS transistor.

Connect Option 5:
It is the same as option 3, but a higher excitation voltage VDD_TX_G can be applied, which is necessary for gas application.

6.4.1 Measurement Sequence Logic (MSEQ)

The MSEQ module delivers the control signals for the transducer driver, MSEQ pins and input multiplexer.

6.5 Power Supplies

The power supplies **VDDx** are never switched off, not to lose charge of the external buffer capacitance. Therefore the supply current only needs to be very small.

7 Functional Description of Digital Modules

7.1 Block Diagram

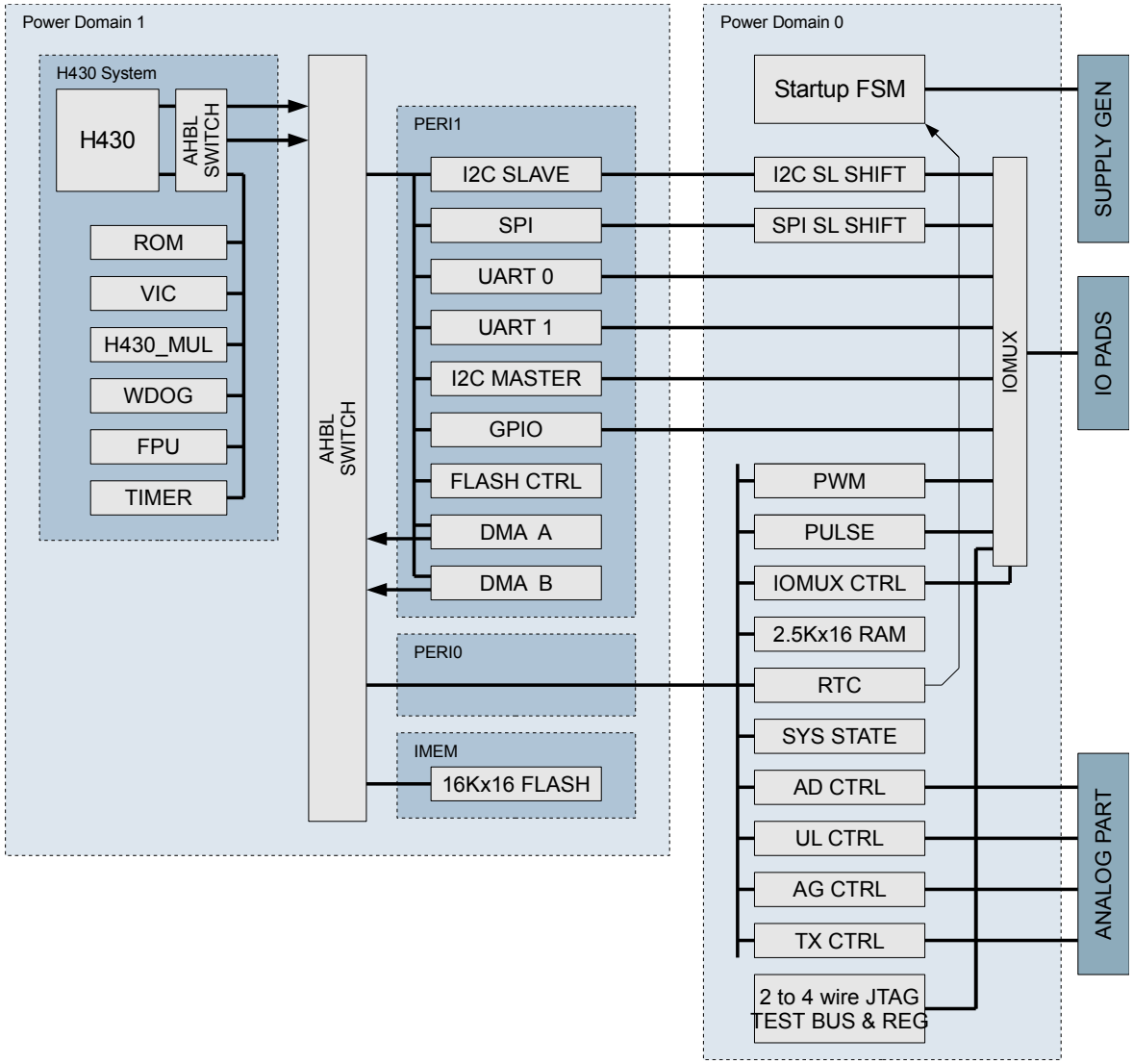


Figure 11: Digital Modules Block Diagram

7.2 General Description

The digital part consists out of two power domains: power domain 0 and power domain 1. The power domain 0 contains all logic, which can not be powered down during the measurement, as the RAM, some interfaces, analogue controller, etc. The power domain 1 can be physically disconnected from the power supply VDDD.

7.3 Memory map

base address	instance size	module abbreviation	instance abbreviation	module name
0x0000	0x0400	WB	WB_CPU_MODS	CPU module bus
0x0000	0x0080	ROM	ROM	Startup ROM
0x0080	0x0080	SWTIMER	SWTIMER	Timer Module
0x0100	0x0080	H430_MUL	H430_MUL	H430 Multiplier
0x0180	0x0080	FPU	FPU	Floating Point Unit
0x0200	0x0080	VIC	VIC	Vector Interrupt Controller Module
0x0280	0x0080	WDOG	WDOG1	Domain 1 Watchdog Module
0x0400	0x0400	WB	WB_DM1_PERI	domain 1 peripheral bus
0x0400	0x0040	SPI	SPI	SPI Master and Slave Module
0x0440	0x0040	UART	UART0	UART Module (Instance 0)
0x0480	0x0040	UART	UART1	UART Module (Instance 1)
0x04C0	0x0040	I2C	I2C	I2C Master Module
0x0500	0x0040	I2C_SLAVE	I2C_SLAVE	I2C Slave Module
0x0580	0x0040	FLASH_CTRL	FLASH_CTRL	FLASH Control Module
0x05C0	0x0040	DMA	DMA_A0	DMA Channel A Subchannel 0
0x0600	0x0040	DMA	DMA_A1	DMA Channel A Subchannel 1
0x0640	0x0040	DMA	DMA_B0	DMA Channel B Subchannel 0
0x0680	0x0040	DMA	DMA_B1	DMA Channel B Subchannel 1
0x0800	0x0800	WB	WB_DM0_PERI	domain 0 peripheral bus
0x0800	0x0080	IOMUX_CTRL	IOMUX_CTRL	IO Control Module
0x0880	0x0080	SYS_STATE	SYS_STATE	System State Module
0x0900	0x0080	RTC	RTC	Real Time Clock Module
0x0980	0x0080	PULSE	PULSE	Pulse Interface
0x0A00	0x0080	PWM	PWM	PWM Module
0x0C80	0x0080	GPIO	GPIO	GPIO Module
0x0D00	0x0080	WDOG	WDOG0L	Domain 0 LCLK Watchdog Module
0x0D80	0x0080	WDOG	WDOG0H	Domain 0 HCLK Watchdog Module
0x1000	0x1400	DMEM	DMEM	SRAM Memory
0x8000	0x8000	IMEM	IMEM	Instruction Memory

7.4 Memory Blocks

- 1 x 32K Byte FLASH
- 1 x 5K Byte SRAM (always powered)

7.5 H430 (16-bit CPU)

7.5.1 Features

- Opcode compatible to TI's MSP430.
- Harvard architecture and AHBL data and instruction bus interfaces.
- RISC architecture with 27 instructions and 7 addressing modes.
- Orthogonal architecture: every instruction usable with every addressing mode.
- Full register access including program counter, status registers, and stack pointer.
- 16 x 16-bit register.
- 64 KByte linear address space.
- 16-bit native data bus width.
- Constant generator provides six most used immediate values and reduces code size.
- Direct memory-to-memory transfers without intermediate register holding.
- Word and byte addressing and instruction formats.
- IAR development IDE compatible JTAG debug interface.
- Several C compilers are available.

7.5.2 Structure

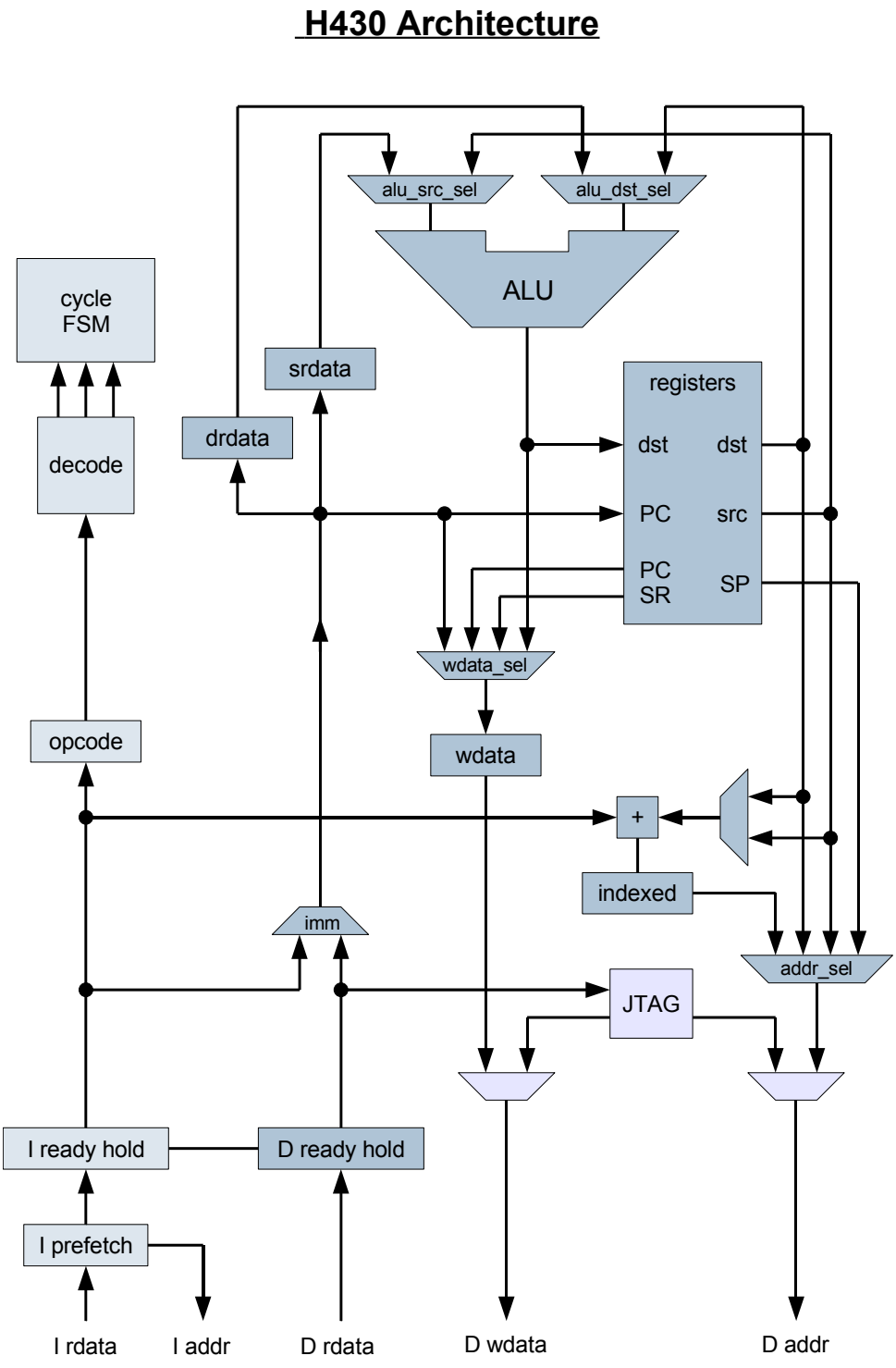


Figure 12: CPU Structure

7.5.3 Interrupts

The embedded H430 IP core does not contain a primary interrupt controller. It has only a IRQ request signal and an address, pointing to a vector table in memory, which contains addresses of the interrupt handlers (see “IRQ System Specification”).

Therefore the H430 IP does not support a fixed number of interrupts. Any number fitting reasonable in the 64k memory range is supported.

All interrupts can be enabled or disabled with the GIE bit in the status register.

Handling an interrupt (other than RESET) consists of:

- Push PC on stack.
- Push SR on stack.
- Choose the highest priority interrupt to service.
- If there are multiple possible sources, leave them for software to poll.
- Clear the SR, which disables interrupts and power-saving.
- Fetch the interrupt vector into the PC
- Start executing the interrupt handler

A reset is similar, but doesn’t save any state.

You can nest interrupt handlers by disabling the current source and setting the GIE bit back to 1.

7.5.4 Byte and Word Issues

The H430 is byte-addressed, and little-endian. Word operands must be located at even addresses.

Most instructions have a byte/word bit, which selects the operand size. Appending ".b" to an instruction makes it a byte operation. Appending ".w" to an instruction, to make it a word operation, is also legal. However, since it is also the default behavior, if you add nothing, it is generally omitted.

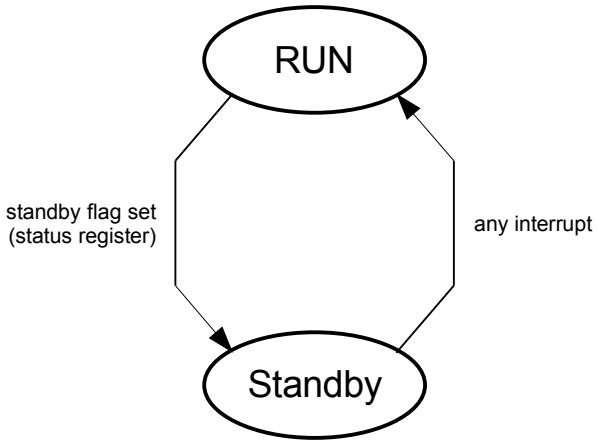
A byte instruction with a register destination clears the high 8 bits of the register to 0. Thus, the following would clear the top byte of the register, leaving the lower byte unchanged:

```
mov.b Rn,Rn
```

Mostly the on-chip peripherals supports only one bus size, e.g. the data width of the processor. These peripherals must be accesses only with the supported access mode and with correct alignment. Any other access may produce an undefined behavior.

When performing a word access, address bit 0 is undefined and has to be ignored.

7.5.5 CPU States



The CPU supports the following states:

RUN	● normal operation of the CPU ● the CPU accesses program storage (e.g. Flash) and RAM ● the CPU returns to RUN state on any interrupt
STANDBY	● the CPU is halted ● the STANDBY state is entered when setting standby flag (CPUOFF) in status register ● the CPU does not access program storage or RAM ● the CPU returns to RUN state on any interrupt

After setting the standby bit in the CPU status register the following instruction will be executed, then standby mode will be entered. A good idea is to use the following sequence to ensure a later wakeup.

- BIS #0x10, SR sets standby flag
- BIS #0x08, SR enables interrupts for wake-up

7.5.6 CPU Registers

The processor has 16 16-bit registers, although only 12 of them are truly general purpose. The first four have dedicated uses:

- R0 (PC) is the program counter. You can jump by assigning to r0, and immediate constants are fetched from the instruction stream using the post-increment addressing mode on r0. The PC value has to be even in any case.
- R1 (SP) is the stack pointer. This is used by call and push instructions, and by interrupt handling. The stack is handled using a pre-decrement, post-increment scheme. The SP value has to be even in any case.
- R2 (SR) is the status register. Its bits are assigned as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved							V	n/a	n/a	OSC OFF	CPU OFF	GIE	N	Z	C

OSCOFF (oscillator off), and CPUOFF are used to enter low-power states. OSCOFF may not be evaluated by the system if other clock controllers are implemented

GIE is the global interrupt enable. Turning off this bit masks interrupts. (NOTE: it may be delayed by 1 cycle, so an interrupt may be taken after the instruction after GIE is cleared. Add a NOP or clear GIE one instruction earlier than your "critical section".)

N, Z, C and V are the usual processor status bits set on instruction execution.

- R3, If specified as source, its value is 0. If specified as a destination, the value is discarded.

Six commonly-used constants are generated with the constant generator registers R2 and R3, without requiring an additional 16-bit word of program code. This is one of the important features of the H430 instruction set, allowing it to achieve a high level of code density, and a flexible instruction set.

These constant registers can provide the numbers -1, 1, 2, 4 or 8. So, for example, the "clr x" is actually emulated by the instruction "mov #0,x". The constant "0" is taken from the constant register r3. The assembler understands both "clr x" and "mov #0,x", and produces the same code for either.

The constants are selected with the source-register addressing modes (As):

Register	As	Constant	Remarks
R2	00	----	Register Mode
R2	01	(0)	Absolute Address Mode

R2	10	0x0004h	+4, bit processing
R2	11	0x0008h	+8, bit processing
R3	00	0x0000h	0, word processing
R3	01	0x0001h	+1
R3	10	0x0002h	+2, bit processing
R3	11	0xFFFFh	-1, word processing

The constant generator advantages are:

- No special instructions required
- No additional code word for the six constants
- No code memory access required to retrieve the constant

The assembler uses the constant generator automatically if one of the six constants is used as an immediate source operand.

7.5.7 Addressing Modes

The available H430 instruction addressing modes have at most two operands, a source and a destination.

All instructions are 16 bits long, followed by at most two optional offsets words, one for each of the source and the destination.

The source operand is specified with 2 addressing mode bits (As):

As	Mnemonic	Remarks
00	Rn	Register direct
01	X(Rn)	Register indexed
10	@Rn	Register indirect
11	@Rn+	Register indirect with post-increment

The destination operand is specified with 1 addressing mode bit (Ad):

Ad	Mnemonic	Remarks
0	Rm	Register direct
1	Y(Rm)	Register indexed

The only addressing mode that uses an extension word is the indexed mode.

The destination operand in a two-operand instruction has only one addressing mode bit, which selects either register direct or indexed. Register indirect can obviously be faked up with a zero index.

When r0 (the program counter) is used as a base address, indexed mode provides PC-relative addressing. This is, in fact, the usual way that the H430 assembler accesses operands when a label is referred to.

@r0 just specifies the following instruction word, but @r0+ specifies that word and skips over it. In other word, an immediate constant! You can just write #1234 and the assembler will specify the addressing mode properly.

r1, the stack pointer, can be used with any addressing mode, but @r1+ always increments by 2 bytes, even on a byte access.

7.5.7.1 Register Direct

Assembler Code	MOV R10,R11
Length	One or two words
Operation	Move the content of R10 to R11. R10 is not affected.
Comment	Valid for source and destination
Note	The data in the register can be accessed using word or byte instructions. If byte instructions are used, the high byte is always 0 in the result. The status bits are handled according to the result of the byte instruction.

7.5.7.2 Register Indexed

Assembler Code	MOV 2(R5),6(R6)
Length	Two or three words
Operation	Move the contents of the source address (contents of R5 + 2) to the destination address (contents of R6 + 6). The source and destination registers (R5 and R6) are not affected. In indexed mode, the program counter is incremented automatically so that program execution continues with the next instruction.
Comment	Valid for source and destination

7.5.7.3 Register Indirect

Assembler Code	MOV @R10,0(R11)
Length	One or two words
Operation	Move the contents of the source address (contents of R10) to the destination address (contents of R11). The registers are not modified.
Comment	Valid only for source operand. The substitute for destination operand is 0(Rd).

7.5.7.4 Register Indirect with post-increment

Assembler Code	MOV @R10+,0(R11)
Length	One or two words
Operation	Move the contents of the source address (contents of R10) to the destination address (contents of R11). Register R10 is incremented by 1 for a byte operation, or 2 for a word operation after the fetch; it points to the next address without any overhead. This is useful for table processing.

Comment	Valid only for source operand. The substitute for destination operand is 0(Rd) plus second instruction INCD Rd.

7.5.8 Instruction Set

The complete H430 instruction set consists of 27 core instructions and 24 emulated instructions. The core instructions are instructions that have unique op-codes decoded by the CPU. The emulated instructions are instructions that make code easier to write and read, but do not have op-codes themselves, instead they are replaced automatically by the assembler with an equivalent core instruction. There is no code or performance penalty for using emulated instruction.

All instructions are 16 bits long, and there are only three instruction formats:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Dual-operand	Opc				S-reg				Ad	B	As		D-reg			
Single-operand	0	0	0	1	0	0	Opc			B	As		D-reg			
Jump	0	0	1	Condition			PC offset (10 bit)									

All single-operand and dual-operand instructions can be byte or word instructions by using .B or .W extensions. Byte instructions are used to access byte data or byte peripherals. Word instructions are used to access word data or word peripherals. If no extension is used, the instruction is a word instruction.

The source and destination of an instruction are defined by the following fields:

src	The source operand defined by As and S-reg
dst	The destination operand defined by Ad and D-reg
As	The addressing bits responsible for the addressing mode used for the source (src)
S-reg	The working register used for the source (src)
Ad	The addressing bits responsible for the addressing mode used for the destination (dst)
D-reg	The working register used for the destination (dst)
B	Byte operation: 0: word operation 1: byte operation

7.5.8.1 Dual-operand instructions

These basically perform dst = src op dst operations. However, MOV doesn't fetch the destination, and CMP and BIT do not write to the destination. All are valid in their 8 and 16 bit forms.

