ScioSense

Datasheet

DS000587

AS6031

SoC for Ultrasonic Flow Meters

v1-00 • 2020-May-06

Content Guide

1 General Description

AS6031 is an ultrasonic flow converter for the next generations of ultrasonic water and heat meters. It is highly integrated and is based on the TDC-GP30 platform. It uses the same high-performant front-end for driving the transducers and processing the receive signal to extract the time of flight information. An additional programmable amplifier allows handling weaker receive amplitudes. Additional features for phase-modulated hit identification give even more robustness to the first hit level detection. The integrated low-power CPU can use 4k of NVRAM and 4k of ROM code to do the flow calculation on chip.

AS6031 simplifies the design of ultrasonic water and heat meters, combining the precise measurement and the complex flow calculation. It provides outstanding low-flow detection capability due to excellent zero-flow drift. Users can apply any of-the-shelf µP for handling all other tasks like display management or wireless communication and a complete meter device. The integrated standard pulse interface enables one-to-one replacement of mechanical meters by AS6031 based single-chip heat and water meters – customer µP and software remains unchanged.

The ultra-low-current capabilities allow the use of standard AA batteries at 6-8 Hz measuring frequency even in the water meter version. The chip comes in a compact QFN48 package and allows compact designs thanks to a small number of external components.

1.1 Key Benefits & Features

The benefits and features of AS6031, SoC for Ultrasonic Flow Meters are listed below:

Figure 1:
Added Value of Using AS6031

Benefits	Features
Single-chip solution provides ready flow information System design compatible with mechanical meters High flexibility in choice for external µP handling communication and further data management	High performance + ultra-low power 32-Bit CPU
	120 * 32-bit NVRAM (non-volatile RAM) for user firmware parameter & data
	3968 * 8-bit NVRAM (non-volatile RAM) for user firmware program code
	4k * 8-bit ROM for system task code and special flow library code
Precision down to low flow rates Leakage detection	Advanced high-precision analog front-end for transducer frequencies of 50 kHz to 4 MHz at 2.5 V to 3.6 V drive voltage
	Integrated PGA, gain 2 to 17 V/V @ 1 MHz

Benefits	Features
Handles weak signals for small transducers and multiple reflections	First hit level and phase detection
	Amplitude measurement
	Up to 31 zero crossing measurements
	Precision time-to-digital converter and low-noise front-end with < 50 ps single shot with good 1MHz transducers on a DN20
	Ultra-low power consumption
Compact design Low BOM	SPI serial interface
	General Purpose I/O Unit incl. pulse interface and 2-wire master interface (I2C like)
	Supply voltage 2.5(1) to 3.6 V 3.0 to 3.6 V NVRAM store only
	Operating temperature -40(1) to 85°C
	QFN48 package (7 x 7mm²)