Opc	Mnemonic	S-Reg, D-Reg	Operation	Status Bits				Remark
				V	N	Z	C	

0100	MOV(.B)	src, dst	dst = src	-	-	-	-	
0101	ADD(.B)	src, dst	dst += src	*	*	*	*	
0110	ADDC(.B)	src, dst	dst += src + C	*	*	*	*	
1000	SUB(.B)	src, dst	dst += ~src + 1	*	*	*	*	
0111	SUBC(.B)	src, dst	dst += ~src + C	*	*	*	*	
1001	CMP(.B)	src, dst	dst - src	*	*	*	*	Sets status only; the destination is not written.
1010	DADD(.B)	src, dst	dst += src + C, BCD	0	*	*	*	
1011	BIT(.B)	src, dst	dst & src	0	*	*	*	Sets status only; the destination is not written.
1100	BIC(.B)	src, dst	dst &= ~src	-	-	-	-	The status flags are NOT set.
1101	BIS(.B)	src, dst	dst |= src	-	-	-	-	The status flags are NOT set.
1110	XOR(.B)	src, dst	dst ^= src	*	*	*	*	
1111	AND(.B)	src, dst	dst &= src	0	*	*	*	

* The status bit is affected

− The status bit is not affected

0 The status bit is cleared

1 The status bit is set

7.5.8.2 Single-operand instructions

The status flags are set by RRA, RRC, SXT, and RETI.

The status flags are NOT set by PUSH, SWPB, and CALL.

Opc	Mnemonic	S-Reg, D-Reg	Operation	Status Bits				Remark
				V	N	Z	C	
000	RRC(.B)	dst	C → MSB → ... → LSB → C	0	*	*	*	9-bit rotate right through carry. Clear the carry bit beforehand to do a logical right shift.
010	RRA(.B)	dst	MSB → MSB →... LSB → C	0	*	*	*	Badly named, this is an 8-bit arith-metic right shift.
100	PUSH(.B)	src	SP − 2 → SP src → @SP	-	-	-	-	Push operand on stack. Push byte decrements SP by 2.
001	SWPB	dst	Swap bytes	-	-	-	-	The destination operand high and low bytes are exchanged. This has no byte form.
101	CALL	src	SP − 2 → SP PC+2 → @SP src → PC	-	-	-	-	Fetch operand, push PC, then assign operand value to PC. Note: the immediate form is the most commonly used. There is no easy way to perform a PC-relative call; the PC-relative addressing mode fetches a

								word and uses it as an absolute address. This has no byte form.
110	RETI		TOS → SR SP + 2 → SP TOS → PC SP + 2 → SP	*	*	*	*	Pop SP, then pop PC. Note: The CPUOFF flag will not be stored to stack on interrupt entry, so the CPU will NOT return to low-power mode it was previously in.
011	SXT	dst	Bit 7 → Bit 8.....Bit 15	0	*	*	*	Sign extend 8 bits to 16. No byte form.

* The status bit is affected

− The status bit is not affected

0 The status bit is cleared

1 The status bit is set

7.5.8.3 Emulated instructions

There are a number of zero- and one-operand pseudo-operations that can be built from these two-operand forms. These are usually referred to as "emulated" instructions:

NOP	MOV r3,r3	Any register from r3 to r15 would do the same thing. Note: that other forms of a NOP instruction can be constructed as emulated instructions, which take different numbers of cycles to execute. These can sometimes be useful in constructing accurate timing patterns in software.
POP dst	MOV @SP+,dst	
BR dst	MOV dst,PC	Branch and return can be done by moving to PC (r0)
RET	MOV @SP+,PC	
CLRC	BIC #1,SR	The constants were chosen to make status register (r2) twiddling efficient
SETC	BIS #1,SR	
CLRZ	BIC #2,SR	
SETZ	BIS #2,SR	
CLRN	BIC #4,SR	
SETN	BIS #4,SR	
DINT	BIC #8,SR	
EINT	BIC #8,SR	
RLA(.B) dst	ADD(.B) dst,dst	Shift and rotate left is done with add
RLC(.B) dst	ADDC(.B) dst,dst	
INV(.B) dst	XOR(.B) #-1,dst	Some common one-operand instructions
CLR(.B) dst	MOV(.B) #0,dst	
TST(.B) dst	CMP(.B) #0,dst	
DEC(.B) dst	SUB(.B) #1,dst	Increment and decrement (by one or two)
DECD(.B) dst	SUB(.B) #2,dst	
INC(.B) dst	ADD(.B) #1,dst	

INCD(.B) dst	ADD(.B) #2,dst	
ADC(.B) dst	ADDC(.B) #0,dst	Increment and decrement carry.
DADC(.B) dst	DADD(.B) #0,dst	
SBC(.B) dst	SUBC(.B) #0,dst	

7.5.8.4 Relative jumps

Conditional jumps support program branching relative to the PC and do not affect the status bits. The possible jump range is from − 511 to +512 words relative to the PC value at the jump instruction. The 10-bit program-counter offset is treated as a signed 10-bit value that is doubled and added to the program counter:

PCnew = PCold + 2 + PCoffset × 2

Opc	Mnemonic	Jump Condition
000	JNE/JNZ	Z == 0
001	JEQ/JZ	Z == 1
010	JNC/JLO	C == 0
011	JC/JHS	C == 1
100	JN	N == 1
101	JGE	N == V
110	JL	N != V
111	JMP	unconditionally

7.5.8.5 Instruction Timing

command type	operation	cycles (dreg != PC)	cycles (dreg == PC)
MOV	sreg -> dreg	1	2
DOUBLE	sreg x dreg -> dreg	1	2
MOV	sreg -> Y(dreg)	2	
DOUBLE	sreg x Y(dreg) -> Y(dreg)	4	
MOV	@sreg -> dreg	3	4
DOUBLE	@sreg x dreg -> dreg	3	4
MOV	@sreg -> Y(dreg)	3	
DOUBLE	@sreg x Y(dreg) -> Y(dreg)	5	
MOV	#N -> dreg	2	3
DOUBLE	#N x dreg -> dreg	2	3
MOV	@sreg+ -> dreg	3	4
DOUBLE	@sreg+ x dreg -> dreg	3	4
MOV	#N -> Y(dreg)	3	
DOUBLE	#N x Y(dreg) -> Y(dreg)	5	
MOV	@sreg+ -> Y(dreg)	3	
DOUBLE	@sreg+ x Y(dreg) -> Y(dreg)	5	
MOV	X(sreg) -> dreg	4	5
DOUBLE	X(sreg) x dreg -> dreg	4	5
MOV	X(sreg) -> Y(dreg)	3	
DOUBLE	X(sreg) x Y(dreg) -> Y(dreg)	5	
SINGLE	dreg	1	2
SINGLE	@dreg	3	
SINGLE	#N	2	
SINGLE	@dreg+	3	
SINGLE	Y(dreg)	4	
JUMP		2	
RETI		4	
IRQE		3	
PUSH	reg	1	
PUSH	@reg	2	
PUSH	#N	2	
PUSH	@reg+	2	
PUSH	X(reg)	3	
CALL	reg	2	
CALL	@reg	3	
CALL	#N	3	
CALL	@reg+	3	
CALL	X(reg)	4	

notes:
SINGLE includes RRC, RRA, SWPB and SXT
DOUBLE includes all double operand instructions except MOV

7.5.9 JTAG Debug Interface

7.5.9.1 Features

- debugging support

- 4-wire standard JTAG interface
- 3 hardware break-point triggers
- stepping in IDE possible (single-step, step-in, step-out, run-to-cursor)
- IAR can be use as debug IDE
- compatible with Olimex Parallel Port adapter and JTAG adapter supplied by D-Flow

7.5.9.2 Debugging

The H430 embedded break-point logic provides the following features:

- 3 break-point triggers
- each trigger can match a separate address or data bus value
- a trigger value compare mask can be defined
- trigger can match a greater, smaller, equal or non equal value
- trigger can be configured for read / write or instruction fetch / non instruction fetch bus cycles
- triggers can be combined (trigger dependency)
- all breakpoints can be used for stepping and run-stop a program

7.6 H430 Multiplier

7.6.1 Description

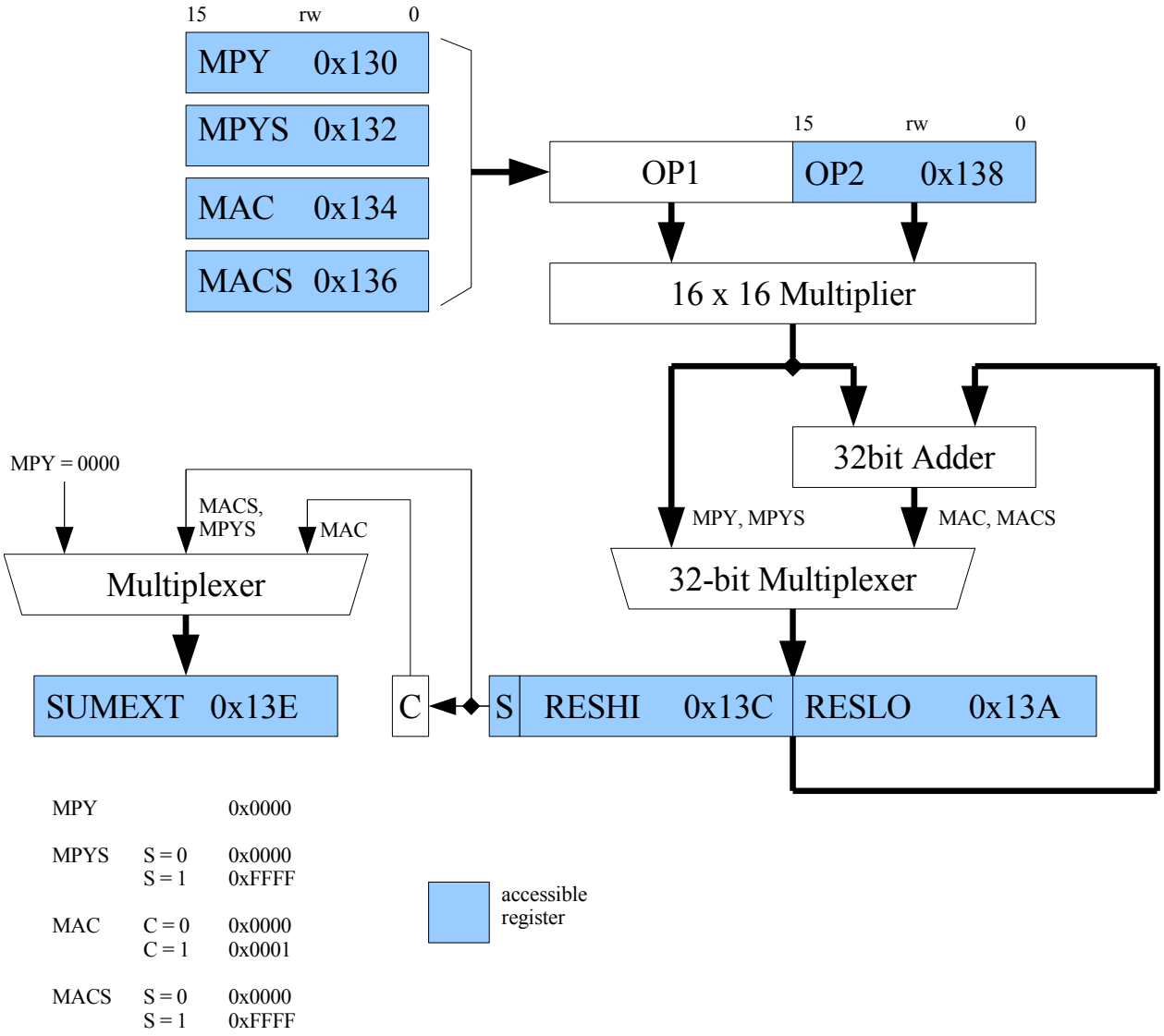


Figure 13: H430 Peripheral Multiplier

The hardware multiplier is a memory mapped peripheral (at a fixed address range from 0x130h to 0x13Fh). It can be accessed by CPU with full support of common compilers. Though the hardware architecture is different, the unit is fully compatible with the MPY16 multiplier unit in chips of the MSP430 family, providing the same interface, software support, and arithmetic results.

The barrel shifter extension allows to shift either the 32/33 bit result of the multiply/MAC operation or any other 32 bit value by a variable number of bits. This unit can be accessed by a normal register interface.

7.6.2 Features

- unsigned/signed multiplication (MPY / MPYS)
- unsigned/signed MAC (multiply and accumulate) operation (MAC / MACS) using the old result and adding the new product
- 16*16, 8*16, 16*8, and 8*8 bit input data width
- 32/33 bit output data width
- no CPU wait states (no NOP required)

7.6.3 Register Interface

The type of operation to be performed is selected by writing the first operand to one of the following four registers. Writing the first operand does not start the operation. The first operand (and thus the type of operation) may remain constant for more than one operation. Writing the second operand starts the operation.

0x30		MPY		
bits	**name**	**default**	**rw**	**description**
15..0	op1	0	rw	first operand for unsigned multiplication

0x32		MPYS		
bits	**name**	**default**	**rw**	**description**
15..0	op1	0	rw	first operand for signed multiplication

0x34		MAC		
bits	**name**	**default**	**rw**	**description**
15..0	op1	0	rw	first operand for unsigned MAC operation (multiply and accumulate)

0x36		MACS		
bits	**name**	**default**	**rw**	**description**
15..0	op1	0	rw	first operand for signed MAC operation (multiply and accumulate)

The four registers for op1 are aliases for one register, so they always read the same value.

Writing the second operand to the following register starts the previously selected operation.

0x38		OP2		
bits	**name**	**default**	**rw**	**description**
15..0	op2	0	rw	second operand (writing starts the operation)

The operand registers of the hardware multiplier behave like the CPU's working registers R0 to R15 when modified in byte mode: the upper byte is cleared in this case. This allows for any combination of 8-bit and 16-bit multiplications. The upper byte cannot be addressed.

The results can be read from the following three registers. For signed operations, results are provided in 2's complement format. The sum extension register SUMEXT allows calculations with results exceeding the 32-bit range. This read-only register holds the most significant part of the result (bits 32 and higher). The register simplifies multiple word operations, because straightforward additions can be performed without conditional jumps.

0x3A		RESLO		
bits	**name**	**default**	**rw**	**description**
15..0	sum_lo	0	rw	bit 15..0 of the result **The high byte cannot be accessed with byte instructions.**

0x3C		RESHI		
bits	**name**	**default**	**rw**	**description**
15..0	sum_hi	0	rw	bit 31..16 of the result for signed multiplication, bit 31 (bit 15 of RESHI) represents the sign bit. for signed MAC operation, the sign bit of the 33 bit result must be read from the SUMEXT register. **The high byte cannot be accessed with byte instructions.**

0x3E		SUMEXT		
bits	**name**	**default**	**rw**	**description**
15:0	sum_ext	0	r	for unsigned multiplication: 0 for signed multiplication: contains the extended sign of the result 0x0000 result was positive or zero 0xFFFF result was negative for unsigned MAC operation: contains the carry of the result 0x0000 result had no carry 0x0001 result had carry for signed MAC operation: contains the extended sign of the result 0x0000 result was positive or zero 0xFFFF result was negative

While all registers can be read back, the operation specified by the first operand address is not recoverable by an interrupt handler. Thus, it is not possible to context-switch the multiplier unless you add some sort of wrapper software (locking or shadow registers) around it.

The hardware multiplier IP core contains an additional readable mode register which contains the last mode:

0x40		LAST_MODE		
bits	**name**	**default**	**rw**	**description**
1..0	last_mode	0	r	last mode of multiply/MAC unit: 0x0 = MPY 0x1 = MPYS 0x2 = MAC 0x3 = MACS

7.7 Vector Interrupt Controller Module

7.7.1 Description

The IRQ Controller IP Core is the primary IRQ controller in a two stage hierarchical interrupt system.

Each of the peripheral modules contains a secondary IRQ controller that processes local events and generates a single interrupt to the primary IRQ controller. These interrupts are level based. The secondary controller contains an IRQ mask and provides the IRQ number for the highest priority interrupt.

The primary IRQ controller combines the level interrupts from all peripheral modules of the SOC system and together with its own IRQ mask it generates an IRQ request to the CPU. Together with the IRQ vector number and the base address of the IRQ vector table an irq pointer is provided to the CPU.

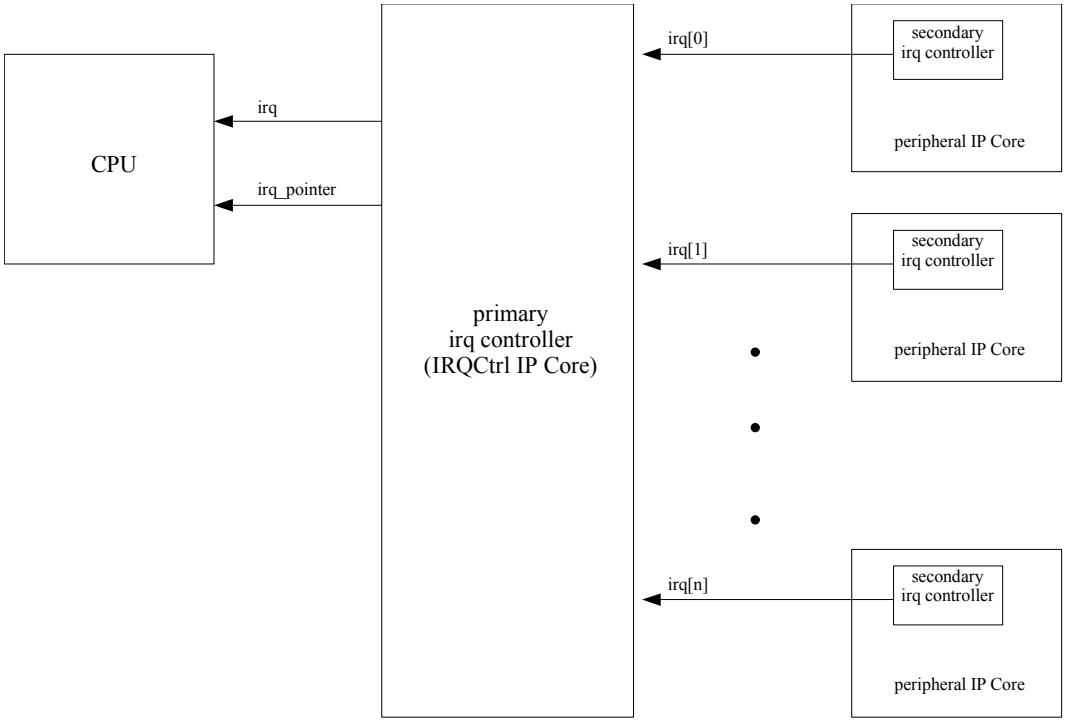


Figure 14: IRQ System

7.7.2 Structure

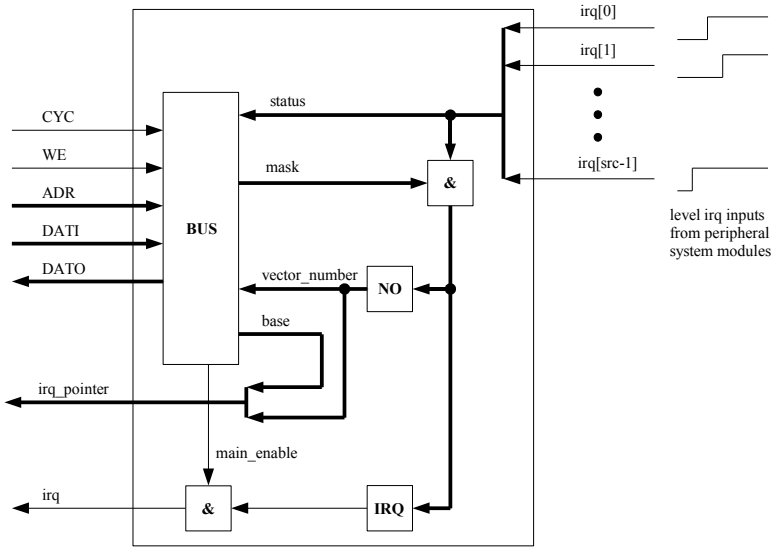


Figure 15: VIC Module Structure

7.7.3 Features

- level irq inputs
- irq number for fast irq processing
- main irq enable to enable or disable all irqs
- irq base address for irq vector table in memory
- prioritized irq sources where irq 0 has highest priority
- vector based interrupt enable and disable
- nested irq support
- input signal reorder config

7.7.4 Register Interface

0x00		TABLE_BASE		
bits	**name**	**default**	**rw**	**description**
15:0	base	0	rw	base address of vector table in memory

0x02		TABLE_TYPE		
bits	**name**	**default**	**rw**	**description**
0	type	0	rw	auto combine vector number and table base to create vector number related CPU interrupt pointer. 0: base value is combined with vector number to be used as CPU interrupt pointer (an interrupt service routine per module) 1: base value is directly used as CPU interrupt pointer (one common interrupt service routine)

0x04		MAIN_ENABLE		
bits	**name**	**default**	**rw**	**description**
0	enable	1	rw	main interrupt enable / disable 1: enabled 0: disabled

0x06		SIG_STATUS0		
bits	**name**	**default**	**rw**	**description**
0	fpu	0	r	Floating Point Module
1	swtimer	0	r	Timer Module
2	spi	0	r	SPI Module
3	uart_0	0	r	UART Module 0
4	uart_1	0	r	UART Module 1
5	i2c	0	r	I2C Master Module
6	i2c_slave	0	r	I2C Slave Module
7	gpio	0	r	GPIO Module
8	sys_state	0	r	System State Module
9	dma_a0	0	r	DMA channel A 0
10	dma_a1	0	r	DMA channel A 1
11	dma_b0	0	r	DMA channel B 0
12	dma_b1	0	r	DMA channel B 1
13	ad_ctrl	0	r	AD_CTRL Module
14	ag_ctrl	0	r	AG_CTRL Module
15	ul_ctrl	0	r	UL_CTRL Module

0x08		SIG_STATUS1		
bits	**name**	**default**	**rw**	**description**
0	pulse	0	r	PULSE Module

0x10		PRIO_IN_SEL_0_1		
bits	**name**	**default**	**rw**	**description**
4:0	sel_0	0	rw	select an IRQ signal for priority 0 input
12:8	sel_1	1	rw	select an IRQ signal for priority 1 input

0x12		PRIO_IN_SEL_2_3		
bits	**name**	**default**	**rw**	**description**
4:0	sel_2	2	rw	select an IRQ signal for priority 2 input
12:8	sel_3	3	rw	select an IRQ signal for priority 3 input

0x14		PRIO_IN_SEL_4_5		
bits	**name**	**default**	**rw**	**description**
4:0	sel_4	4	rw	select an IRQ signal for priority 4 input
12:8	sel_5	5	rw	select an IRQ signal for priority 5 input

0x16		PRIO_IN_SEL_6_7		
bits	**name**	**default**	**rw**	**description**
4:0	sel_6	6	rw	select an IRQ signal for priority 6 input
12:8	sel_7	7	rw	select an IRQ signal for priority 7 input

0x18		PRIO_IN_SEL_8_9		
bits	**name**	**default**	**rw**	**description**
4:0	sel_8	8	rw	select an IRQ signal for priority 8 input
12:8	sel_9	9	rw	select an IRQ signal for priority 9 input

0x1A		PRIO_IN_SEL_10_11		
bits	**name**	**default**	**rw**	**description**
4:0	sel_10	10	rw	select an IRQ signal for priority 10 input
12:8	sel_11	11	rw	select an IRQ signal for priority 11 input

0x1C		PRIO_IN_SEL_12_13		
bits	**name**	**default**	**rw**	**description**
4:0	sel_12	12	rw	select an IRQ signal for priority 12 input
12:8	sel_13	13	rw	select an IRQ signal for priority 13 input

0x1E		PRIO_IN_SEL_14_15		
bits	**name**	**default**	**rw**	**description**
4:0	sel_14	14	rw	select an IRQ signal for priority 14 input
12:8	sel_15	15	rw	select an IRQ signal for priority 15 input

0x20		PRIO_IN_SEL_16		
bits	**name**	**default**	**rw**	**description**
4:0	sel_16	16	rw	select an IRQ signal for priority 16 input

0x30		IRQ_STATUS0			
bits	**name**	**default**	**rw**	**type**	**description**
0	prio_0	0	rw	status	interrupt priority 0 input
1	prio_1	0	rw	status	interrupt priority 1 input
2	prio_2	0	rw	status	interrupt priority 2 input
3	prio_3	0	rw	status	interrupt priority 3 input
4	prio_4	0	rw	status	interrupt priority 4 input
5	prio_5	0	rw	status	interrupt priority 5 input
6	prio_6	0	rw	status	interrupt priority 6 input
7	prio_7	0	rw	status	interrupt priority 7 input
8	prio_8	0	rw	status	interrupt priority 8 input
9	prio_9	0	rw	status	interrupt priority 9 input
10	prio_10	0	rw	status	interrupt priority 10 input
11	prio_11	0	rw	status	interrupt priority 11 input
12	prio_12	0	rw	status	interrupt priority 12 input
13	prio_13	0	rw	status	interrupt priority 13 input
14	prio_14	0	rw	status	interrupt priority 14 input
15	prio_15	0	rw	status	interrupt priority 15 input

0x32		IRQ_STATUS1			
bits	**name**	**default**	**rw**	**type**	**description**
0	prio_16	0	rw	status	interrupt priority 16 input

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag

0: do not change flag

0x34		IRQ_MASK0		
bits	**name**	**default**	**rw**	**description**
15:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x36		IRQ_MASK1		
bits	**name**	**default**	**rw**	**description**
0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x38		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
4:0	vno	0	w	vector number of interrupt to enable

0x3A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
4:0	vno	0	w	vector number of interrupt to disable

0x3C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
4:0	vmax	17	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x3E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
4:0	vno	17	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.8 Watchdog Module

7.8.1 Description

The Watchdog module wdog provides a mechanism to check if the software is still operating in the correct sequence. A*llok_out* stays asserted only if the software has written a correct sequence of magic cookies within a given time interval since the last correct sequence has been written. If the time interval expires (time-out) before the software has written the correct sequence or if a wrong value is written the *allok_out* signal will not be asserted until the correct sequence has been written again (recovery)

The start-up behavior can be configured at compile time: Either allok_out is asserted after nres and stays asserted until the wdog module is started and the first timeout occurs (watchdog scenario). Or it is not asserted after nres and will be asserted for the first time when the first correct sequence has been written (guardian scenario). In both cases the allok_out signal can be reasserted after a timeout by writing the correct sequence.

Operating Environment

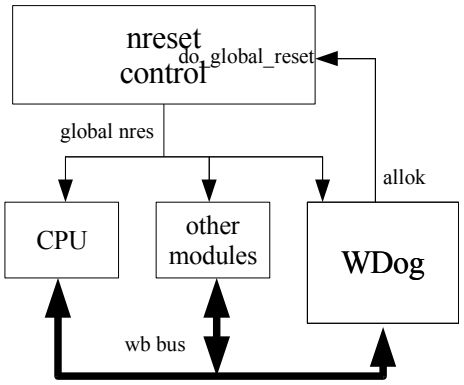


Figure 16: wdog module used as a reset generator for the entire system (watchdog)

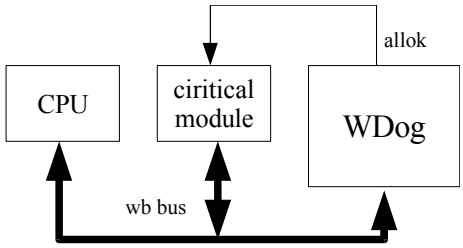
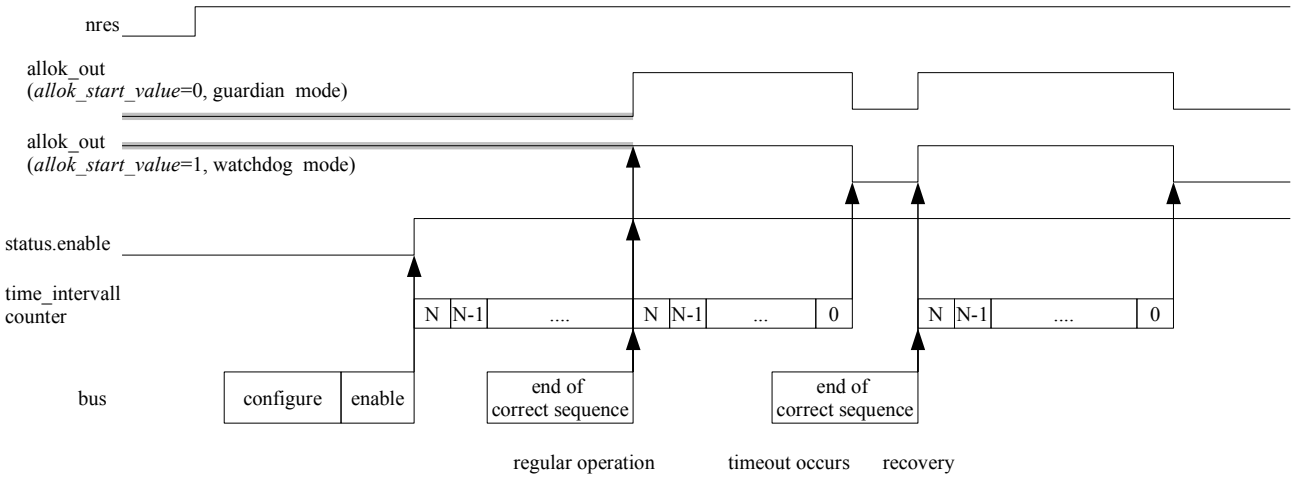
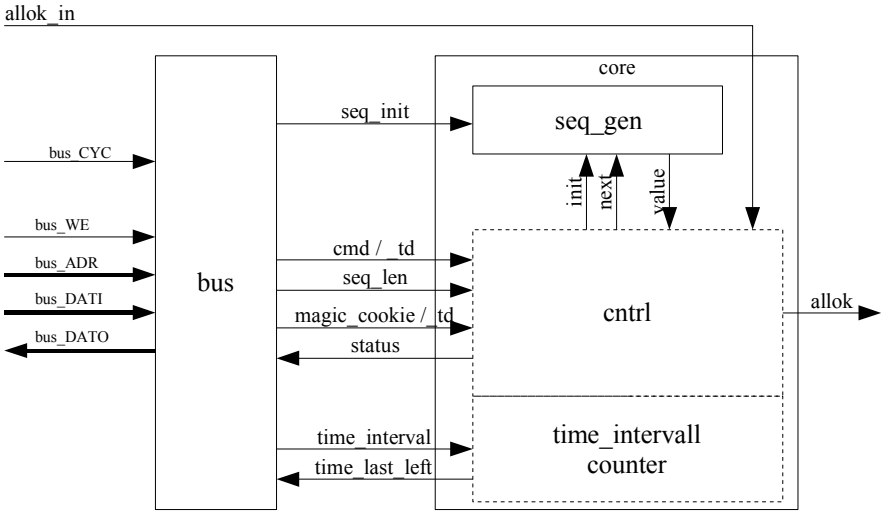


Figure 17: wdog module used to enable the execution of critical functions (guardian)

Timing Diagram



7.8.2 Structure



7.8.3 Physical Interface

Name	Dir	Description
bus_*	IO	Internal bus system
allok_out	O	"allocate" output
halt	I	stops the countdown timer (e. g. when CPU in standby)

7.8.4 Features

- run time configurable magic cookie sequence . The magic cookie sequence is generated from a start value with a simple linear feedback shift register. This allows to have multiple wdog modules (e.g. for multiple concurrent software processes) in the system with different magic cookie sequences.
- sequence length configurable
- timeout configurable
- once the watchdog is enabled the configuration is protected
- a reset is generated if watchdog is enabled and no correct byte is written within a certain time
- Sequences are calculated using the following formula:
  - ➔ temp = (prev >> 7) ^ (prev >> 5) ^ (prev >> 4) ^ (prev >> 3);
  - ➔ next = ((prev >> 1) & 0x7F) | (temp << 7);
  - ➔ This is equivalent to a linear feedback shift register which produces a maximum length sequence. Writing any other value will lead to the immediate reset.
  - ➔ Example: seq_init = 0x7A, seq_len = 4 -> magic_cookie[0..3] = 0x7A, 0xBD, 0x5E, 0x2F

7.8.5 Register Interface

7.8.5.1 Configuration Registers

The following configuration registers *must* all be written before the wdog is started for the first time. They can *only* be written when the wdog is disabled, all other writes will be ignored.

0x00		SEQ_INIT		
bits	**name**	**default**	**rw**	**description**
7:0	seq_init	0	rw	initial value for the magic cookie sequence generator. Value of 0xFF is not recommended.

0x02		SEQ_LEN		
bits	**name**	**default**	**rw**	**description**
7:0	seq_len	0	rw	number of magic cookies in the sequence seq_len=1..(2^seq_width)-1

0x04		TIME_INTERVAL_LW		
bits	**name**	**default**	**rw**	**description**
15:0	time_lw	0	rw	low word in system clock cycles

0x06		TIME_INTERVAL_HW		
bits	**name**	**default**	**rw**	**description**
15:0	time_hw	0	rw	high word

7.8.5.2 Status Registers

The following registers reflects the status of the Watchdog:

0x08		STATUS		
bits	**name**	**default**	**rw**	**description**
0	status	0	r	1: wdog is running, 0: wdog is disabled
1	timer_ready	0	r	1: timer is ready, 0: timer is not ready

0x0A		TIME_LAST_LEFT_LW		
bits	**name**	**default**	**rw**	**description**
15:0	left_lw	0	r	this register captures the value of the time_interval counter when a successful sequence of magic cookies is completed, i.e. it reflects how much time was left in the previous interval low word

0x0C		TIME_LAST_LEFT_HW		
bits	**name**	**default**	**rw**	**description**
15:0	left_hw	0	r	high word

0x0E		TIMER_VAL_LW		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	r	current timer value low word Note: TIMER_HW is sampled by reading TIMER_LW.

0x10		TIMER_VAL_HW		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	r	current timer value high word Note: TIMER_HW is sampled by reading TIMER_LW.

7.8.5.3 Command Registers

0x12		COMMAND		
bits	**name**	**default**	**rw**	**description**
7:0	cmd	0	w	writing 88h will start the wdog. This is reflected in the status register.

0x14		MAGIC_COOKIE		
bits	**name**	**default**	**rw**	**description**
7:0	mc	0	w	The software must write the next magic cookie here. The first magic cookie is seq_init.

7.9 Floating Point Unit

7.9.1 Description

This module provides IEEE 754 compliant 64 bit floating point arithmetic.

7.9.2 Features

- 64 bit floating point arithmetic operations
  - addition unit
  - subtraction unit
  - multiply unit
  - divide unit
  - type cast unit (integer to float, float to integer)
- subnormal number handling
- exception handling (NaN, overflow, underflow, ...)
- ACCU based data handling

7.9.3 Register Interface

0x00 0x02 0x04 0x06		**ADD_0 ADD_1 ADD_2 ADD_3**		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	w	first operand for addition (writing starts the operation) ACCU = ACCU + val

0x08 0x0A 0x0C 0x0E		**SUB_0 SUB_1 SUB_2 SUB_3**		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	w	first operand for subtraction (writing starts the operation) ACCU = ACCU - val

0x10 0x12 0x14 0x16		MUL_0 MUL_1 MUL_2 MUL_3		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	w	first operand for multiply (writing starts the operation) ACCU = ACCU * val

0x18 0x1A 0x1C 0x1E		DIV_0 DIV_1 DIV_2 DIV_3		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	w	first operand for divide (writing starts the operation) ACCU = ACCU / val

0x20 0x22 0x24 0x26		ACCU_0 ACCU_1 ACCU_2 ACCU_3		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	rw	accumulator value, calculation result

0x28 0x2A 0x2C 0x2E		I2F_0 I2F_1 I2F_2 I2F_3		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	rw	- integer to float cast input (write triggers cast, cast result will be written to accumulator) - float to integer cast result

0x30 0x32 0x34 0x36		F2I_0 F2I_1 F2I_2 F2I_3		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	w	- float to integer cast input (write triggers cast, cast result will be written to I2F)

0x38		CONTROL		
bits	**name**	**default**	**rw**	**description**
1:0	rmode	1	rw	rounding mode 0 : round toward zero 1 : round to nearest even 2 : round upward 3 : round downward

0x40		IRQ_STATUS			
bits	**name**	**default**	**rw**	**description**	**type**
0	evt_snan	0	rw	signaling NaN operand event	event
1	evt_qnan	0	rw	quiet NaN result event	event
2	evt_div0	0	rw	divide by 0 event - not set on NaN or infinite operands - set if result is infinite	event
3	evt_invalid	0	rw	invalid result event - not set on NaN or infinite operands - set if NaN calculation result value	event
4	evt_inexact	0	rw	inexact result event - not set on NaN or infinite operands - set if rounded result differs from exact calculated value	event
5	evt_underflow	0	rw	result underflow event - not set on NaN or infinite operands - set if result is sub-normal and inexact	event
6	evt_overflow	0	rw	result overflow event - not set on NaN or infinite operands - set if rounded value is to large to be represented	event
7	evt_infinity	0	rw	infinite result event	event
8	evt_zero	0	rw	zero result event	event
9	evt_negative	0	rw	negative result event	event

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag
0: do not change flag

0x44		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
9:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x48		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to enable

0x4A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to disable

0x4C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
3:0	vmax	10	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x4E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
3:0	vno	10	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.10 Timer Module

7.10.1 Description

The timer module contains hardware timers to generate periodic or interrupt events for a CPU. Typical used for software task scheduling purposes.

7.10.2 Structure

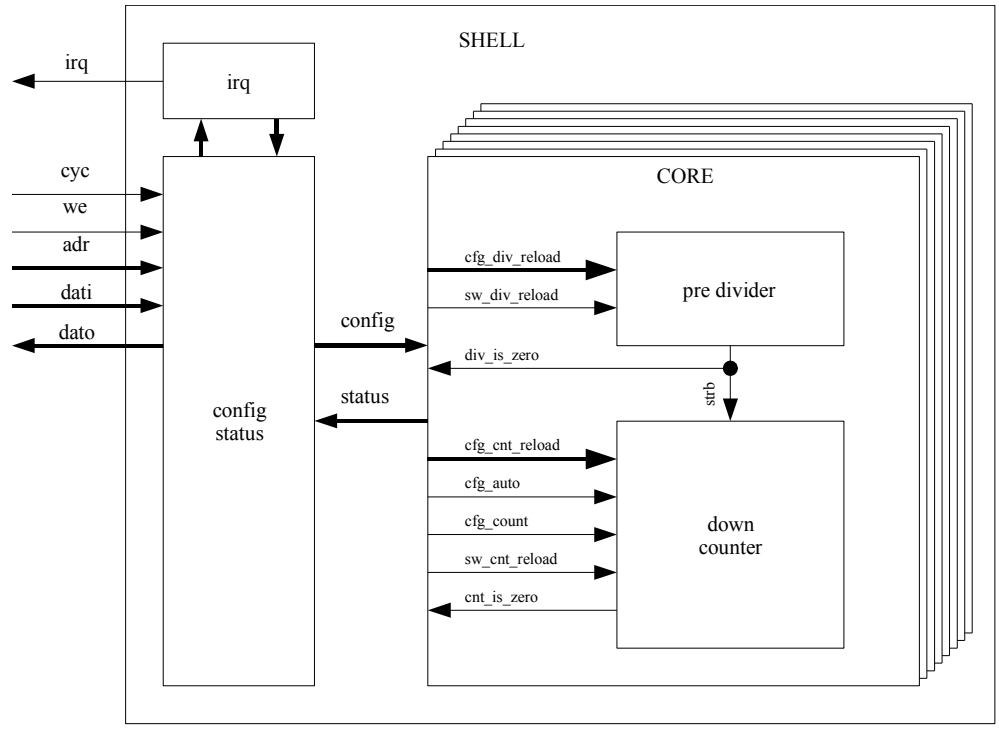


Figure 18: Timer Module

7.10.3 Physical Interface

Name	Dir	Description
bus_*	in	bus interface
irq	out	irq request to interrupt controller

7.10.4 Features

The following features are provided:

- **6 timers**

- predivider with:
  a) reload value
  b) trigger reload possible by software
  c) event when zero
- counter with:
  d) reload value
  a) trigger reload possible by software
  e) can be started / stopped
  f) automatic reload possible,
  otherwise counter stops when reaching zero (single shot mode)
  g) event when zero

7.10.5 Register Interface

Using the counter

1. set reload values for the counter and prescaler
2. enable the counter via the config register, choosing between free running and single shot mode
3. reload/restart the counter and or its prescaler by triggering a reload via the command register, e.g. to realign the phase of several counters

0x00		CONFIG		
bits	**name**	**default**	**rw**	**description**
5..0	enables	0	rw	1: counter n and prescaler n enabled, 0: disabled bit 0 = timer 0, bit 1 = timer 1, . Disabling a running counter will stop the counter and prescaler immediately.
11.6	cnt_auto_reloads	0	rw	1: auto reload for counter n enabled, 0: disabled bit 6 = timer 0, bit 7 = timer 1, .

0x02		COMMAND		
bits	**name**	**default**	**rw**	**description**
5..0	cnt_reload_cmd	0	w	1: trigger counter n reload / restart bit 0 = counter 0, bit 1 = counter 1, ...
11..6	div_reload_cmd	0	w	1: trigger predivider n reload / restart bit 6 = predivider 0, bit 7 = predivider 1, ...

Remark: The COMMAND register will be reset by hardware.