(1) With workaround or limitation in combinations

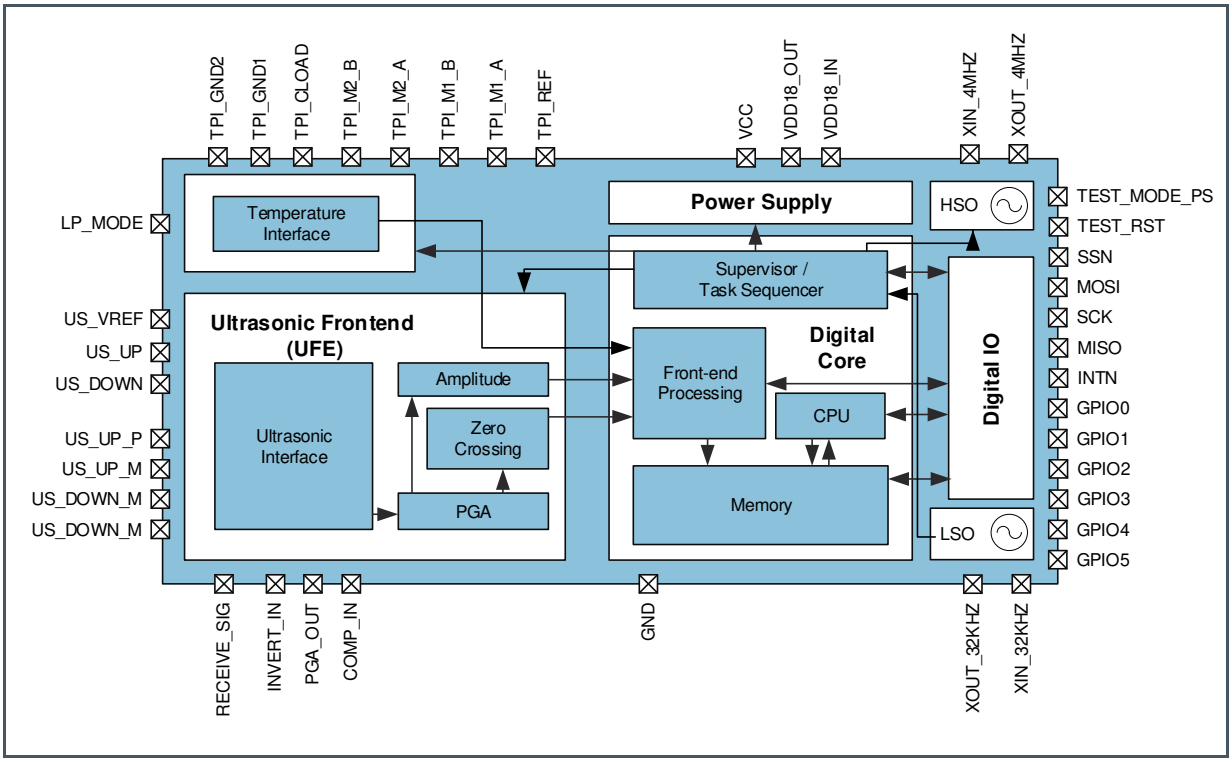
1.2 Applications

- Water meters for utilities
- Industrial water meters
- Heat meters
- Gas meters
- Volume counters

1.3 Block Diagram

The functional blocks of this device are shown below:

Figure 2 :
Functional Blocks of AS6031



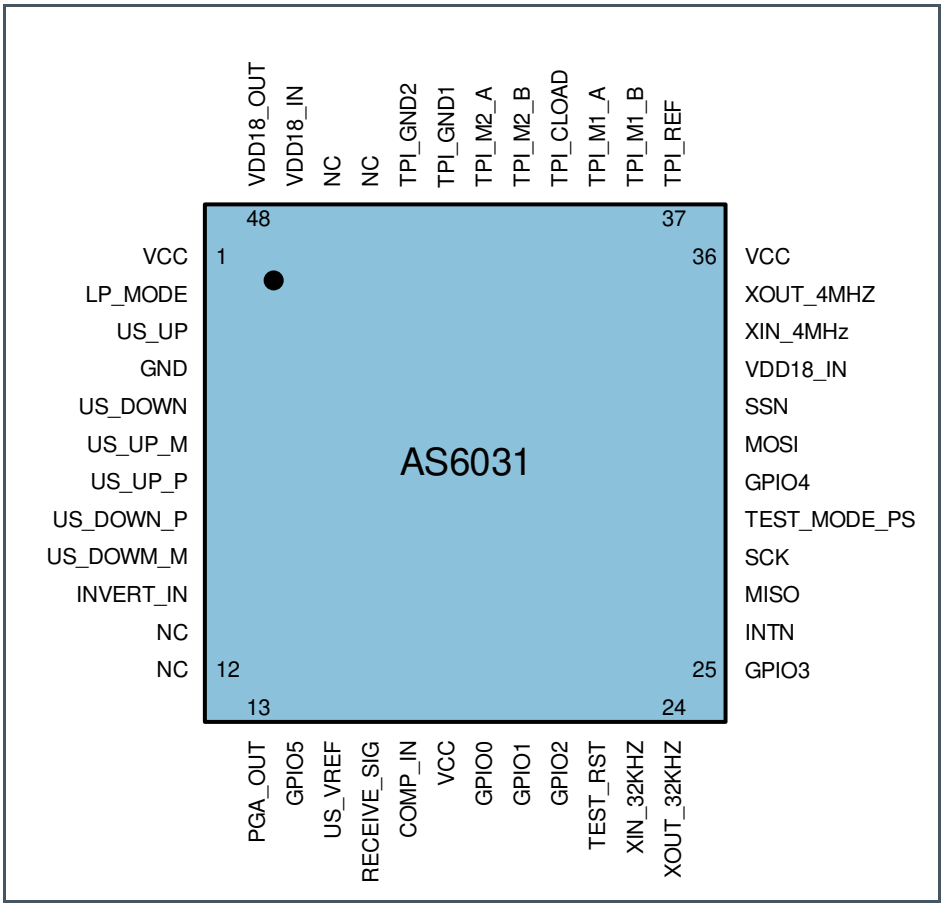
2 Ordering Information

Ordering Code	Package	Marking	Delivery Form	Delivery Quantity
AS6031-BQFM	QFN48	AS6031-BQF	Tape & Reel	500 pcs/reel

3 Pin Assignment

3.1 Pin Diagram

Figure 3:
AS6031 Pin Diagram



3.2 Pin Description

Figure 4:
Pin Description of AS6031

Pin Number	Pin Name	Pin Type(1)	Description
1	VCC	S	IO & Analog Supply
2	LP_MODE	DI	Low Power Mode, fix internal pull-up

Pin Number	Pin Name	Pin Type[1]	Description
3	US_UP	AIO	Transducer Port LV interface
4	GND	S	Ground
5	US_DOWN	AIO	Transducer Port LV interface
6	US_UP_M	AIO	Not used
7	US_UP_P	AIO	Not used
8	US_DOWN_P	AIO	Not used
9	US_DOWN_M	AIO	Not used
10	INVERT_IN	AIO	Inverting Input PGA
11	NC		
12	NC		
13	PGA_OUT	AIO	Output PGA
14	GPIO5	DIO	General Purpose 5, optional pull-up/pull-down
15	US_VREF	AIO	Ultrasonic Reference Voltage
16	RECEIVE_SIG	AIO	Receive Signal
17	COMP_IN	AIO	Comparator Input
18	VCC		IO & Analog Supply
19	GPIO0	DIO	General Purpose 0, optional pull-up/pull-down
20	GPIO1	DIO	General Purpose 1, optional pull-up/pull-down
21	GPIO2	DIO	General Purpose 2, optional pull-up/pull-down
22	TEST_RST	DI	Test Reset, must be connected to GND
23	XIN_32KHZ	AIO	Low Speed Oscillator
24	XOUT_32KHZ	AIO	Low Speed Oscillator
25	GPIO3	DIO	General purpose 3, optional pull-up/pull-down
26	INTN	DO	Interrupt, low active
27	MISO	DO	SPI: Slave Out with tristate
28	SCK	DI[2]	SPI: Serial Clock
29	TEST_MODE_PS	DI	Test Mode, fix internal pull-down
30	GPIO4	DIO	General Purpose 4, optional pull-up/pull-down
31	MOSI	DI[2]	SPI: Slave In
32	SSN	DI[2]	SPI: Slave Select, low active
33	VDD18_IN	S	VDD18 Digital Core Supply
34	XIN_4MHZ	AIO	High Speed Oscillator
35	XOUT_4MHZ	AIO	High Speed Oscillator
36	VCC	S	IO & Analog Supply
37	TPI_REF	AIO	Temperature Interface: Ref. Port

Pin Number	Pin Name	Pin Type(1)	Description
38	TPI_M1_B	AIO	Temperature Interface: Port M1 B
39	TPI_M1_A	AIO	Temperature Interface: Port M1 A
40	TPI_CLOAD	AIO	Temperature Interface: Port CLOAD
41	TPI_M2_B	AIO	Temperature Interface: Port M2 B
42	TPI_M2_A	AIO	Temperature Interface: Port M2 A
43	TPI_GND1		Temperature Interface: Ground1
44	TPI_GND2		Temperature Interface: Ground2
45	NC		
46	NC		
47	VDD18_IN	S	VDD18 Digital Core Supply
48	VDD18_OUT	S	VDD18 Voltage Regulator Output

(1) Explanation of abbreviations:

DI	Digital Input	AIO	Analog Input/Output
DO	Digital Output	S	Supply
DIO	Digital Input/Output		

(2) For standalone operation without external controller, the unconnected inputs should be configured with internal pull up by SPI_INPORT_CFG in CR_IFC_CTRL.

4 Absolute Maximum Ratings

Stresses beyond those listed under “Absolute Maximum Ratings“ may cause permanent damage to the device. These are stress ratings only. Functional operation of the device at these or any other conditions beyond those indicated under “Operating Conditions” is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Figure 5
Absolute Maximum Ratings of AS6031

Symbol	Parameter	Min	Max	Unit	Comments
Electrical Parameters					
V_{CC}	Supply Voltage to Ground	-0.3	4.0	V	
V_{oth}	All other Pins Voltage to Ground	-0.3	V_{CC} + 0.6	V	
Electrostatic Discharge					
ESD_{HBM}	Electrostatic Discharge HBM		± 1k	V	JS-001-2017
ESD_{CDM}	Electrostatic Discharge CDM		± 0.5k	V	JS-002-2018
Temperature Ranges and Storage Conditions					
T_{STRG}	Storage Temperature Range	- 55	150	°C	
T_{BODY}	Package Body Temperature		260	°C	IPC/JEDEC J-STD-020 (1)
RH_{NC}	Relative Humidity (non-condensing)	5	85	%	
MSL	Moisture Sensitivity Level		1		Maximum floor lifetime of 168h
t_{STRG_DOF}	Storage Time for DOF/Die or Wafers on Foil		3	months	Refers to indicated date of packing
T_{STRG_DOF}	Storage Temperature for DOF/Die or Wafers on Foil	17	28	°C	
RH_{OPEN_DOF}	Relative Humidity for DOF/Die or Wafers on Foil in Open Package		15	%	Opened package
RH_{UNOPEN_DOF}	Relative Humidity for DOF/Die or Wafers on Foil in Sealed Package	40	60	%	Sealed bag
t_{STRG_WP}	Storage Time for WP/Wafers or Die in Waffle Pack		6	months	17 °C – 28 °C 40 % – 60 % relative humidity storage in original Ultrapack boxes
t_{STRG_WP}	Storage Time for WP/Wafers or Die in Waffle Pack		2	years	19 °C – 25 °C <15 % relative humidity storage in closed cabinet with dry air
t_{STRG_WP}	Storage Time for WP/Wafers or Die in Waffle Pack		5	years	19 °C – 25 °C <5 % relative humidity storage in closed cabinet with dry air