0x04 0x06 0x08 0x0A 0x0C 0x0E		CNT0_RELOAD CNT1_RELOAD CNT2_RELOAD CNT3_RELOAD CNT4_RELOAD CNT5_RELOAD		
bits	**name**	**default**	**rw**	**description**
15..0	cnt_reload	0	rw	counter reload value (cnt_reload_val = clock cycles -1)

0x10 0x12 0x14 0x16 0x18 0x1A		DIV0_RELOAD DIV1_RELOAD DIV2_RELOAD DIV3_RELOAD DIV4_RELOAD DIV5_RELOAD		
bits	**name**	**default**	**rw**	**description**
7..0	div_reload	0	rw	predivider reload value (div_reload_val = clock cycles -1)

0x1C 0x1E 0x20 0x22 0x24 0x26		CNT0_VALUE CNT1_VALUE CNT2_VALUE CNT3_VALUE CNT4_VALUE CNT5_VALUE		
bits	**name**	**default**	**rw**	**description**
15..0	cnt_val	0	r	current counter value (clock cycles = cnt_val +1)

0x30		IRQ_STATUS			
bits	**name**	**default**	**rw**	**type**	**description**
5..0	evt_cnt_zero	0	rw	event	counter n has been zero bit 0 = counter 0, bit 1 = counter 1, ...
11..6	evt_div_zero	0	rw	event	predivider n has been zero bit 6 = predivider 0, bit 7 = predivider 1, ...

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear “event” type flags

1: clear related flag
0: do not change flag

0x34		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
11:0	mask	0	rw	enable irq source: 1: enabled, 0: disabled

0x38		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to enable (bit set in IRQ_MASK)

0x3A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to disable (bit clear in IRQ_MASK)

0x3C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
3:0	vmax	12	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x3E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
3:0	vno	12	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.11 FLASH Control Module

7.11.1 Description

This module implements FLASH protection, erase and program functionality. Execution of program code located in FLASH memory strictly requires "read" mode. When switching to other FLASH modes (program or erase) the user has to ensure that during this time code is executed in RAM. Before returning to code in FLASH, mode has to be switched back to "read".

7.11.2 Features

- area protection (4 Kbyte areas)
- supported modes:
  - mass erase
  - page erase
  - program
  - read
- down to double-word (32 bit) programming
- system frequency adaptive

7.11.3 Register Interface

0x02		**AREA_8_15**		
bits	**name**	**default**	**rw**	**description**
7:0	area	0	rw	writable 0 : area protected 1 : area writable areas 8 .. 15: area 8 : 0x8000 - 0x8FFF area 9 : 0x9000 - 0x9FFF ... area 15 : 0xF000 - 0xFFFF
15:8	pass	0x96	rw	password must be written as 0xA5 will always be read as 0x96

0x04		MODE		
bits	**name**	**default**	**rw**	**description**
7:0	mode	0x01	rw	0x01 : read 0x04 : program 0x10 : page erase 0x40 : mass erase any other written mode value results in FLASH read mode, program/erase modes: write access to appropriate flash address starts program/erase cycle (see busy flag of status register, consider word config and row programming incomplete flag in program mode)
15:8	pass	0x96	rw	password must be written as 0xA5 will always be read as 0x96

0x06		STATUS		
bits	**name**	**default**	**rw**	**description**
0	busy	0	r	FLASH program/erase state 0 : ready 1 : busy
1	incomplete	0	r	row programming incomplete current number of programmed row words != word config
2	write_error	0	r	unexpected write to area protected memory occurred, will be cleared when STATUS is read

0x0E		WORD_CONFIG		
bits	**name**	**default**	**rw**	**description**
5:0	config	63	rw	number of words to program within a row 1 : 2 words … 31 : 32 words ... 63 : 64 words (complete row) Note: number of words has to be even (double-word programming only, low address has to be written first)
15:8	pass	0x96	rw	password must be written as 0xA5 will always be read as 0x96

0x10		FREQ_CONFIG		
bits	**name**	**default**	**rw**	**description**
1:0	config	0x1	rw	system frequency config to get a correct erase and program timing 0 : system frequency is 8 MHz 1 : system frequency is 16 MHz 2 : system frequency is 24 MHz 3 : system frequency is 32 MHz
15:8	pass	0x96	rw	password must be written as 0xA5 will always be read as 0x96

7.12 DMA Sub-Module

7.12.1 Description

This module implements a one channel direct memory access unit which assists the CPU in data handling. Two of these modules can be used to set up a full duplex Host Interface in combination with an interface module like a SPI slave or an I²C Slave.

7.12.2 Features

- configurable DMA channel
- host interface full duplex mode support (together with a 2nd corresponding DMA channel)

7.12.3 Register Interface

0x00		CONTROL		
bits	**name**	**default**	**rw**	**description**
0	enable	0	rw	channel enable
1	size	0	rw	data size 0 : byte handling 1 : word handling
2	src_inc	0	rw	source address auto increment 0 : no increment 1 : data size dependent increment (byte : 1, word : 2)
3	dst_inc	0	rw	destination address auto increment 0 : no increment 1 : data size dependent increment (byte : 1, word : 2)
4	use_len	0	rw	use LENGTH register value to stop transfer 0 : for host interfaces 1 : for generic data streams
5	trf_sel	0	rw	selection of TRF registers information 0 : select source side information to save 1 : select destination side information to save
6	duplex	0	rw	allow a corresponding DMA channel to connect to work in full duplex mode as a host interface handler. The corresponding DMA channel will then be able to set the source address. 0 : for generic data stream mode 1 : for duplex host interface mode
8	area_0_sel	0	rw	selection of AREA_0 register usage 0 : use AREA_0 regs to allow source side mem access 1 : use AREA_0 regs to allow destination side mem access
9	area_1_sel	0	rw	selection of AREA_1 register usage 0 : use AREA_1 regs to allow source side mem access 1 : use AREA_1 regs to allow destination side mem access

Note: If both area_0_sel and area_1_sel are configured for source side then destination side access will be allowed without restrictions and vice verse.

0x02		SRC_ADDR		
bits	**name**	**default**	**rw**	**description**
15:0	addr	0	rw	source address pointer

0x04		DST_ADDR		
bits	**name**	**default**	**rw**	**description**
15:0	addr	0	rw	destination address pointer

0x06		LENGTH		
bits	**name**	**default**	**rw**	**description**
15:0	len	0	rw	transfer data length in units of "data size"

0x08		TRF_START_ADDR		
bits	**name**	**default**	**rw**	**description**
15:0	addr	0	r	current / last DMA data transfer block start address

0x0A		TRF_LENGTH		
bits	**name**	**default**	**rw**	**description**
15:0	len	0	r	current / last DMA data transfer block length in units of "data size"

0x0C		REQ_SELECT		
bits	**name**	**default**	**rw**	**description**
2:0	src	0	rw	selection of a DMA master (source) request source 0 : permanent request (memory to memory generic data stream) 1 : SPI slave (host interface) 2 : SPI (generic data stream) 3 : I²C slave (host interface) 4 : I²C slave (generic data stream) 5 : I²C master (generic data stream) 6 : UART0 (generic data stream) 7 : UART1 (generic data stream)
6:4	dst	0	rw	selection of a DMA slave (destination) request source 0 : permanent request (memory to memory generic data stream) 1 : SPI slave (host interface) 2 : SPI (generic data stream) 3 : I²C slave (host interface) 4 : I²C slave (generic data stream) 5 : I²C master (generic data stream) 6 : UART0 (generic data stream) 7 : UART1 (generic data stream)

0x10 0x12		AREA_0_START_ADDR AREA_1_START_ADDR		
bits	**name**	**default**	**rw**	**description**
15:0	addr	0	rw	allowed DMA access area start address

0x14 0x16		AREA_0_LENGTH AREA_1_LENGTH		
bits	**name**	**default**	**rw**	**description**
15:0	len	0x8000	rw	allowed DMA access area length in words

0x20		IRQ_STATUS			
bits	**name**	**default**	**rw**	**description**	**type**
0	evt_finished	0	rw	data block transfer has finished	event
1	evt_src_area	0	rw	src area access denied event	event
2	evt_dst_area	0	rw	dst area access denied event	event

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags

1: clear related flag
0: do not change flag

0x24		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
2:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x28		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
1:0	vno	0	w	vector number of interrupt to enable

0x2A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
1:0	vno	0	w	vector number of interrupt to disable

0x2C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
1:0	vmax	3	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x2E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
1:0	vno	3	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.13 SPI Master / Slave Interface

7.13.1 Description

The Serial Peripheral Interface (SPI) is a full-duplex serial communication interface used between microprocessors and peripheral chips. It is a synchronous data link between masters and slaves, whereby the master generates the clock signal. Synchronous protocols are more tolerant to clock mismatch, since the slave has a time reference for each bit.

The SPI uses a four wire interface which contains the already mentioned clock signal (sck), an inverted slave select (nss) and a data signal for each direction (sdo and sdi).

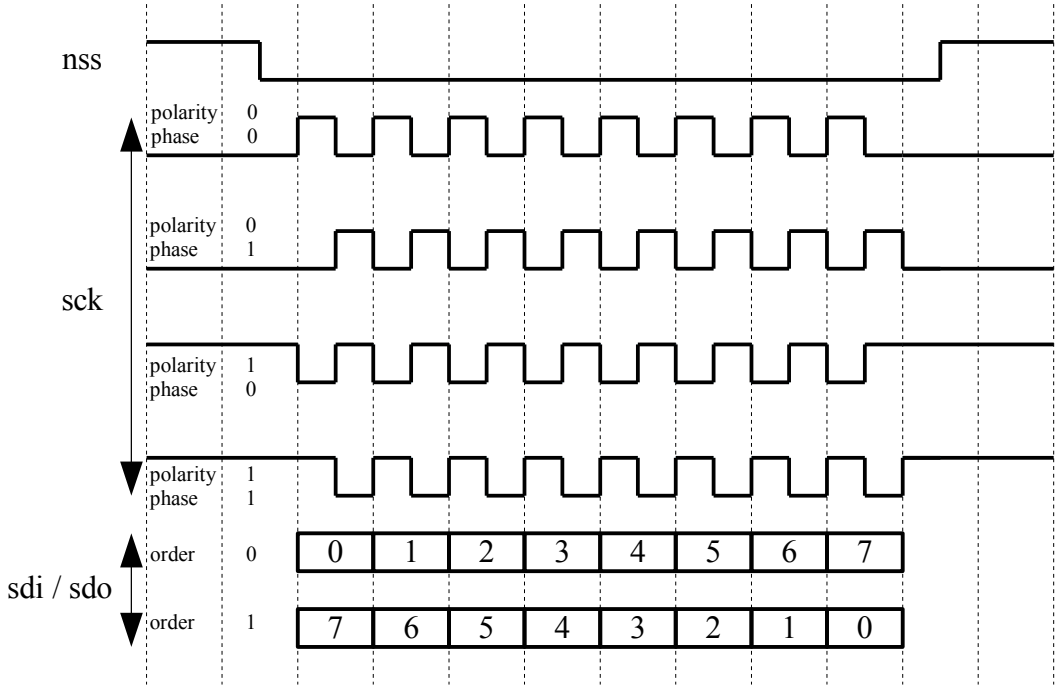


Figure 19: SPI Signal Timing

The figure above shows a typical timing for a SPI transfer. The master selects the device by lowering the nss signal and then generates the sck clock signal. The sdo and sdi contain the data bits. Another data word may be transferred by keeping the nss signal low. The master can address multiple slaves but for every slave a separate nss signal is needed.

Different clock signals can be generated. The desired shape is selected with the clock polarity and clock phase. This makes the master compatible with other existing interfaces.

7.13.2 Structure

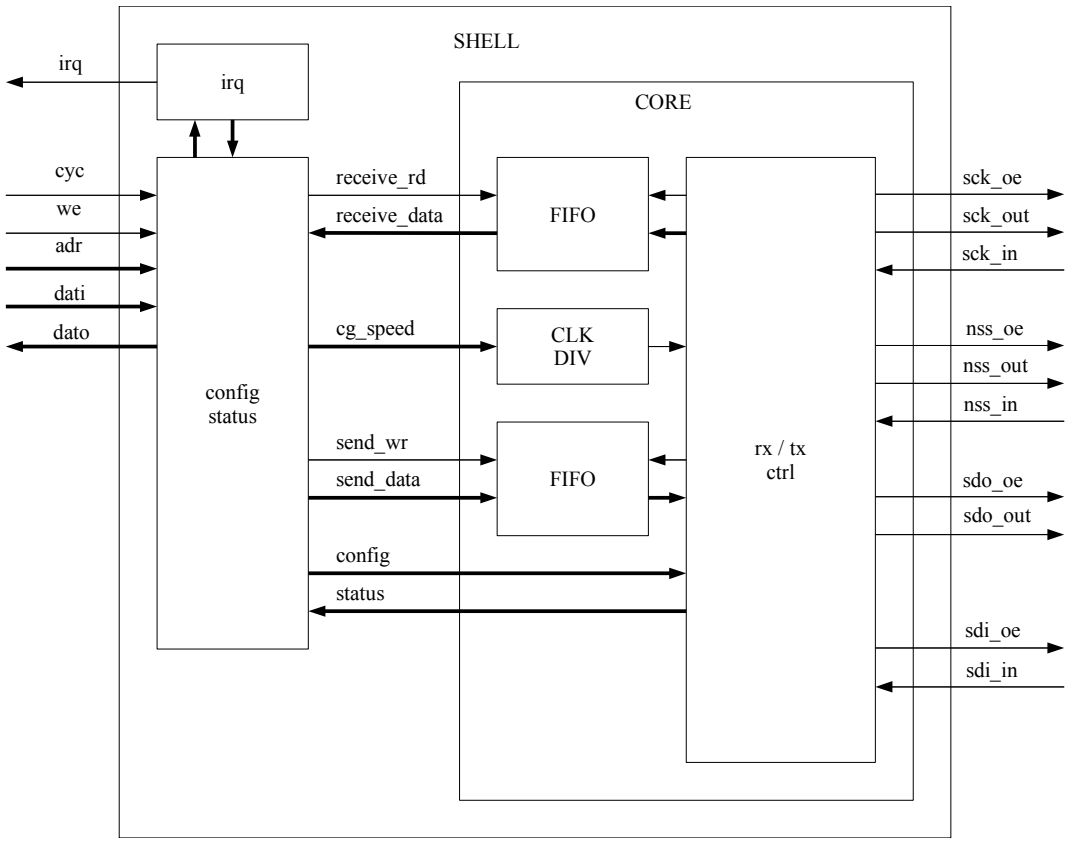


Figure 20: SPI Module Structure

7.13.3 Physical Interface

Signal	Dir	Description
bus_*	IO	internal bus system
irq	O	level interrupt output
nss_*	IO	slave select (direction depending on master/slave function)
sck_*	IO	clock (direction depending on master/slave function)
sdo_*	O	output data
sdi_*	I	input data

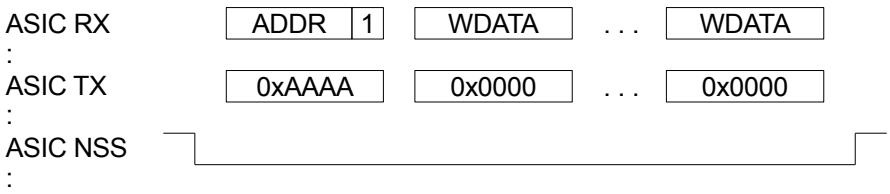
7.13.4 Features

- can be used as master or slave
  - master speed up to f_sys / 4
  - slave speed upto f_sys / 8
- the SPI Interface consists of the following 4 signals:
  - SCK SPI clock (driven by master)
  - NSS low active slave select (driven by master)
  - SDO data out
  - SDI data in
- configurable phase, polarity and bit order
- configurable transfer word bit length
- slave mode SPI clock monitoring (timeout)
- 4 byte transmit and receive FIFOs

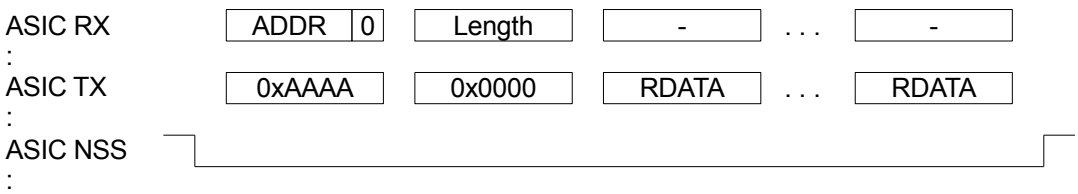
7.13.5 SPI host interface protocol

ASIC NSS has to be inactive (high level) between two host interface accesses to signal a new address value. Bit 0 of address word is used to signal the access type (1 = write, 0 = read). Address, length and data values are 16 bit words. The ASIC answers the address value with a 0xAAAA value which shows that the ASIC is ready for a transmission. When the ASIC is waking up this value can be polled to check the ASIC state.

SPI Hostinterface Write



SPI Hostinterface Read



7.13.6 Register Interface

0x00		DATA		
bits	**name**	**default**	**rw**	**description**
15:0	data	0	rw	write: transmit data read: receiver data does not keep NSS active after data transmit

0x02		DATA_KEEP_NSS		
bits	**name**	**default**	**rw**	**description**
15:0	data	0	w	write: transmit data keep NSS active after data transmit

0x04		CONFIG		
bits	**name**	**default**	**rw**	**description**
0	order	1	rw	order 0: LSB first 1: MSB first
1	phase	0	rw	phase 0: 1st edge shift, 2nd edge sample 1: 1st edge sample, 2nd edge shift
2	polarity	0	rw	polarity 0: clock off level 0 1: clock off level 1
3	slave	1	rw	slave 0: master 1: slave
7:4	length	7	rw	data bit count to transfer data word length = length + 1 example: length = 7 => 8 bit transfer example: length = 15 => 16 bit transfer
8	slave_high_z	0	rw	when SPI interface is configured as slave: 0: drive miso / sdo 1: miso / sdo switched to high Z when SPI interface is configured as master: this config bit is ignored
9	invert_nss	0	rw	invert SPI select signal 0 : active low 1 : active high
10	invert_data	0	rw	invert SPI input and output data
11	pause_nss	0	rw	minimum inter shift NSS inactive time 0: 1 bit length 1: 2 bit lengths when SPI interface is configured as slave: this config bit is ignored
12	sdi_irq_pol	0	rw	SDI interrupt active level, in case of inactive NSS
13	swap_sdi_sdo	0	rw	0: SDI = data input, SDO = data output 1: SDI = data output, SDO = data input
15	enable	0	rw	0 : SPI interface is disabled 1 : SPI interface is enabled Note: enable = 0 clears interface FIFO's (as long as the interface is disabled, the FIFO's are kept clean)

0x06		BAUD_CONFIG		
bits	**name**	**default**	**rw**	**description**
15:0	divider	0	rw	divider = (system clock frequency) / (2 * baudrate) min divider value = 2 -> max baudrate = Fsys / 4 max divider value = 65535 -> min baudrate = Fsys / 131070 example: Fsys = 16MHz ● max baudrate = 4 MBaud ● min baudrate = 122 Baud

0x08		TIMEOUT_CONFIG		
bits	**name**	**default**	**rw**	**description**
15:0	sclk_timeout	0xFFFF	rw	sclk timeout value maximum allowed count of system clock cycles between 2 SPI clock edges

0x0A		RX_FIFO_TIMEOUT		
bits	**name**	**default**	**rw**	**description**
15:0	rx_fifo_timeout	0	rw	time in bits until receive timeout event timeout will be restarted when new incoming data byte timeout will be set when counted to zero and RX_FIFO.state = non empty

0x0C		FIFO_CLEAR		
bits	**name**	**default**	**rw**	**description**
0	rx_fifo_clear	0	w	1: receiver FIFO clear
1	tx_fifo_clear	0	w	1: transmit FIFO clear
2	reset	0	w	1: module reset

0x0E		FIFO_LEVELS		
bits	**name**	**default**	**rw**	**description**
2:0	rx_fifo_level	0	rw	receive FIFO fill level (read only fifo status)
6:4	tx_fifo_level	0	rw	transmit FIFO fill level (read only fifo status)
10:8	rx_fifo_high_water	0x2	rw	high water receive FIFO level interrupt will be asserted when receive FIFO fill level increases to this value
14:12	tx_fifo_low_water	0	rw	low water transmit FIFO level interrupt will be asserted when transmit FIFO fill level decreases to this value

0x20		IRQ_STATUS			
bits	**name**	**default**	**rw**	**type**	**description**
0	evt_rx_fifo_ov_err	0	rw	event	software did not read incoming data fast enough
1	evt_rx_fifo_ur_err	0	rw	event	software has read from empty receive fifo
2	evt_tx_fifo_ov_err	0	rw	event	software wrote data to full transmit fifo
3	evt_tx_fifo_ur_err	0	rw	event	software has not written outgoing data fast enough
4	evt_sot	0	rw	event	start of transfer (nss has changed to active level)
5	evt_eot	0	rw	event	end of transfer (nss has changed to inactive level)
6	evt_sclk_timeout	0	rw	event	sclk timeout interrupt
7	evt_shift_done	0	rw	event	per word shift interrupt when a word shift completed
8	rx_fifo_nempty	0	rw	status	receive fifo is not empty
9	rx_fifo_timeout	0	rw	status	receive fifo timeout
10	rx_fifo_high_water	0	rw	status	receive fifo elements > high water level
11	rx_fifo_full	0	rw	status	receive fifo is full
12	tx_fifo_empty	0	rw	status	transmit fifo is empty
13	tx_fifo_low_water	0	rw	status	transmit fifo elements <= low water level
14	tx_fifo_nfull	0	rw	status	transmit fifo is not full
15	evt_sdi	0	rw	event	SDI had config.sdi_irq_pol polarity while NSS was active

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag
0: do not change flag

0x24		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
15:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x28		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to enable

0x2A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to disable

0x2C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
4:0	vmax	16	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x2E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
4:0	vno	16	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.14 UART Interface

7.14.1 Description

The UART protocol is commonly used with personal computers as a debug interface. It may also be used as an external boot source in the software flow.

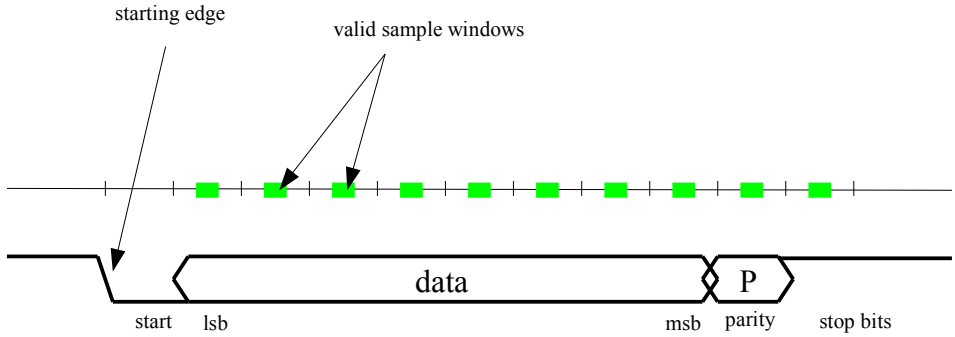


Figure 21: UART transmission

It is an asynchronous protocol with start and stop bits. The number of data bits, the use of parity, the number of stop bits an

The Figure 21 shows a sample transmission with the UART protocol. The edge of the start bit indicates the beginning of a transfer. Eight data bits, beginning with the least significant bit, and one parity bit with (even or odd) parity are used in this example.

Even or (odd parity) means that the parity bit completes the number of '1' bits to an even (or odd) number. A parity of none means that no parity bit is sent at all.

From the first edge, the rest of the packet is subdivided into bits of the same length given by the transfer speed. For optimal results, the incoming data is sampled between two edges.

The receiver must be able to sample the data at least in a time window around the middle of the last bit (stop bit). The possible error is determined by the quantization error of the start edge and the accumulated quantization error for all bits until the stop bit.

Assuming 1 start bit, 8 data bits, 1 parity bit and a 50% window around the middle of the stop bit the maximum clock tolerance can be calculated with:

$$clock\ tolerance < \pm\ 25\ \% / (1 + 8 + 1 + 0.5)$$

This results in a needed clock tolerance smaller than $\pm\ 2.38\ \%$.

A phase jitter can be ignored, because its possible value is very small compared to the total bit length.

For using the UART core a physical transmitter is required which implements physical layer standards like RS232 or RS485.

7.14.2 Structure

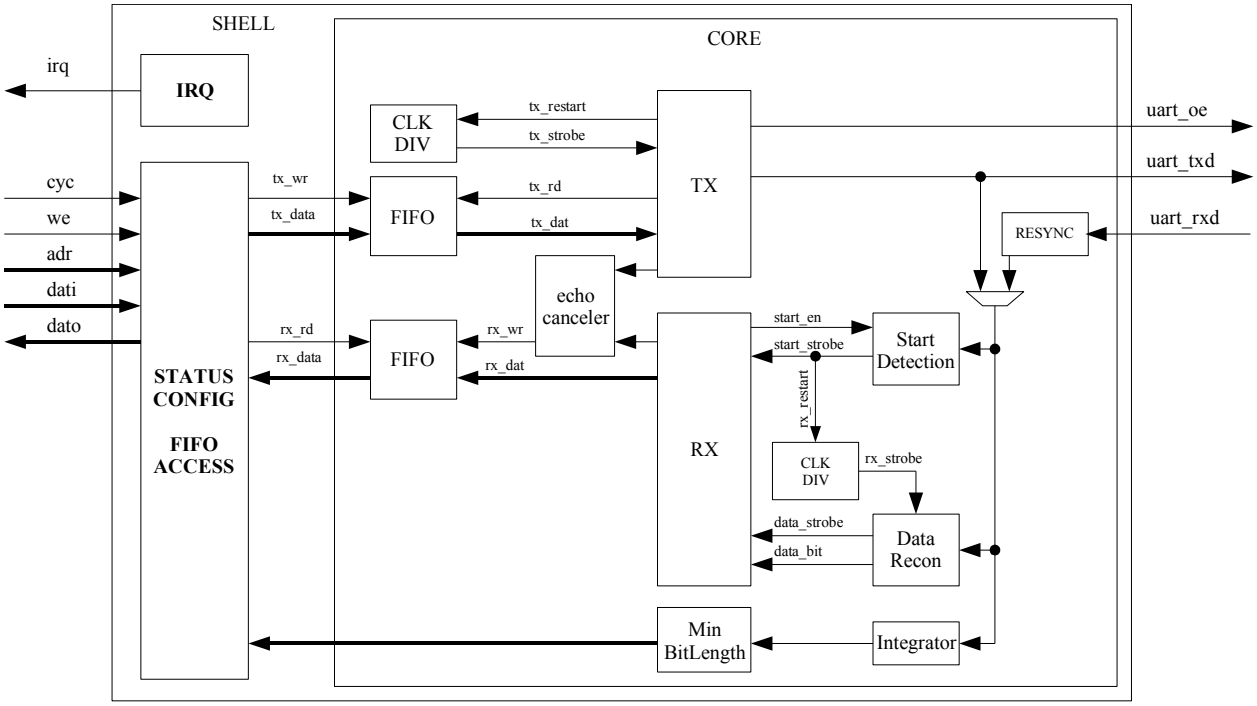


Figure 22: UART Module

7.14.3 Physical Interface

Signal	Dir	Description
bus_*	IO	internal bus system
irq	O	level interrupt output
uart_oe	O	high active output enable
uart_txd	O	UART transmit data
uart_rxd	I	UART receive data

7.14.4 Features

- variable data bits (7 or 8)
- variable stop bits (1 or 2)
- variable parity generation (none, odd, even)
- variable fractional baud rate (bit length)
- variable irq generation
- receive can be disabled
- full-duplex mode
- half-duplex mode
  - hardware controlled output enable with lead-in and lead-out
  - optional hardware echo canceling
  - also possible with software
- input filters
  - start bit filter with threshold at 50% of bit length
  - data bit filter with offset at 25% of bit length and length of 50% of bit length
  - data bit filter also used for parity and stop bit
- error detection
  - start bit error
  - parity error
  - stop bit error
- minimum bit length (time between two edges) readable by software
  - value is reset when read
  - equipped with integrator with threshold at 12.5% of bit length
- FIFOs
  - RX FIFO with 4 elements
  - TX FIFO with 4 elements
  - separate clear function for receive and send
  - send fifo low water
  - receive fifo high water and timeout

7.14.5 Register Interface

0x00		CONFIG		
bits	**name**	**default**	**rw**	**description**
0	rx_enable	0	rw	enable receive function
2..1	par_type	0	rw	type of parity bit 0: no parity bit 1: odd parity 2: even parity
3	stop_bits	0	rw	number of stop bits for transmit 0: one stop bit 1: two stop bits
4	data_bits	0	rw	number of data bits 0: 8 data bits 1: 7 data bits
5	loopback_en	0	rw	enable internal loop-back
9..6	rx_fifo_high_water	0	rw	receive FIFO high water level from 0...FIFOS_SIZE-1
13..10	tx_fifo_low_water	0	rw	send FIFO low water level from 0...FIFOS_SIZE-1
14	echo_cancel	0	rw	remove all receive echo until 8 bit times after last transmitted stop bit
15	invert_data	0	rw	invert UART input and output data signals

0x02		BAUD_CONFIG		
bits	**name**	**default**	**rw**	**description**
1..0	frac	0	rw	fractional value
15..2	divider	0	rw	divider value

baudrate = system_frequency / (1 + divider + fraq/4)

min divider value = 15

0x04		RX_FIFO_TIMEOUT		
bits	**name**	**default**	**rw**	**description**
15..0	rx_fifo_timeout	0	rw	time in bits until receive timeout event timeout will be restarted when new incoming data byte timeout will be set when counted to zero and RX_FIFO.state = non empty

0x06		FIFO_CLEAR		
bits	**name**	**default**	**rw**	**description**
0	rx_fifo_clear	0	w	when 1, clear receive fifo
1	tx_fifo_clear	0	w	when 1, clear transmit fifo

0x08		RX_DATA		
bits	**name**	**default**	**rw**	**description**
7..0	rx_data	0	r	receive data from fifo (element will be removed from fifo)
8	valid	0	r	1 = at least one element in receive fifo

0x0A		TX_DATA		
bits	**name**	**default**	**rw**	**description**
7..0	tx_data	0	w	transmit data to fifo (element will be inserted into fifo)

0x0C		FIFO_LEVELS		
bits	**name**	**default**	**rw**	**description**
3..0	rx_fifo_level	0	r	number of elements in rx fifo (from 0 to FIFO_SIZE)
7..4	tx_fifo_level	0	r	number of elements in tx fifo (from 0 to FIFO_SIZE)

0x0E		MIN_BIT_LENGTH		
bits	**name**	**default**	**rw**	**description**
13..0	min_bit_length	0x3FFF	r	minimum bit length (number of clock cycles between two edges)

Remark: The min_bit_length is reset to maximum value when reading the register. The first value after reset is invalid.

0x20		IRQ_STATUS			
bits	**name**	**default**	**rw**	**type**	**description**
0	evt_rx_fifo_ov_err	0	rw	event	software did not read incoming data fast enough
1	evt_rx_fifo_ur_err	0	rw	event	software has read from empty receive fifo
2	evt_tx_fifo_ov_err	0	rw	event	software wrote data to full transmit fifo
3	evt_rx_start_err	0	rw	event	start bit error
4	evt_rx_par_err	0	rw	event	parity error
5	evt_rx_stop_err	0	rw	event	stop bit error
6	evt_rx_fifo_ne_timeout	0	rw	event	receive fifo timeout in case of an non empty RX fifo
7	evt_rx_fifo_timeout	0	rw	event	receive fifo timeout - independent of RX FIFO state
8	rx_fifo_nempty	0	rw	status	receive fifo is not empty
9	rx_fifo_high_water	0	rw	status	receive fifo elements > high water level
10	rx_fifo_full	0	rw	status	receive fifo is full
11	tx_fifo_empty	0	rw	status	transmit fifo is empty
12	tx_fifo_low_water	0	rw	status	transmit fifo elements <= low water level
13	tx_fifo_nfull	0	rw	status	transmit fifo is not full
14	tx_not_transmitting	0	rw	status	the UART is not transmitting (TX FIFO empty and not currently sending a byte)

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag
0: do not change flag

0x24		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
14:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x28		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to enable

0x2A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to disable

0x2C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
3:0	vmax	15	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x2E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
3:0	vno	15	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.14.6 Finding the Baud Rate

The Min Bit Length module counts the number of clock cycles between two edges. To reduce the effects of disturbances on the UART line, resulting in a to small value, an integrator was inserted between resynchronized data and the Min Bit Length detection. This Integrator uses a threshold of $\frac{Speed}{8}$ in clock cycles.

Therefore setting the expected speed value too high, will not allow the Min Bit Length detection to find a value. So the expected Speed Value has to be set to the highest value (e.g. 115200 kBaud + 20%) and start to listen for an applicable UART Packet (e.g. 0x55). The first value after reset has to be dropped, since it contains the number of clock cycles from reset to first start edge. The optimal result can be achieved by sending multiple 0x55 UART packets.

After dropping the first value the next following value can directly used as speed value. After that the correct UART data bytes should be received.

A higher level software protocol has to be implemented, to inform the sender that the Baud Rate finding phase can be finished and the desired data transfer can be started.

A possible algorithm may work as following:

1. set Speed Value to maximum Baud Rate + 20% but **do not enable** Receiver
2. receive 0x55 packet and drop Min Bit Length Value
3. receive 0x55 packet and set Min Bit Length Value as Speed Value and enable Receiver
4. receive 0x55 packet and check RX Data
   if data matches inform sender to send desired data, else a timeout should trigger Sender to send new 0x55 packet

7.14.7 Baud Rate Errors

As explained in the introductory chapter the Absolute Baud Rate Error has to be smaller than 2.38 % and the Accumulated Bit Error has to be smaller than 25% for correct operation of the UART module. The Error: Reference source not found shows the resulting Accumulated Bit Error for different Baud Rates at variable system frequencies. The diagram shows an absolute theoretical limit. For robust operation the system frequency should be higher.

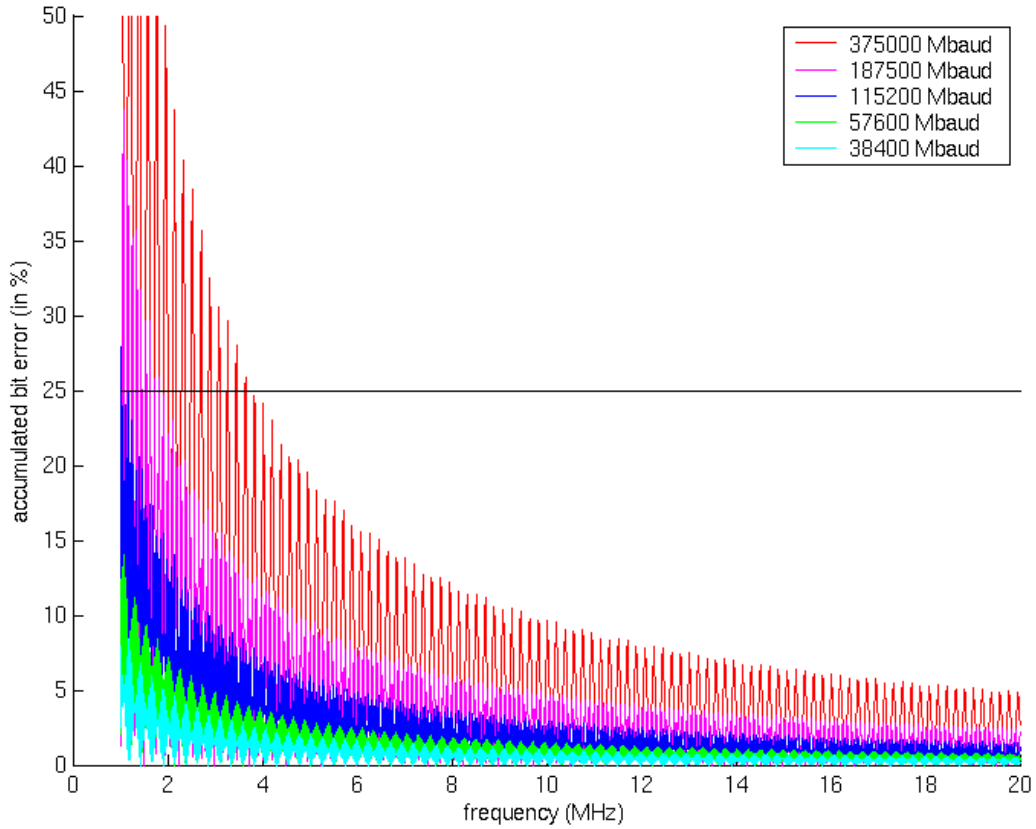


Figure 23: Accumulated bit error

The next table shows exemplary Speed Values to be set in the register interface for a wanted baud rate at a given core frequency.

System Frequency (MHz)	Wanted Baud Rate (bit/s)	Speed Value (Registerinterface)	Possible Baud Rate (bit/s)	Absolute Baud Rate Error (%)	Accumulated Bit Error (%)
0.5	9600	52	9434	1.73	19.2
1	19200	52	18868	1.73	19.2
2	38400	52	37736	1.73	19.2
3	57600	52	56604	1.73	19.2
6	115200	52	113208	1.73	19.2
10	187500	53	185185	1.23	15
20	375000	53	370370	1.23	15

Table 1: baud rate errors

To improve the performance at a lower system frequency, a fractional value can be used. The Error: Reference source not found shows the theoretical limit when using fractional values. It can be seen that the resulting errors are much smaller.

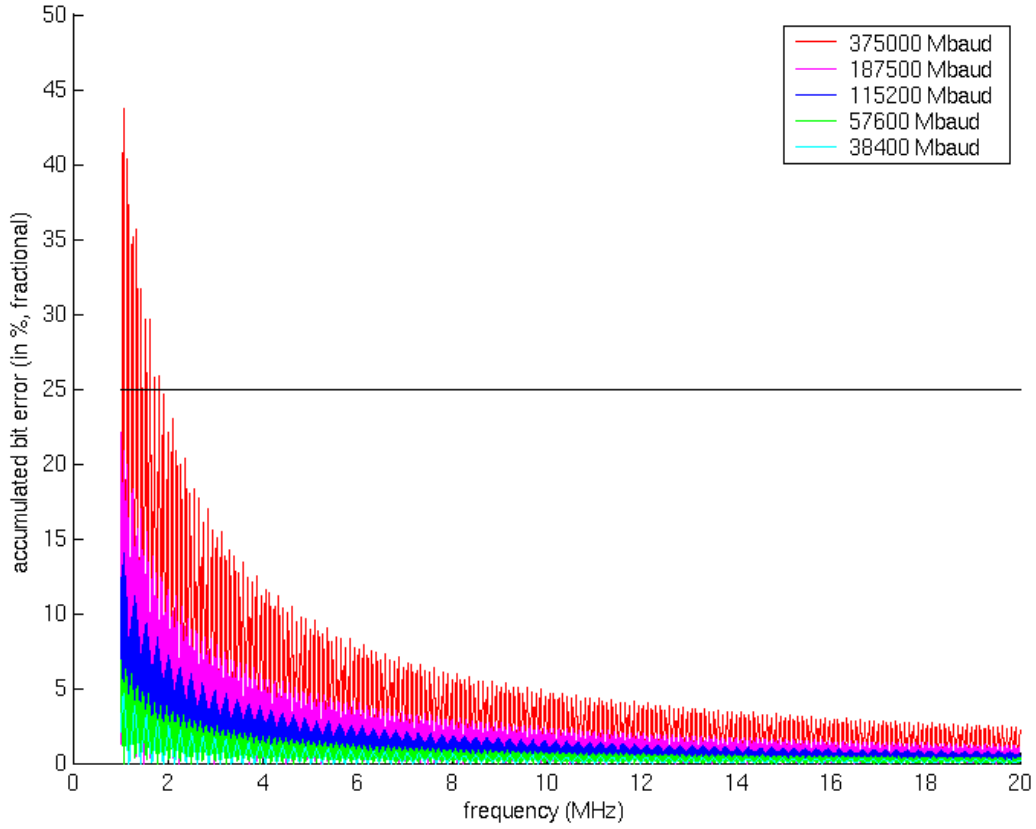


Figure 24: Accumulated bit error (fractional)

7.14.8 Half-Duplex Operation

The UART can be used with tri-state drivers where tx data is connected to rx data. To support these configurations the UART has a hardware controlled output enable and an hardware echo cancellers.

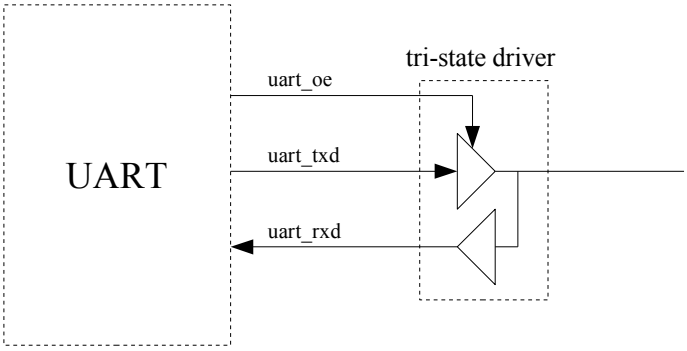


Figure 25: half-duplex mode

The following figure shows how the optional hardware echo cancellers works if enabled. Any data received until 8 bit times after the transmitted stop bit will be suppressed. Any other data will not be affected.

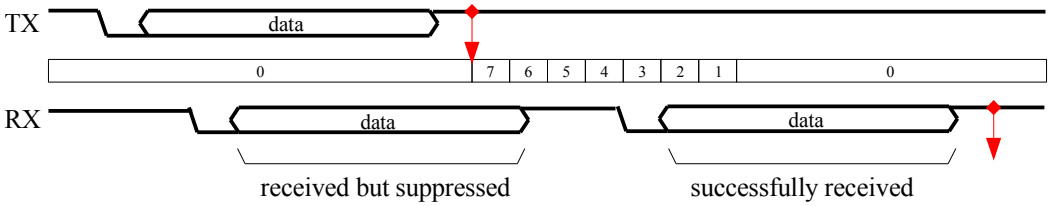


Figure 26: echo canceling

If not enabled all transmitted data will be echoed back to the software.