Symbol	Parameter	Min	Max	Unit	Comments
t_{STRG_WP}	Storage Time for WP/Wafers or Die in Waffle Pack		10	years	19 °C – 25 °C <5 % relative humidity storage in closed cabinet and closed Ultrapak box with safeguarded Nitrogen atmosphere
Bump Temperature (soldering)					
T_{PEAK}	Peak Temperature	235	245	°C	Solder Profile
t_{WELL}	Well Time above 217 °C	30	45	s	

(1) The reflow peak soldering temperature (body temperature) is specified according to IPC/JEDEC J-STD-020 “Moisture/Reflow Sensitivity Classification for Non-hermetic Solid-State Surface Mount Devices.” The lead finish for Pb-free leaded packages is “Matte Tin” (100 % Sn)

5 Electrical Characteristics

Electrical characteristics and operating conditions indicate conditions for which the device is functional, but do not guarantee specific performance limits. Test conditions for guaranteed specification are expressly denoted.

All data given at V_{CC} = 3.0 V ± 0.3 V, ambient temperature -10 °C to 85 °C unless otherwise specified.

Notes (a) to (e) that describe the test levels are valid for the whole sections 5 and 6.

Figure 6:
Recommended operating conditions of AS6031

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Power Supply						
V_{CC}	Supply Voltage to Ground[(a)]	All operations besides NVRAM store	2.5	3.0	3.6	V
		Only for NVRAM store	3.0	3.0	3.6	V
V_{DD18}	Core Voltage to Ground[(a)]		1.71	1.8	1.89	V
T_A	Operating ambient temperature		-40		+85	°C
Oscillators						
f_{LSO}	Low speed oscillator (LSO) frequency[(e)]			32.768		kHz
f_{HSO}	High-speed oscillator (HSO) frequency[(e)]	For standard transducers, max. 2 MHz,		4		MHz
		For max. 4 MHz transducers		8		MHz
		Other frequencies in the range from 2 MHz to 8.8 MHz may be possible with limitations				
f_{SPI}	SPI Interface Clock Frequency[(d)]				8	MHz
f_{TOF}	TOF measurement frequency[(e)]	$f_{TOF} = \frac{1}{(TOF\_RATE * t_{cycle})}$	0.004	1 to 8	80[(1)]	Hz
t_{cycle}	Measurement rate cycle time[(d)]	Measurement cycle time 1 LSB = LP_MODE = 0: 1 ms LP_MODE = 1: 976.5625 µs			1024 1000	ms
f_{CPU}	CPU clock frequency[(c)]	T_a = 25 °C, V_{DD} = 1.8 V		13		MHz

(a) 100% production tested
(b) 100% production tested at 85°C wafer sort and guaranteed by design and characterization at specified temperatures.
(c) Sample tested only
(d) Parameter is guaranteed by design and characterization testing
(e) Parameter is a typical value only
(1) Maximum value with limited functionality only, e.g. no 20 ms delay between up and down measurements.

Figure 7:
DC Characteristics (V_{CC} = 3.0 V, T_j = -40 to +85 °C) of AS6031

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$I_{Standby}$	Supply current only 32 kHz, Standby mode[e]	only 32 kHz oscillator running @ 25 °C, VCC = 3.6 V = 3.0 V		 3.6 2.2		 µA µA
I_{hs}	Operation current 4 MHz oscillator[e]	VCC = 3.6 V = 3.0 V off		80 65 <1		µA µA µA
I_{tmu}	Current into time measuring unit including ultrasonic frontend[e]	Only during active TOF time measurement		1.7		mA
I_{PGA}	Current into PGA[e]	When active		0.9		mA
I_{UFE}	Average operating current ultrasonic front end only[e]	TOF_UP+DOWN, 8Hz 3V, 75µs ToF See also section 7		5.5		µA
I_O	Average operating current[e] incl. CPU processing current	TOF_UP+DOWN, 8 Hz 3V, 75µs ToF, phase detection FW see also section 7		9.2		µA
V_{oh}	High level output voltage[a]	I_{oh} = 4 mA	V_{CC} – 0.4			V
V_{ol}	Low level output voltage[a]	I_{oh} = 4 mA			0.4	V
V_{ih}	Logic high level input voltage[a]	for proper logic function for low leakage current	0.7 * V_{CC} V_{CC} - 0.2			V
V_{il}	Logic low level input voltage[a]	for proper logic function for low leakage current			0.3 * V_{CC} 0.2	V

Figure 8:
Terminal Capacitance

Symbol	Terminal	Conditions	Min	Typ	Max	Unit
C_i	Digital input[e]	measured @ VCC = 3.0 V f = 1 MHz, Ta = 25 °C		7		pF
C_o	Digital output[e]			7		
C_{io}	Bidirectional[e]			7		

Figure 9:
NVRAM

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
	Data Retention[d]	@ 85 °C, V_{CC}= 3.0[1] to 3.6 V		20		Years
	Endurance[d]	@ 25 °C, V_{CC}= 3.0[1] to 3.6 V		10^5		Cycles
		@ 85 °C, V_{CC}= 3.0[1] to 3.6 V		10^4		Cycles

(1) Lower limit of 3.0V is valid only for STORE. Can be lower during operation without store.