The “TX transmitting” status signal can be used to implement half-duplex mode in software, which means controlling the output enable and dis-/enabling the receiver to prevent echos.

The UART output enable signal is activated 1 bit length before start bit and disabled directly after stop bit.

There is no additional delay inserted between two successive data transmits.

7.15 I²C Master Interface

7.15.1 Description

The I²C master interface is a two wire interface (SDA, SCL) compatible to the well known standard. It comprises a transmit and receive FIFO. For maximum flexibility and compatibility, only the lowest levels of communication are handled in hardware.

7.15.2 Structure

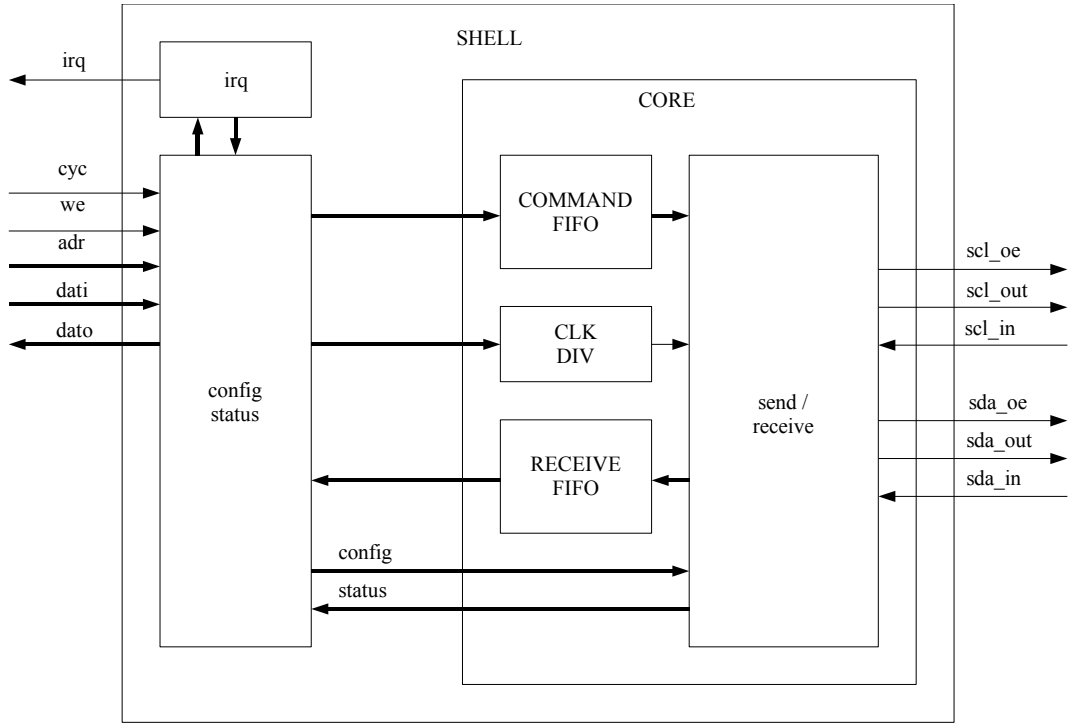


Figure 27: I²C Module

7.15.3 Physical Interface

Signal	Dir	Description
bus_*	IO	internal bus system
irq	O	level interrupt output
i2c_scl_oe	O	high active I²C clock output enable
i2c_scl_out	O	I²C clock out
i2c_scl_in	I	I²C clock in
i2c_sda_oe	O	high active I²C data output enable
i2c_sda_out	O	I²C data out
i2c_sda_in	I	I²C data in

7.15.4 Features

- 8 word command (transmit) FIFO
- 4 byte receive FIFO
- programmable baud rate
- supports multi byte transfers

7.15.5 Register Interface

0x00		CMD_DATA		
bits	**name**	**default**	**rw**	**description**
7..0	data	0	w	data word to send
8	send	0	w	1: data is sent, 0: data is received
9	start	0	w	1: start bit is sent
10	ack	0	w	acknowledge which is sent/expected
11	stop	0	w	1: stop bit is sent

The command FIFO has a depth of 8 words

0x02		REC_DATA		
bits	**name**	**default**	**rw**	**description**
7..0	data	0	r	data word which was received

The receive FIFO has a depth of 4 bytes

0x04		CONFIG		
bits	**name**	**default**	**rw**	**description**
2..0	rec_fifo_level	0x2	rw	high water receive FIFO level interrupt will be asserted when receive FIFO fill level increases to this value
7..4	cmd_fifo_level	0	rw	low water command FIFO level interrupt will be asserted when command FIFO fill level decreases to this value

0x06		BAUD_CONFIG		
bits	**name**	**default**	**rw**	**description**
15..0	divider	0	rw	baud divider

The baud divider is determined through the following equation:

$$baud\ divider = \frac{system\ clock}{8 \cdot baud\ rate}$$

0x08		FIFO_CLEAR		
bits	**name**	**default**	**rw**	**description**
0	rec_fifo_clear	0	w	1: receiver FIFO clear
1	cmd_fifo_clear	0	w	1: transmit FIFO clear

0x0A		FIFO_LEVELS		
bits	**name**	**default**	**rw**	**description**
2..0	rec_fifo_level	0	r	receive FIFO fill level
7..4	cmd_fifo_level	0	r	transmit FIFO fill level

0x20		IRQ_STATUS			
bits	**name**	**default**	**rw**	**type**	**description**
0	evt_rec_fifo_ov_err	0	rw	event	software did not read incoming data fast enough
1	evt_rec_fifo_ur_err	0	rw	event	software has read from empty receive fifo
2	evt_cmd_fifo_ov_err	0	rw	event	software wrote data to full transmit fifo
3	evt_cmd_fifo_ur_err	0	rw	event	software has not written outgoing data fast enough
4	evt_ack_err	0	rw	event	acknowledge error
5	evt_bit_err	0	rw	event	bit error (looped back transmit bit mismatch)
6	rec_fifo_nempty	0	rw	status	receive fifo is not empty
7	rec_fifo_high_water	0	rw	status	receive fifo elements > high water level
8	rec_fifo_full	0	rw	status	receive fifo is full
9	cmd_fifo_empty	1	rw	status	transmit fifo is empty
10	cmd_fifo_low_water	1	rw	status	transmit fifo elements <= low water level
11	cmd_fifo_nfull	1	rw	status	transmit fifo is not full
12	scl_stretch	0	rw	status	scl is stretched by slow slave

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag
0: do not change flag

0x24		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
12:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x28		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to enable

0x2A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to disable

0x2C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
3:0	vmax	13	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x2E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
3:0	vno	13	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.16 I²C Slave

7.16.1 Description

This module implements an I²C Slave module. It can be used as a host interface for direct ASIC memory access by an external I²C master when configured with 2 corresponding DMA channels.

7.16.2 Features

- core state dependent slave-address-byte acknowledge
  - allows I²C access handling while system is coming up from standby without loss of data
- 16 bit extended address mode (device select = 1010 XXX R)
- 10 bit address mode (device select = 11110 AA R)
- 7 bit address mode (device select = AAAAAAA R)
  - addresses 0 to 7, 80 to 87, 120 to 127 are reserved
- I²C protocol standard compliant
  - up to 1 Mbit
  - automatic clock stretching
- supported access modes:
  - random access read / write modes
  - current address read/write modes
  - sequential read/write modes
- configurable memory access base address
- DMA request interface compliant to set up an ASIC direct memory access host interface

7.16.3 Register Interface

0x00		CONTROL		
bits	**name**	**default**	**rw**	**description**
0	enable	0	rw	0: I²C slave interface is disabled 1: I²C slave interface is enabled Note: enable = 0 clears interface FIFO's (as long as the interface is disabled, the FIFO's are kept clean)
1	common_base	1	rw	base address mode 0 : separate base addresses for source and destination 1 : common base address for source and destination

0x02		BASE_ADDR_SRC		
bits	**name**	**default**	**rw**	**description**
15:0	addr	0	rw	memory source / common access base address Note: Base address bits 7:0 will be forced to zero in 10 and 16 bit addressing modes !

0x04		BASE_ADDR_DST		
bits	**name**	**default**	**rw**	**description**
15:0	addr	0	rw	memory destination access base address Note: Base address bits 7:0 will be forced to zero in 10 and 16 bit addressing modes !

0x06		TX_DATA		
bits	**name**	**default**	**rw**	**description**
7:0	data	0	w	transmit data

0x08		RX_DATA		
bits	**name**	**default**	**rw**	**description**
7:0	data	0	r	receive data / address

0x0A		STATUS		
bits	**name**	**default**	**rw**	**description**
1:0	rx_fifo_level	0	r	receive fifo fill level
3:2	tx_fifo_level	0	r	transmit fifo fill level
6:4	rx_data_type	0	r	current (next read) receive data type 000 : data 010 : low byte of ALL address 011 : high byte of ALL address 100 : low byte of SRC address 101 : high byte of SRC address 110 : low byte of DST address 111 : high byte of DST address

0x0C		DEVICE_SELECT		
bits	**name**	**default**	**rw**	**description**
7:0	mask	0	rw	device select byte compare mask (bit 0 is unused)
15:8	value	0	rw	device select byte compare value (bit 0 is unused) Note: Incoming device select byte will be AND masked and compared with this value. In case of matching the slave is accessed.

0x0E		TIMEOUT		
bits	**name**	**default**	**rw**	**description**
15:0	timeout	0xFFFF	rw	scl activity timeout [system clock cycles]

0x20		IRQ_STATUS			
bits	**name**	**default**	**rw**	**description**	**type**
0	evt_rx_overflow	0	rw	receive FIFO overflow	event
1	evt_sot	0	rw	start of transfer (after own device ID was recognized)	event
2	evt_eot	0	rw	end of transfer	event
3	evt_timeout	0	rw	scl activity timeout event	event
4	rx_fifo_laddr	0	rw	receive FIFO is not empty (low addr byte is pending)	status
5	rx_fifo_haddr	0	rw	receive FIFO is not empty (high addr byte is pending)	status
6	rx_fifo_lsaddr	0	rw	receive FIFO is not empty (low src addr byte is pending)	status
7	rx_fifo_hsaddr	0	rw	receive FIFO is not empty (high src addr byte is pending)	status
8	rx_fifo_ldaddr	0	rw	receive FIFO is not empty (low dst addr byte is pending)	status
9	rx_fifo_hdaddr	0	rw	receive FIFO is not empty (high dst addr byte is pending)	status
10	rx_fifo_data	0	rw	receive FIFO is not empty (data byte is pending)	status
11	evt_rdata_req	0	rw	read data request	event
12	tx_fifo_empty	0	rw	transmit FIFO is empty	status
13	tx_no_transfer	0	rw	currently no transfer in progress	status
14	scl_stretch	0	rw	scl is currently stretched (transmit data needed)	status

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag
0: do not change flag

0x24		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
14:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x28		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to enable

0x2A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
3:0	vno	0	w	vector number of interrupt to disable

0x2C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
3:0	vmax	15	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

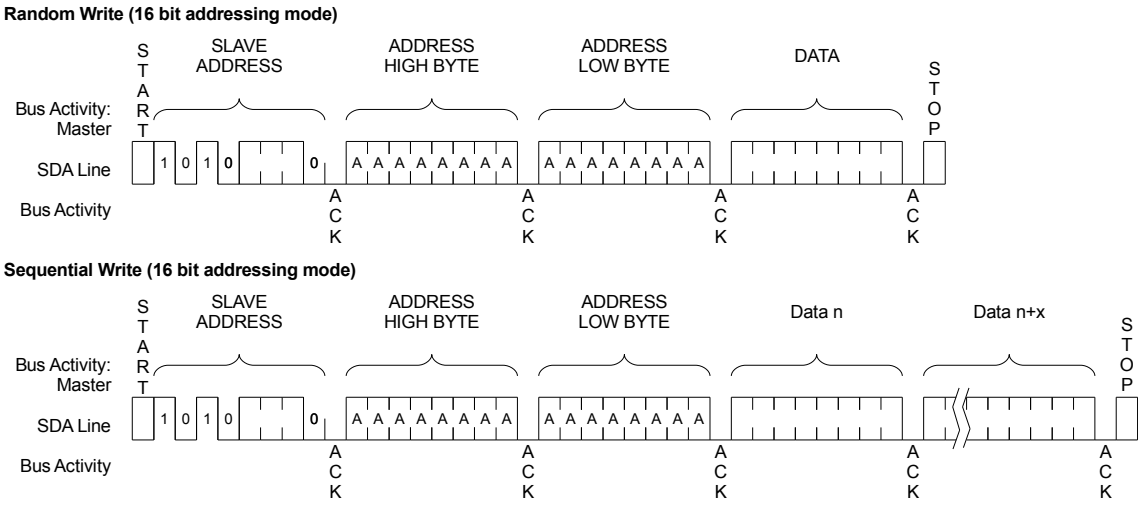
0x2E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
3:0	vno	15	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.17 I²C Protocol

The first byte sent by a I²C master is the standard-I²C-address. In which the first 7 bits are the address bits and the 8th bit is the direction bit (read/write). The following table gives an overview of the I²C address ranges.

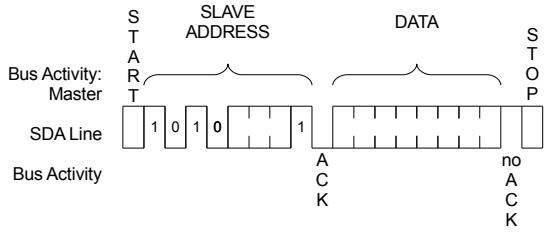
Slave Address (Device Select)	Description
0000 XXX	Reserved addresses
1010 XXX	Reserved for 16 Bit addressing mode
1111 0AA	Reserved for 10 Bit addressing mode
1111 1XX	Reserved addresses
all other addresses	Usable address space
X = don't care; A = upper bits of 10 bit address	

7.17.1 write (16 bit addressing mode)

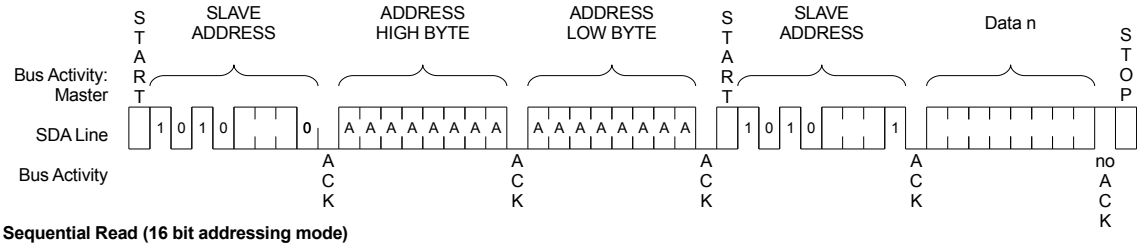


7.17.2 read (16 bit addressing mode)

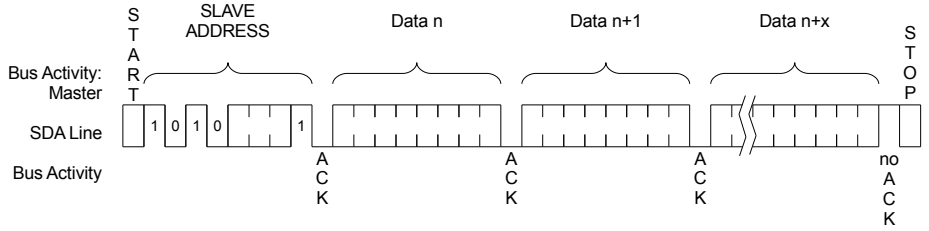
Current Address Read (16 bit addressing mode)



Random Read (16 bit addressing mode)

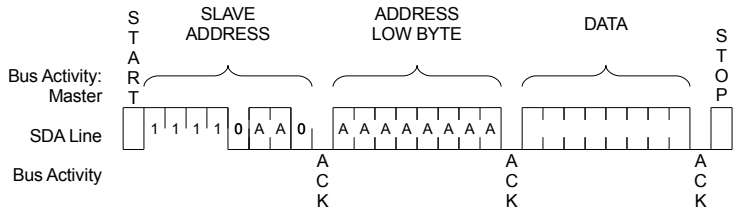


Sequential Read (16 bit addressing mode)

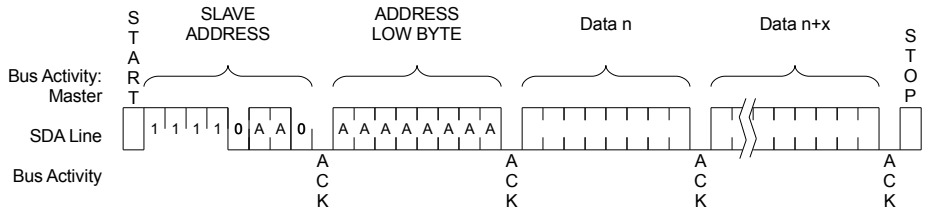


7.17.3 write (10 bit addressing mode)

Random Write (10 bit addressing mode)

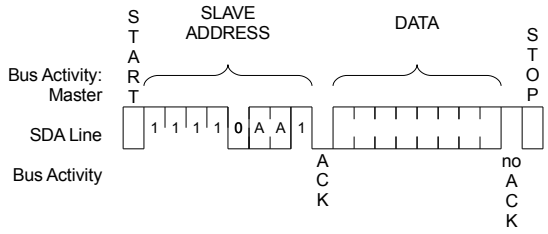


Sequential Write (10 bit addressing mode)

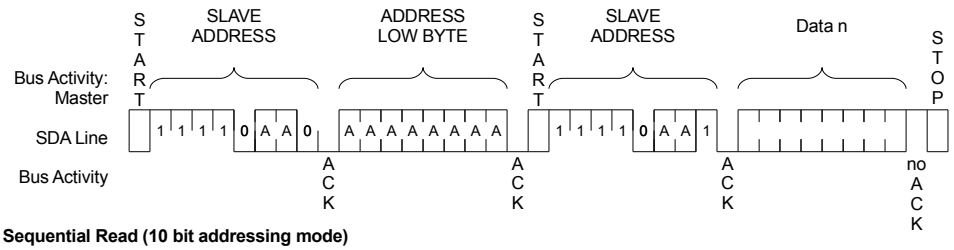


7.17.4 read (10 bit addressing mode)

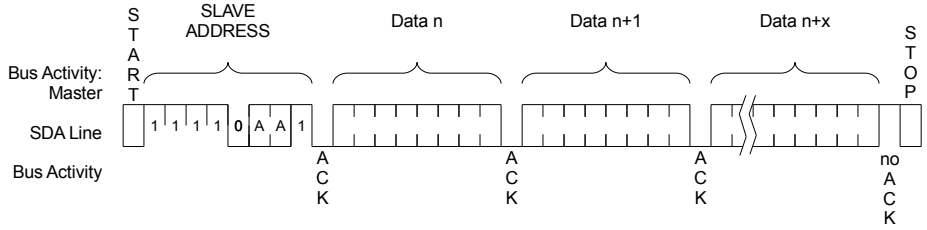
Current Address Read (10 bit addressing mode)



Random Read (10 bit addressing mode)

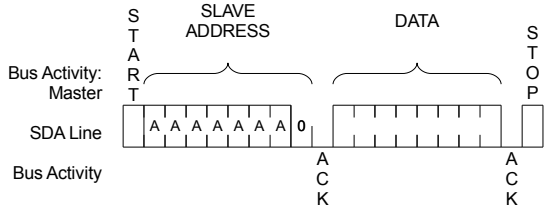


Sequential Read (10 bit addressing mode)

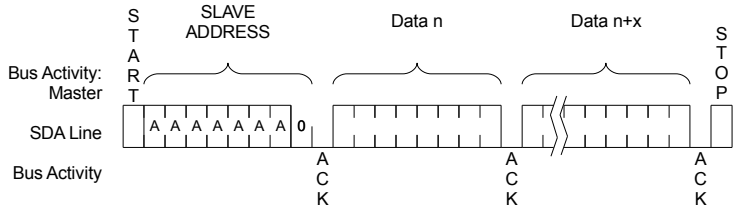


7.17.5 write (7 bit addressing mode)

Random Write (7 bit addressing mode)

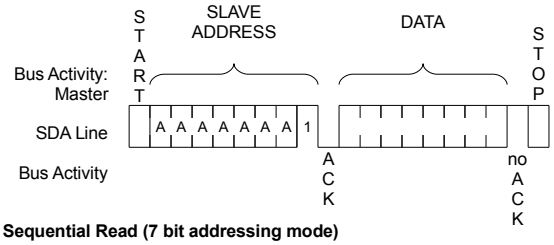


Sequential Write (7 bit addressing mode)

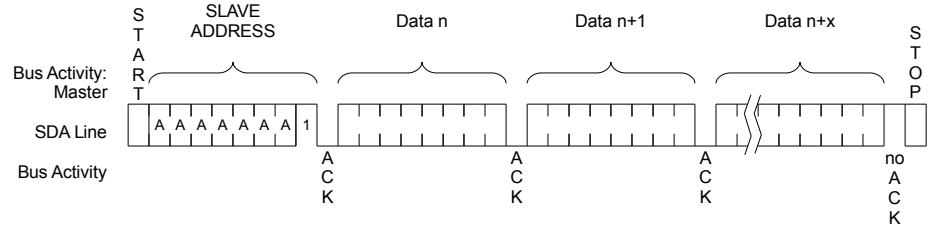


7.17.6 read (7 bit addressing mode)

Random Read (7 bit addressing mode)



Sequential Read (7 bit addressing mode)



7.18 GPIO Interface

7.18.1 Description

This module gives access to general purpose digital analog analog IO's.