5.1 Ultrasonic Frontend

Figure 10:
Ultrasonic Frontend

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
	Output Voltage High[d]	Signal Offset = ½ V_{CC}		±¼ V_{CC}	±½ V_{CC}[1]	V
	Received Signal Amplitude at COMP_IN w/o PGA[d]	Signal Offset = V_{REF}	±100	±400	±V_{REF}	mV
	Received Signal Amplitude at COMP_IN with PGA[d]	Signal Offset = V_{REF}	±100	±400	±600	mV
	Accuracy, Amplitude Measurement[e]			±10 [2]		mV
V_{FHL_STEP}	Input Offset/Level Step size [e]	At COMP_IN		0.879		mV
V_{FHL_LEVEL}	Limits[d]	At COMP_IN	0		175	mV
V_{REF}	Reference Voltage[e]			0.7		V
	Transducer Interface Impedance (selectable by TI_PATH_SEL)[e]	TI_PATH_SEL = 01 TI_PATH_SEL = 10 TI_PATH_SEL = 11		550 350 214		Ω Ω Ω
f_{fire}	Typical Fire buffer frequency[d]		0,04		4	MHz
g_{PGA}	PGA gain (selectable by PGA_TRIM)[d]	@ 25 °C 1MHz @ 25 °C 2MHz @ 25 °C 4MHz	2.0 2.1 2.2		17.3 14.4 8.4	V/V
e_n	PGA Noise, referred to input[e]	10kHz to 500MHz		70		μV_{rms}
V_{OS}	PGA output offset voltage[e]	After auto-zero			1	mV

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
	Comparator input offset voltage (calibrated by Zero[e] Cross Calibration)				< 1.6	mV

(1) Without external load
(2) Only for measurements on settled maximum of amplitude plateau

5.2 Time-to-Digital Converter

Figure 11:
Time Measuring Unit (V_{CC} = 3.0 V, T_j = 25 °C)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
LSB	TDC Resolution[e]	Single-shot		22		ps
	LSB (output format)	@ 4 MHz HSO		4		
		@ 8 MHz HSO		2		
t_m	Measurement range[d]	TOF measurement	10		1000	µs
t_m	Measurement range[d]	Temperature interface measurement	10		1024	µs

5.3 Temperature Measuring Unit

Figure 12:
Temperature Measuring Unit (V_{CC} = 3.0 V, T_j = 25 °C)[(1)]

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
	Resolution RMS[e]	PT500, PT1000		17		Bit
	Gain-Drift[(2)] vs. V_{CC}	PT500, PT1000		0.01		%/V
	Gain-Drift[(2)] vs. Temperature[e]	PT500		< 4		ppm/K
		PT1000		< 2		
	Initial Zero Offset[e]	PT500		< 40		mK
	T_{ref} <–> (T_{cold}, T_{hot})	PT1000		< 20		

(1) 2-Wire measurement with compensation of Rds(on) and gain (Schmitt trigger). All values measured at Cload = 100 nF for PT1000 and 200 nF for PT500 (C0G-type)
(2) Compared to an ideal gain of 1.0

5.4 Supply Voltage Measuring Unit

Figure 13:
Supply Voltage Measuring Unit

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
	Measurement Range[d]		2.15		3.725	V
	Resolution[e]	1 LSB		25		mV
	Accuracy[d]	T_a = 25 °C T_a = whole range		± 50 ± 150		mV

6 Timing Characteristics

At V_{CC} = 3.0 V ± 0.3 V, ambient temperature -40 °C to 85 °C unless otherwise specified.

Figure 14:
Oscillator specifications

Symbol	Parameter	Min	Typ	Max	Unit
f_{LSO} (e)	32 kHz reference oscillator at frequency f_{LSO}		32.768		kHz
t_{LSO_ST} (e)	32 kHz oscillator start-up time after power-up		< 1		Sec.
f_{HSO} (e)	High-speed reference oscillator at frequency f_{HSO}		4 (8)	8	MHz
$t_{HSO\_CER\_ST}$ (e)	Oscillator start-up time with ceramic resonator		< 100		µs
$t_{HSO\_CRY\_ST}$ (e)	Oscillator start-up time with crystal oscillator (not recommended)		3		ms
f_{PS_CLK} (e)	Power supply clock. Used during power on to release cyclic measurement and for watchdog		8.7		kHz

Information

We strongly recommend using a ceramic oscillator for HSO_CLK, because a quartz oscillator needs much longer time to settle than a ceramic oscillator. This consumes a lot of current, but using a quartz oscillator has no advantage when high speed clock calibration is done as supported by AS6031.

Figure 15:
Power-on Timings

Symbol	Parameter		Min	Typ	Max	Unit
t_{VDD18_STB} (d)	Time when VDD18 is stable after power on of V_{CC} (C_L=100µF on VDD18_OUT)				20	ms
t_{RC_RLS} (d)	Time when remote communication is released after power on of V_{CC} (POR in analog and digital part finished)			37	94	ms
t_{MC_RLS} (d)	Start-up time after power-on. Time before cycle measurements are released	85 °C			1.63	s
		25 °C		1.42	2.09	
		-40 °C			3.63	

Information

During Power-on time, excess currents in the mA-range have to be considered. VCC and VDD18 have to be 0V before applying the supply voltage (e.g. in case of battery change).