7.18.2 Structure

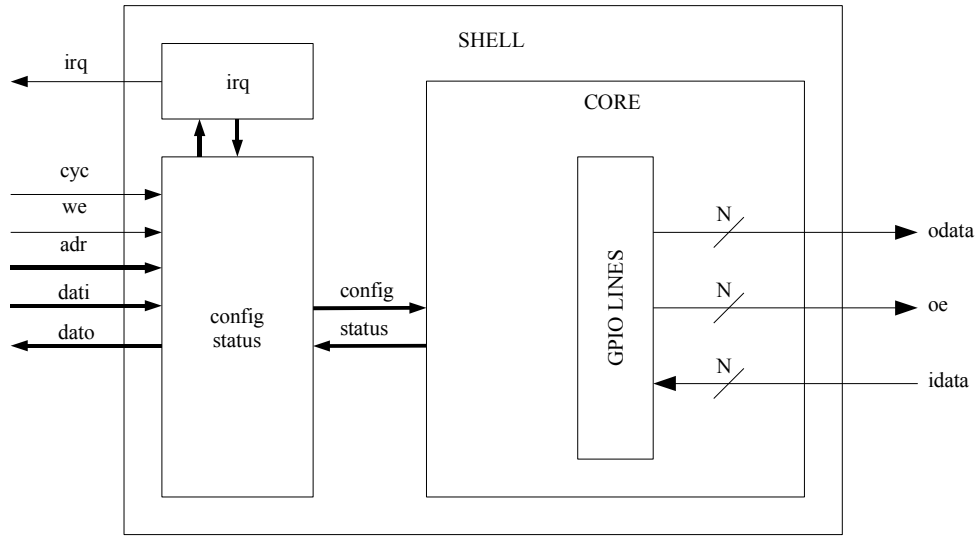


Figure 28: GPIO module structure

7.18.3 Physical Interface

Name	Dir	Description
bus_*	in	bus interface
odata	out	output data
oen	out	output enable
idata	in	input data
irq	out	

7.18.4 Features

- 16 GPIO lines
- digital push pull and open drain functionality
- digital IO toggle interrupt

7.18.5 Register Interface

0x00		DIRECTION		
bits	**name**	**default**	**rw**	**description**
0..15	direction	0	rw	0: input 1: output

0x02		DRIVER		
bits	**name**	**default**	**rw**	**description**
0..15	driver	0	rw	0: push pull 1: open drain

0x04		DATAOUT		
bits	**name**	**default**	**rw**	**description**
0..15	data	0	rw	data out

0x06		INPUT_ENABLE		
bits	**name**	**default**	**rw**	**description**
0..15	enable	0	rw	GPIO IO input enable 0: input disabled (in this case ASIC input must not be pulled up or pulled down when unconnected) 1: input enabled (when used as input ASIC pin must be driven)

0x08		DATAIN		
bits	**name**	**default**	**rw**	**description**
0..15	data	0	r	data in

0x20		IRQ_STATUS			
bits	**name**	**default**	**rw**	**type**	**description**
15:0	evt_edge	0	rw	event	an edge was detected on GPIO when configured as input write '1' to bit clears event read access clears all events

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag
0: do not change flag

0x24		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
15:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x28		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
4:0	vno	0	w	vector number of interrupt to enable

0x2A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
4:0	vno	0	w	vector number of interrupt to disable

0x2C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
4:0	vmax	16	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x2E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
4:0	vno	16	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.19 AD Control Module (AD_CTRL)

7.19.1 Description

This module implements 5 ADC channels. The AD_CTRL module is controlled by the UFO2 library software. This part of the library software is under development and the AD_CTRL module section will be updated soon.

7.19.2 Features

- channel multiplexed Sigma Delta ADC

  10 - 16 bit digital integral ADC filter for all channels (bit width configurable)

  SINC2 ADC filter

- ASIC temperature measurement
- 3 external temperature sensor measurement channels (RTD)
- differential RTD_45 - RTD_23 measurement
- battery voltage measurement channel
- A_IN pin voltage measurement

7.20 Loop Control Module (UL_CTRL)

7.20.1 Description

This module implements control and status functionality of the analog parts concerning the ultrasonic measurement loop. The UL_CTRL module is controlled through the UFO2 library software.

7.20.2 Library actions

- A **HF_CLK calibration** can be issued.
- An **ultrasonic measurement** can be issued.

7.20.3 Library configurations

- The **LOOPC** parameter can be set to between 1 and 65535. LOOPC defines the number of sing-around loops. Depending on the measuring mode LOOPC can be set any value between 1 and 65535. Normally LOOPC would be set to between 1 and 50.
- The **SMARK** parameter can be set to between 1 and 65535 16MHz clock cycles. SMARK = 1600 would for example correspond to 100µs. SMARK defines the beginning of the time window in which the received signal should appear. SMARK starts counting at the moment of excitation on the sending transducer. SMARK starts counting at the moment of excitation on the sending transducer. In power down mode the analog modules are turned off until SMARK. In the case of power down 3µs should be subtracted from SMARK in order to allow the analog modules to settle after power on.
- The **SSPACE** parameter can be set to between 1 and 65535 16MHz clock cycles. SSPACE = 1600 would for example correspond to 100µs.The SSPACE register defines the end of the time window in which the received signal should appear. SSPACE starts counting at the moment of SMARK. If SSPACE elapse without a signal being detected a time-out is issued.

7.20.4 Library readouts

- The result of a HF_CLK calibration can be read out.
- The measurement result of an ultrasonic measurement can be read out.
- The SSPACE time-out status can be read out.
- The HF_CLK calibration status (finished / not finished) can be read out.
- The ultrasonic measurement status (finished / not finished) can be read out.
- The ultrasonic excitation status (transducer excited / not excited) can be read out.

7.21 Auto Gain Control Module (AG_CTRL)

7.21.1 Description

This module implements the analog control and status functionality concerning the signal conditioning. The AG_CTRL module is controlled through the UFO2 library software.

7.21.2 Library actions

- A gain calibration can be issued. The procedure of the calibration is described below.

7.21.3 Library configurations

- The **R1** parameter can be set to between 3 and 31. Stage one of the gain control is settable and can perform automatic gain adjustments. R1 defines the gain in stage one together with gm1. If an automatic adjustment of the automatic gain control is issued R1 will likely get a new value.
- The **gm1** parameter can be set to between 1 and 15. gm1 defines the gain in stage one together with R1. If an automatic adjustment of the automatic gain control is issued gm1 will change but only if R1 reach an end value.
- The **R2** parameter can be set to between 1 and 31. Stage two of the gain control is settable and fixed. R2 defines the gain in stage two. If an automatic adjustment of the automatic gain control is issued R2 still stays fixed.
- The **PTRIG** parameter can be set to between 1 and 31. PTRIG defines the threshold value defining the arrival of the signal. When PTRIG is exceeded the signal detection and time measurement functions in the UL_CTRL module are armed. PTRIG should be set higher than the signal noise.
- The **PTRIG_LOW** parameter can be set to between 1 and 31. PTRIG_LOW defines a threshold value used by the automatic gain control. The automatic gain control strives to exceed PTRIG_LOW. PTRIG_LOW should be set higher than PTRIG.
- The **PTRIG_HIGH** parameter can be set to between 1 and 31. PTRIG_HIGH defines a last threshold value. The status of PTRIG_HIGH is included in the status read out but it is not used by the automatic gain control. PTRIG_HIGH is preferably set higher than PTRIG_LOW.

7.21.4 Library readouts

- The result of a gain calibration can be read out.
- The gain calibration status (finished / not finished) can be read out.
- The PTRIG_LOW status (level exceeded / not exceeded) can be read out. The status from the gain calibration and the ultrasonic measurements in both directions can be read.
- The PTRIG_HIGH status (level exceeded / not exceeded) can be read out. The status from the gain calibration and the ultrasonic measurements in both directions can be read.

7.21.5 Gain calibration procedure

The Automatic Gain Calibration is a process of finding ***R1*** and ***gm1*** so that the gain of the AG module is cor-

rect, which means that the amplitude of an incoming signal is higher than **ptrig_low** and preferably lower than **ptrig_high**. Figure 29 shows the desired signal amplitude of a highly simplified signal.

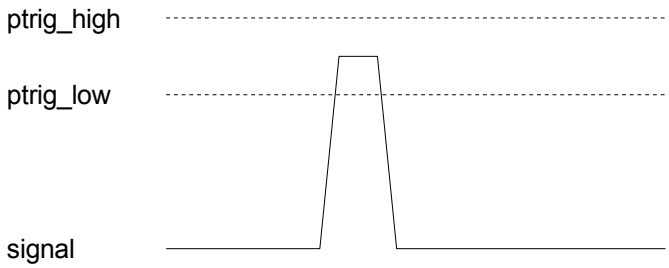


Figure 29: AG Control, level triggers and incoming signal

The search is done linearly, starting at the present ***R1*** and ***gm1*** values. The sending transducer is excited and the statuses of PTRIG_LOW and PTRIG_HIGH are observed. The algorithm decides based on the statuses if the gain need to be increased (by increasing R1 or gm1) or decreased (by decreasing R1 or gm1). After SMARK+SSPACE the sending transducer is again exited. This procedure is continued until PTRIG_LOW is just exceeded. The algorithm is summarized in table 2.

The procedure starts with R1. If the amplitude of the signal is lower than PTRIG_LOW, R1 is increased by one, until the PTRIG_LOW level is reached (see figure 30). PTRIG_HIGH is not considered for stopping the search.

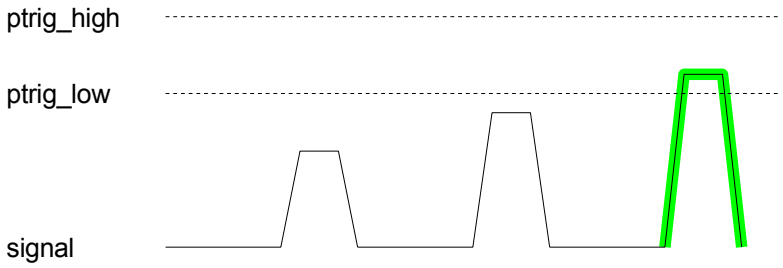


Figure 30: AG_CTRL, R1 search, gain too low

If the amplitude of the signal is higher than PTRIG_LOW, R1 is decreased by one, until the amplitude is lower than PTRIG_LOW. The second last R1 setting is chosen. This is illustrated in figure 107.

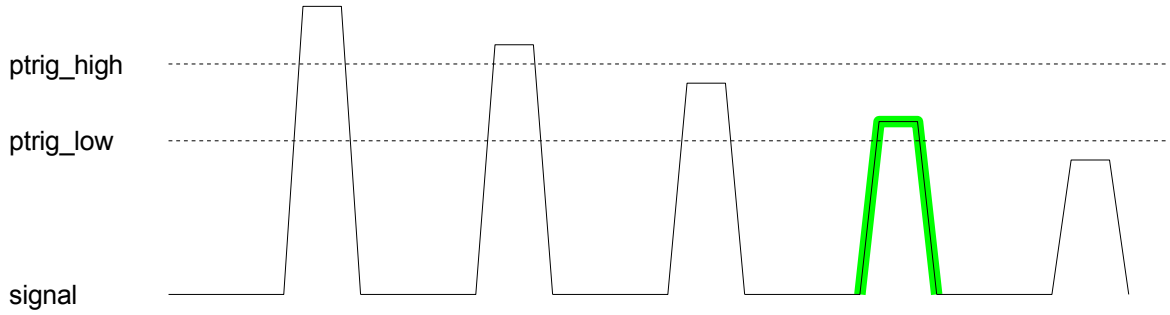


Figure 31: AG_CTRL, R1 search, gain too high

PTRIG_HIGH status at start	PTRIG_LOW status at start	action
0	0	Increase R1 until PTRIG_LOW = 1
0	1	Decrease R1 until PTRIG_LOW = 0, take previous R1
1	0	Increase R1 until PTRIG_LOW = 1
1	1	Decrease R1 until PTRIG_LOW = 0, take previous R1

Table 2: Search decisions and stopping conditions

If the range of R1 is not sufficient to reach PTRIG_LOW the procedure continues and performs the same search now with gm1.

The total dynamic of stage one of the amplifier is about 1:29. The dynamic range of stage 1 and stage 2 is about 1:500.

7.22 Transmit Control Module (TX_CTRL)

7.22.1 Description

This module implements the analog control and status functionality concerning the ultrasonic transducer interface. The AG_CTRL module is controlled through the UFO2 library software.

7.22.2 Library configurations

There is no support in the library software for external transducer drivers and the use of the MSEQ module. This is under development at this point. Measurement mode 1 to 4 all refers to the internal transducer drivers.

- **Measuring mode1:** This mode is suitable in applications with large transducer distances, gas applications and applications where near zero flow performance is less critical. This mode uses two ultrasonic transducers. The LOOPC register is preferably set to 1 (single shot) or 2. The upstream transducer is connected to TX0 single ended. The downstream transducer is connected to TX3.

- **Measuring mode 2:** This mode is intended to be used in symmetrical 2-path meters with four transducers. Measurements are performed interlinked in the two paths – two measurement loops in the first path, the next two loops in the second path, the next two back in the first and so on. Equally many sing-around loops are performed in each path and the resulting time measurement is an average of the two paths. The LOOPC register is preferably set to multiples of 4. The transducers are connected single ended, the upstream transducer in path 1 to TX0, the downstream one to TX3, the upstream transducer in path2 to TX1 and the downstream one to TX2.

- **Measuring mode 3:** This mode is used for LOOPC higher than 2 in 1-path (2 transducers) applications. The LOOPC register is preferably set to n*8+4 (n≥0). The upstream transducer is connected to TX0 and TX1 differentially and the downstream one to TX2 and TX3. TX0 and TX3 connects to the positive side.

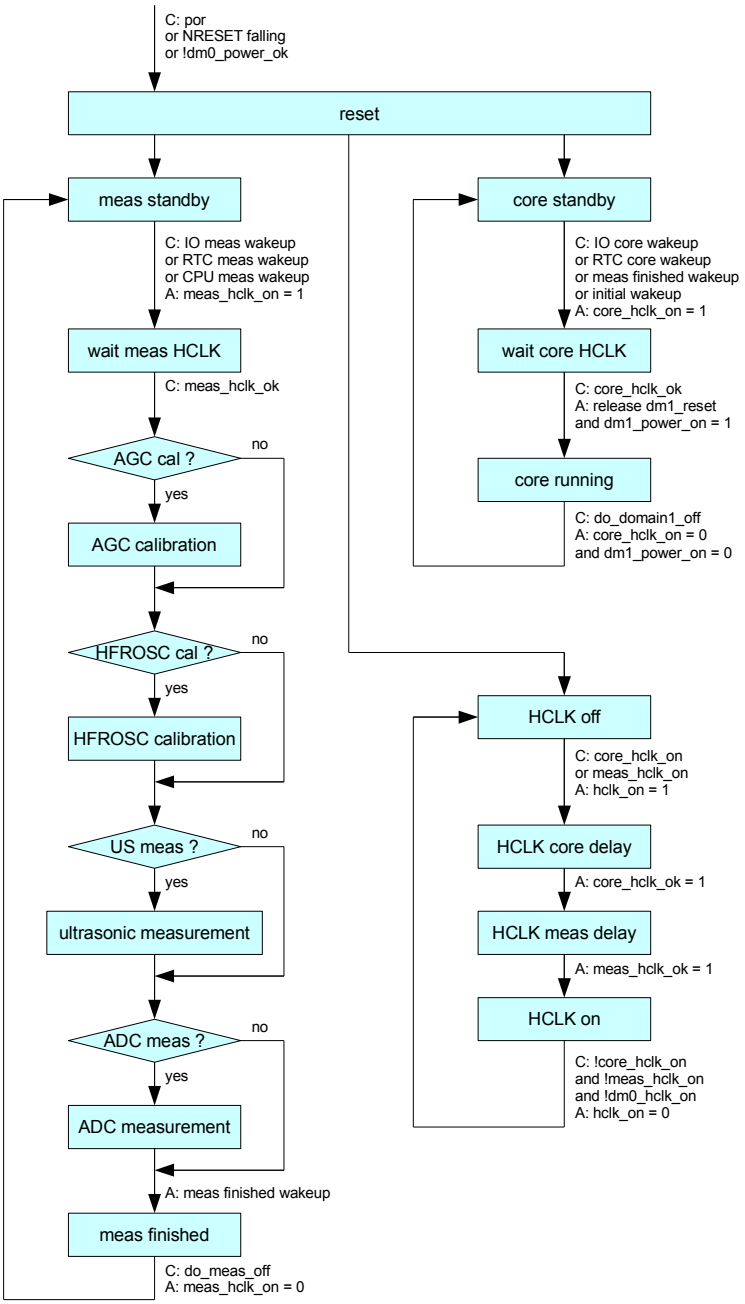
- **Measuring mode 4:** This mode is used for distance measurements using only one transducer that simultaneously sends and receives sound . This mode is under development.

7.23 System State Module

7.23.1 Description

This module implements some system state control and status functionality.

7.23.2 System State Diagram



7.23.3 Features

- core and measurement wake-up enable configuration and status
- RTC wake-up configuration
- module / interface clock control (on / off)
- measurement configuration
- software triggered measurement
- software triggered standby

7.23.4 Register Interface

0x00		CORE_WAKEUP_ENABLE		
bits	**name**	**default**	**rw**	**description**
0	rxd0	0	rw	enable UART instance 0 RXD to wake-up core
1	rxd1	0	rw	enable UART instance 1 RXD to wake-up core
2	sdi	0	rw	enable SPI input data SDI to wake-up core
3	csb	0	rw	enable SPI chip select CSB to wake-up core
4	sck	0	rw	enable SPI clock SCK to wake-up core
5	scl	0	rw	enable I2C_SLAVE clock SCL to wake-up core
6	sda	0	rw	enable I2C_SLAVE data SDA to wake-up core
7	gpio10	0	rw	enable GPIO10 to wake-up core
8	gpio11	0	rw	enable GPIO11 to wake-up core
9	gpio12	0	rw	enable GPIO12 to wake-up core
10	gpio13	0	rw	enable GPIO13 to wake-up core
11	gpio14	0	rw	enable GPIO14 to wake-up core
12	gpio15	0	rw	enable GPIO15 to wake-up core
13	rtc_trigger	0	rw	enable RTC periodic trigger to wake-up core
14	meas_finished	1	rw	enable finish of measurement to wake-up core
15	power_up	1	rw	enable ASIC power up to wake-up core (always on)

0x02		CORE_WAKEUP_STATUS		
bits	**name**	**default**	**rw**	**description**
0	rxd0	0	r	core wake-up done by UART instance 0 RXD
1	rxd1	0	r	core wake-up done by UART instance 1 RXD
2	sdi	0	r	core wake-up done by SPI input data SDI
3	csb	0	r	core wake-up done by SPI chip select CSB
4	sck	0	r	core wake-up done by SPI clock SCK
5	scl	0	r	core wake-up done by I2C_SLAVE clock SCL
6	sda	0	r	core wake-up done by I2C_SLAVE data SDA
7	gpio10	0	r	core wake-up done by GPIO10
8	gpio11	0	r	core wake-up done by GPIO11
9	gpio12	0	r	core wake-up done by GPIO12
10	gpio13	0	r	core wake-up done by GPIO13
11	gpio14	0	r	core wake-up done by GPIO14
12	gpio15	0	r	core wake-up done by GPIO15
13	rtc_trigger	0	r	core wake-up done by RTC periodic trigger
14	meas_finished	0	r	core wake-up done by finish of measurement
15	power_up	0	r	core wake-up done by ASIC power up (always on)

0x04		MEAS_WAKEUP_ENABLE		
bits	**name**	**default**	**rw**	**description**
0	rxd0	0	rw	enable UART instance 0 RXD to wake-up measurement
1	rxd1	0	rw	enable UART instance 1 RXD to wake-up measurement
2	sdi	0	rw	enable SPI input data SDI to wake-up measurement
3	csb	0	rw	enable SPI chip select CSB to wake-up measurement
4	sck	0	rw	enable SPI clock SCK to wake-up measurement
5	scl	0	rw	enable I2C_SLAVE clock SCL to wake-up measurement
6	sda	0	rw	enable I2C_SLAVE data SDA to wake-up measurement
7	gpio10	0	rw	enable GPIO10 to wake-up measurement
8	gpio11	0	rw	enable GPIO11 to wake-up measurement
9	gpio12	0	rw	enable GPIO12 to wake-up measurement
10	gpio13	0	rw	enable GPIO13 to wake-up measurement
11	gpio14	0	rw	enable GPIO14 to wake-up measurement
12	gpio15	0	rw	enable GPIO15 to wake-up measurement
13	rtc_trigger	0	rw	enable RTC periodic trigger to wake-up measurement

0x06		MEAS_WAKEUP_STATUS		
bits	**name**	**default**	**rw**	**description**
0	rxd0	0	r	measurement wake-up done by UART instance 0 RXD
1	rxd1	0	r	measurement wake-up done by UART instance 1 RXD
2	sdi	0	r	measurement wake-up done by SPI input data SDI
3	csb	0	r	measurement wake-up done by SPI chip select CSB
4	sck	0	r	measurement wake-up done by SPI clock SCK
5	scl	0	r	measurement wake-up done by I2C_SLAVE clock SCL
6	sda	0	r	measurement wake-up done by I2C_SLAVE data SDA
7	gpio10	0	r	measurement wake-up done by GPIO10
8	gpio11	0	r	measurement wake-up done by GPIO11
9	gpio12	0	r	measurement wake-up done by GPIO12
10	gpio13	0	r	measurement wake-up done by GPIO13
11	gpio14	0	r	measurement wake-up done by GPIO14
12	gpio15	0	r	measurement wake-up done by GPIO15
13	rtc_trigger	0	r	measurement wake-up done by RTC periodic trigger

0x08		CLEAR_WAKEUP		
bits	**name**	**default**	**rw**	**description**
0	core	0	w	clear core (domain1) wake-up flags
1	meas	0	w	clear measurement wake-up/trigger flags

0x0A		STATE_CONTROL		
bits	**name**	**default**	**rw**	**description**
0	do_meas	0	w	trigger measurement
1	domain1_off	0	w	switch off power domain1
2	meas_off	0	w	switch measurement part to standby state (must only be used when measurement has finished)

0x0C		HCLK_DELAY		
bits	**name**	**default**	**rw**	**description**
7:0	core	160	rw	HCLK stabilization delay to use for core operation [LCLK_CYCLE]
15:8	meas	32	rw	HCLK stabilization delay to use for measurement [LCLK_CYCLE] Note: This delay is additional to core delay.