6.1 SPI Interface

Figure 16:
SPI Timings

Symbol	Parameter	Min	Typ	Max	Unit
f_{SCK}	Serial clock frequency			8	MHz
t_{SCK}	Serial clock time period	125			ns
t_{pwh}	Serial clock, pulse width high	0.45 * t_{SCK}			ns
t_{pwl}	Serial clock, pulse width low	0.45 * t_{SCK}			ns
t_{sussn}	SSN enable to valid latch clock	0.5 * t_{SCK}			ns
t_{hssn}	SSN hold time after SCK falling	0.5 * t_{SCK}			ns
t_{pwssn}	SSN pulse width between two cycles	t_{SCK}			ns
t_{sud}	Data set-up time prior to SCK falling	5			ns
t_{hd}	Data hold time before SCK falling	5			ns
t_{vd}	Data valid after SCK rising			25	ns

The serial interface is SPI compatible, with clock phase bit =1 and clock polarity bit =0. It's strongly recommended to keep SSN = HIGH when SPI interface is in IDLE state.

Figure 17:
SPI Write

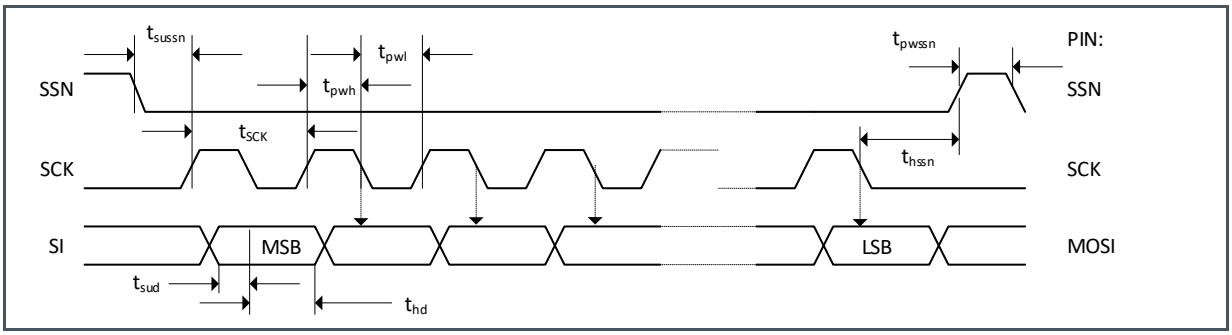
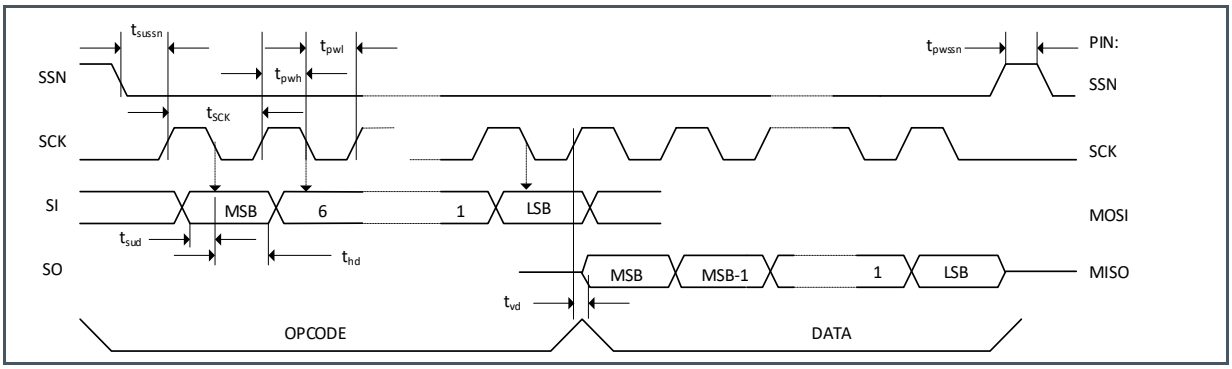