0x0E		CONTROL		
bits	**name**	**default**	**rw**	**description**
0	refclk_cal	0	rw	enable the HFROSC calibration with the REFCLK for next measurement wake-up
1	agc_cal	0	rw	enable AGC calibration for next measurement wake-up
2	adc_meas	0	rw	enable ADC measurement for next measurement wake-up
3	us_meas	1	rw	enable ultrasonic measurement for next measurement wake-up
6	sram_parity_on	0	rw	enable SRAM parity reset
7	drv_strength	0	rw	digital pad drive strength (see parameter DIO_IOx_Dx) 0 : low drive strength 1 : high drive strength
10:8	rtc_trigger	6	rw	RTC wake-up trigger config trigger frequency [Hz] = 2^(7-rtc_trigger) 7 : 1 Hz 6 : 2 Hz .... 0 : 128 Hz

0x10		MODULE_ENABLE		
bits	**name**	**default**	**rw**	**description**
0	pulse	0	rw	enable **LCLK** in PULSE module
1	pwm	0	rw	enable **LCLK** in PWM module
2	wdog	0	rw	enable CPU watchdog module note: disabling wdog has no effect if it has been started
3	swtimer	0	rw	enable timer module note: disabling swtimer has no effect as long as at least one of the timers is enabled and has a non-zero value
4	gpio	0	rw	enable GPIO module
5	spi	0	rw	enable SPI module
6	uart0	0	rw	enable UART instance 0 note: disabling uart0 has no effect as long as it is either sending or receiving data
7	uart1	0	rw	enable UART instance 1 note: disabling uart1 has no effect as long as it is either sending or receiving data
8	i2c_master	0	rw	enable I2C master module note: disabling i2c_master has no effect as long as there is a transfer in progress
9	i2c_slave	0	rw	enable I2C slave module
10	flash_ctrl	0	rw	enable FLASH control module
11	iomux_ctrl	0	rw	enable IOMUX control module
12	ad_ctrl	0	rw	enable AD_CTRL module
13	ag_ctrl	0	rw	enable AG_CTRL module
14	ul_ctrl	0	rw	enable UL_CTRL module
15	tx_ctrl	0	rw	enable TX_CTRL module

0x12		HLCLK_OUT_DIVIDER		
bits	**name**	**default**	**rw**	**description**
2:0	hclk_div	4	rw	Divide the HCLK clock at output pin HCLK : output_frequency = HCLK / (2^{hclk_div}) Note: 2nd function has to be selected in IOMUX_CNTRL
6:4	lclk_div	1	rw	Divide the LCLK clock at output pin LCLK: output frequency = LCLK / (2^{lclk_div}) Note: 2nd function has to be selected in IOMUX_CNTRL

0x14		CALIBRATION		
bits	**name**	**default**	**rw**	**description**
2:0	analog_cal_ref_vbg	4	rw	Note: read only when lock_cal is set
5:3	analog_cal_ref_v	4	rw	Note: read only when lock_cal is set
9:6	analog_cal_ref_i	15	rw	Note: read only when lock_cal is set

0x16		CALIBRATION_LOCK		
bits	**name**	**default**	**rw**	**description**
0	lock_cal	0	rw	Lock calibration values (cal values are changed to read only when lock_cal was set)
1	force_pw_ok	0	rw	force "blanked" power watch status bits to OK state 0 : not forced 1 : forced

0x18		RESET_STATUS		
bits	**name**	**default**	**rw**	**description**
0	vbat_vref	0	rw	power on reset (write access clears all bits)
1	vddd	0	r	digital core power watch
2	vddio	0	r	IO power watch
3	sram_parity	0	r	SRAM parity error
4	wdog0L	0	r	watchdog 0 (LCLK) domain 0 and domain 1 reset
5	wdog0H	0	r	watchdog 0 (HCLK) domain 0 and domain 1 reset
6	wdog1	0	r	watchdog 1 domain 1 reset
7	nreset	0	r	nreset pad was active

0x1A		POWER_WATCHES		
bits	**name**	**default**	**rw**	**description**
0	pw_vbat_vref_ok	1	r	current power watch status
1	pw_vdda_ok	1	r	current power watch status
2	pw_vddd0_ok	1	r	current power watch status
3	pw_vddio_ok	1	r	current power watch status
4	pw_vddtx_ok	1	r	current power watch status
5	pw_nb_vbat_ok	1	r	current power watch status (non blanked)
6	pw_nb_vref_ok	1	r	current power watch status (non blanked)
7	pw_nb_vbat_vref_ok	1	r	current power watch status (non blanked)
8	pw_nb_vdda_ok	1	r	current power watch status (non blanked)
9	pw_nb_vddd0_ok	1	r	current power watch status (non blanked)
10	pw_nb_vddio_ok	1	r	current power watch status (non blanked)
11	pw_nb_vddtx_ok	1	r	current power watch status (non blanked)

0x1C		NEEDS_HCLK_STATUS		
bits	**name**	**default**	**rw**	**description**
0	lclk_div	0	r	LCLK divider needs HCLK for sync purpose
1	hclk_div	0	r	HCLK divider needs HCLK for sync or functional purpose
2	pulse	0	r	PULSE module needs HCLK for sync or functional purpose
3	pwm	0	r	PWM module needs HCLK for sync or functional purpose
4	rtc	0	r	RTC needs HCLK for sync purpose
5	wdog0L	0	r	WDOG 0 (LCLK) needs HCLK for sync purpose

0x1E		LCLK_CTRL		
bits	**name**	**default**	**rw**	**description**
0	bypass	0	rw	bypass internal LCLK oscillator
1	ix0p5	0	rw	half LCLK oscillator current
2	ix2	0	rw	double LCLK oscillator current

0x20		LCLK_LEVEL		
bits	**name**	**default**	**rw**	**description**
0	val	0	r	current LCLK level

0x40		IRQ_STATUS			
bits	**name**	**default**	**rw**	**description**	**type**
0	pw_vdda	0	rw	analog power watch triggered	event
1	pw_vddtx	0	rw	TX power watch triggered	event
2	core_wakeup	0	rw	core wake-up change	event
3	meas_wakeup	0	rw	measurement wake-up change	event
4	meas_finished	0	rw	measurement finished	event

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag
0: do not change flag

0x44		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
4:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x48		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
2:0	vno	0	w	vector number of interrupt to enable

0x4A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
2:0	vno	0	w	vector number of interrupt to disable

0x4C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
2:0	vmax	5	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x4E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
2:0	vno	5	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.24 PWM Interface

7.24.1 Description

This module implements 2 PWM channels.

7.24.2 Features

- base clock selection
- 8 bit clock prescaler
- 16 bit PWM pulse resolution

7.24.3 Register Interface

0x00		CONTROL		
bits	**name**	**default**	**rw**	**description**
0	clk_sel	0	rw	0: LCLK is selected as PWM base clock 1: HCLK is selected as PWM base clock

0x02		PRESCALER		
bits	**name**	**default**	**rw**	**description**
7:0	scale	0	rw	base clock prescaler value

0x04		PERIOD		
bits	**name**	**default**	**rw**	**description**
15:0	period	0	rw	PWM period length value

0x06 0x08		PULSE0 PULSE1		
bits	**name**	**default**	**rw**	**description**
15:0	pulse	0	rw	PWM pulse length value

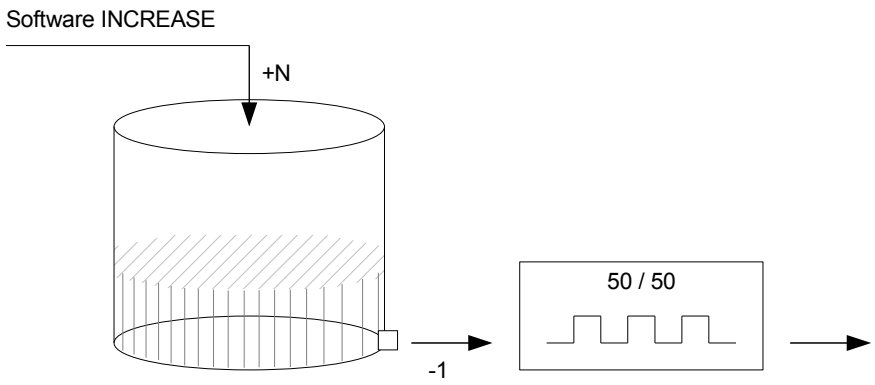
0x0A		UPDATE		
bits	**name**	**default**	**rw**	**description**
0	update	0	w	update PWM period and pulse lengths to internal values

7.25 Pulse Interface

7.25.1 Description

This module implements a pulse interface with an accumulator which can be increased by writing the INCREASE register. Output pulses are generated as long as the accumulator value is not zero. Each pulse decrements the accumulator by 1. If the CPU causes an accumulator overflow the accumulator value will be saturated to its maximum value and an interrupt will be generated.

Pulse Interface



7.25.2 Features

- base clock selection
- 8 bit clock prescaler
- 16 bit accumulator
- 50% duty cycle pulse waveform of half prescaled "base clock" frequency
  - pulse frequency = base clock frequency / (2 * prescaler)

7.25.3 Register Interface

0x00		CONTROL		
bits	**name**	**default**	**rw**	**description**
0	clk_sel	0	rw	0: LCLK is selected as pulse module base clock 1: HCLK is selected as pulse module base clock

0x02		PRESCALER		
bits	**name**	**default**	**rw**	**description**
7:0	scale	0	rw	base clock prescaler value

0x04		INCREASE		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	w	accumulator increase value

0x06		ACCU		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	r	current accumulator value

0x08		STATUS		
bits	**name**	**default**	**rw**	**description**
0	busy	0	r	0 : ready : accumulator read value is valid 1 : busy : accumulator read value may be invalid because of ongoing value sync

0x20		IRQ_STATUS			
bits	**name**	**default**	**rw**	**description**	**type**
0	evt_overflow	0	rw	accumulator overflow was detected	event
1	accu_is_zero	0	rw	accumulator has zero value ("underflow")	status

NOTE: IRQ_STATUS read: unmasked status of all pending irqs
1: pending
0: no request

NOTE: IRQ STATUS write: clear event flags
1: clear related flag
0: do not change flag

0x24		IRQ_MASK		
bits	**name**	**default**	**rw**	**description**
1:0	mask	0	rw	enable irq source 1: enabled 0: disabled

0x28		IRQ_VENABLE		
bits	**name**	**default**	**rw**	**description**
0	vno	0	w	vector number of interrupt to enable

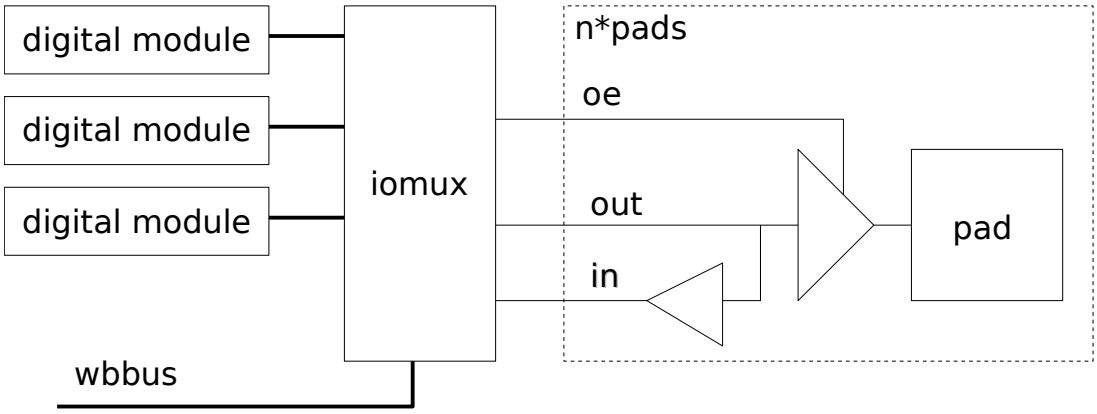
0x2A		IRQ_VDISABLE		
bits	**name**	**default**	**rw**	**description**
0	vno	0	w	vector number of interrupt to disable

0x2C		IRQ_VMAX		
bits	**name**	**default**	**rw**	**description**
1:0	vmax	2	rw	needed for nested interrupt support software writes current vector number to this register, so only interrupts with higher priority (lower vector number) can nest

0x2E		IRQ_VNO		
bits	**name**	**default**	**rw**	**description**
1:0	vno	2	rw	**read:** vector number of enabled pending interrupt with highest priority (smallest vector number). when no irq is pending the first unused irq number is returned. **write:** vector number of interrupt event to clear

7.26 IOMux Control Module

7.26.1 Operating Environment



7.26.2 Description

This module control the IO function selection.

7.26.3 Physical Interface

Signal	Dir	Description
bus_*	IO	internal bus system

7.26.4 Features

- Test Enable Signal selects JTAG Interface

7.26.5 Register Interface

0x00		FUNC_SELECT		
bits	**name**	**default**	**rw**	**description**
15:0	select	0	rw	GPIO IO function select 0 : 1st function 1 : 2nd function see Pin Description for details

0x02		PULSE_SELECT		
bits	**name**	**default**	**rw**	**description**
0	lclk	0	rw	2nd level GPIO IO function select for pad LCLK 0: 1st function or 2nd function selected by FUNC_SELET 1: PULSE module output on LCLK pad **This select has priority over FUNC_SELECT.**
1	hclk	0	rw	2nd level GPIO IO function select for pad HCLK 0: 1st function or 2nd function selected by FUNC_SELET 1: PULSE module output on HCLK pad **This select has priority over FUNC_SELECT.**

0x04		CLK_OUT_SELECT		
bits	**name**	**default**	**rw**	**description**
0	pwm0	0	rw	2nd level GPIO IO function select for pad PWM0 0: 1st function or 2nd function selected by FUNC_SELET 1: LCLK output on PWM0 pad **This select has priority over FUNC_SELECT.**
1	pwm1	0	rw	2nd level GPIO IO function select for pad PWM1 0: 1st function or 2nd function selected by FUNC_SELET 1: HCLK output on PWM1 pad **This select has priority over FUNC_SELECT.**

7.27 Real Time Clock Module

7.27.1 Description

This module implements a real time clock module. It is based on the LCLK clock.

7.27.2 Features

- 32 bit second counter

7.27.3 Register Interface

0x00 **0x02**		**TIME_LOW** **TIME_HIGH**		
bits	**name**	**default**	**rw**	**description**
15:0	val	0	rw	time stamp value [s]

0x04		**STATUS**		
bits	**name**	**default**	**rw**	**description**
0	busy	0	r	time set status (when a new time stamp was set by software) 0 : ready 1 : time stamp setting ongoing (time stamp set by software is not allowed in this case)
1	busy_read	0	r	time set status regarding reading back the time stamp 0 : ready to read back 1 : time stamp setting or back-synchronization ongoing

8 Package

8.1 Inscription on Package (tbd.)

D-Flow
UFO2-B1P
MSC800014
Year/week (for example 1236)

8.2 Package Dimensions of QFN56

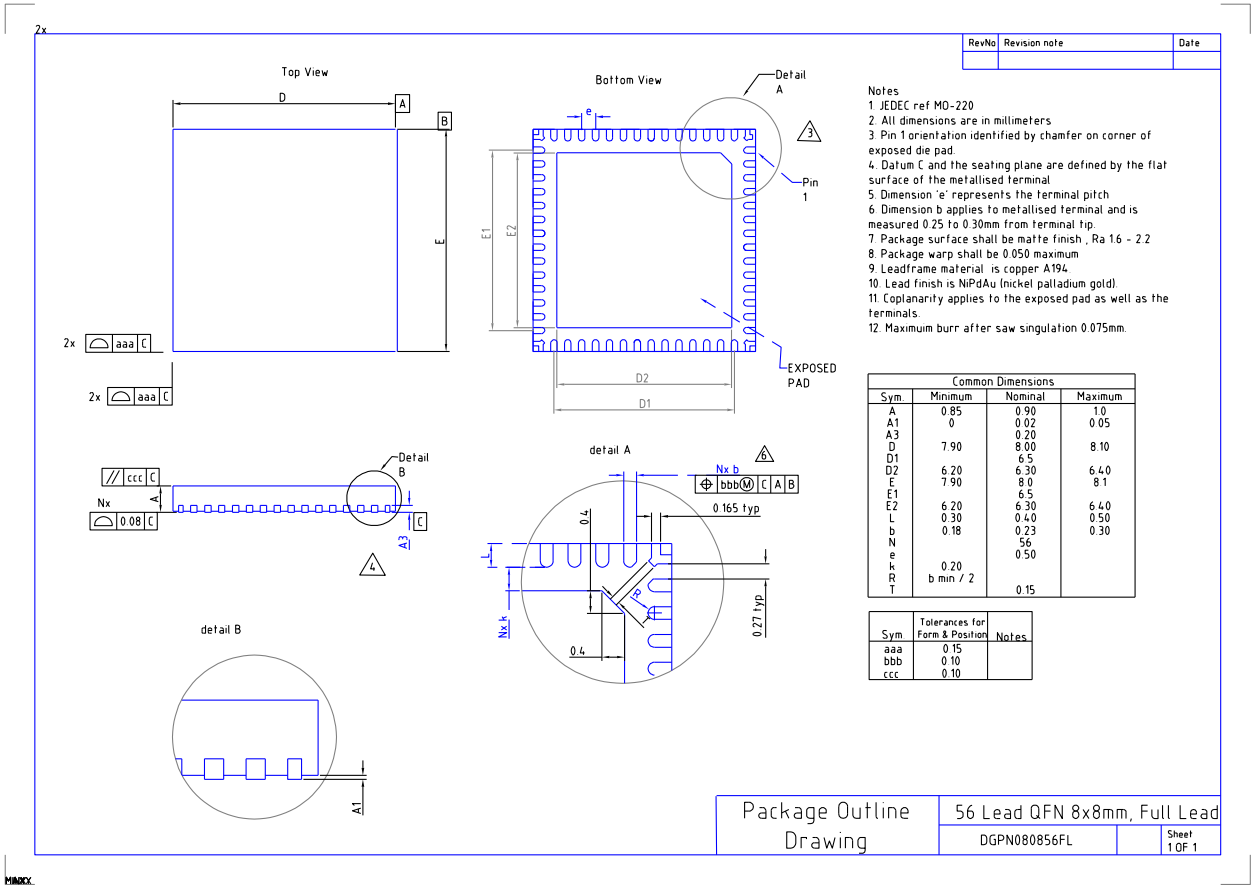


Figure 32: Package outline drawing

9 Index

Table of Content

Table of Figures