Figure 18:
SPI Read



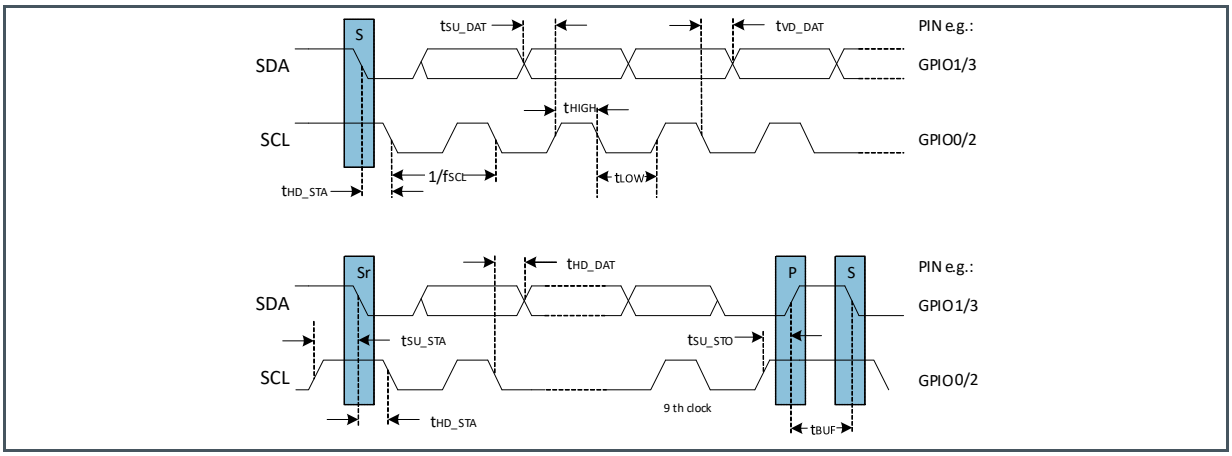
6.2 2-wire Master Interface

The integrated 2-wire master interface is very similar to I^2C, but not following the full I^2C specification. It can be used to communicate with an external EEPROM, but also any other external device like e.g. a pressure sensor.

Figure 19:
2-wire Master Timings (f_{HSO} = 4 MHz)

Symbol	Parameter	Min	Typ	Max	Unit
f_{SCL}	SCL clock frequency			400	kHz
t_{LOW}	Low period of SCL clock	1300	1500		ns
t_{HIGH}	High period of SCL clock	600	1000		ns
t_{HD_STA}	Hold time for (repeated)	600	1000		ns
t_{SU_STA}	START condition (S & Sr)	600	750		ns
t_{SU_DAT}	Setup time for repeated	100	750		ns
t_{HD_DAT}	START condition (Sr)	0	750		ns
t_{VD_DAT}	Setup time data		750	900	ns
t_{SU_STO}	Hold time data	600	1750		ns
t_{BUF}	Valid time data	1300			ns

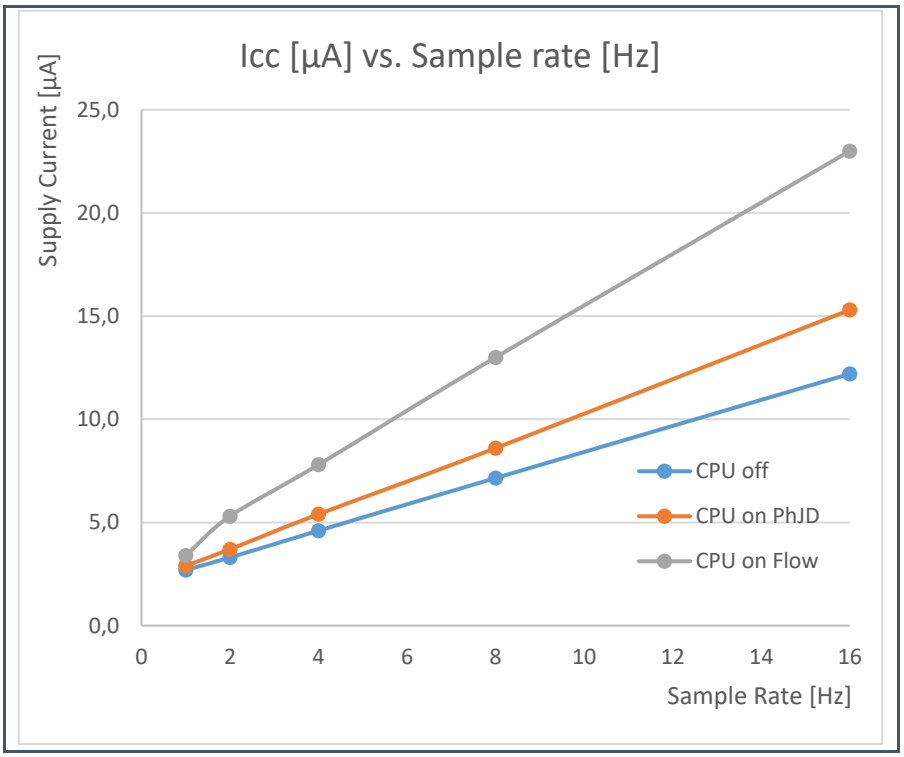
Figure 20:
2-wire Master Interface Timing



A detailed description of the 2-wire interface is given in section 10.4 2-wire Master Interface.

7 Typical Operating Characteristics

Figure 21:
Typical Current Consumption vs. Sample Rate
(VCC = 3.0V, PGA gain 2V/V, TOF 75µs, no CPU, CPU w. simple FW for phase jump detection, CPU with complex flow firmware)



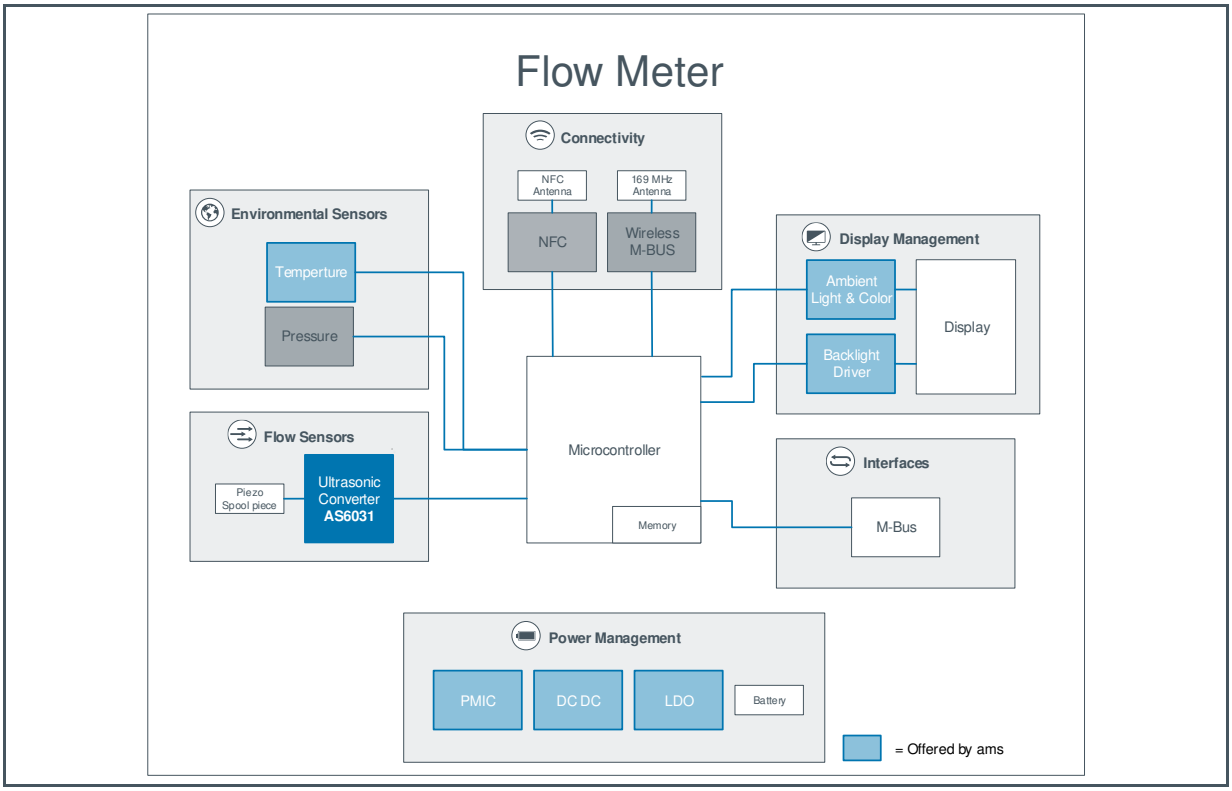
8 Functional Description

8.1 System Concept

AS6031 is a complete ultrasonic flow converter (UFC) to measure and calculate the flow in a time-of-flight based ultrasonic water or heat meter. This includes the driver for the piezoelectric transducers, the analog switches, the programmable gain amplifier and the offset stabilized comparator, the CPU to calculate the flow, the clock control unit and, above all, the measure rate control and task sequencer unit. AS6031 is an autonomous system, with the task sequencer managing the complete measurement sequence independently from an external CPU. It can be used as a pure font-end with time of flight information as an output. But by means of the internal CPU the time information can be converted into a flow calibrated information already.

A major reason to go with this concept is that the AS6031 covers the complete ultrasonic flow measurement task, but doesn't touch all the other tasks of the central microcontroller. The user therefore has high flexibility in the choice of the central microcontroller and can even go with the same used e.g. for mechanical meters. On the other hand, the user does not need to step into design of electronics for ultrasonic flow but can setup a meter in a short period of time.

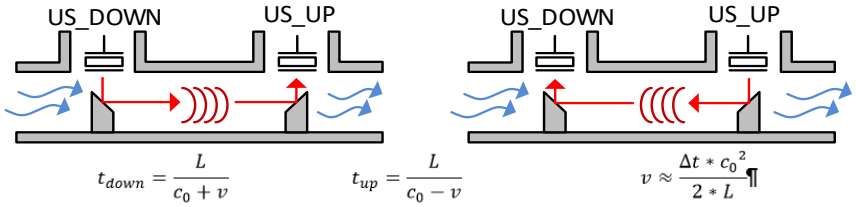
Figure 22:
Application Diagram Flow Meter



8.1.1 Measuring Principle

The **AS6031** measures flow by measuring the difference in time-of-flight (TOF) of ultrasonic pulses which travel with the flow (downstream) and opposite to the flow (upstream). For water meters, water temperature can be calculated from the time-of-flight data, too. For heat meters, a high-precision temperature measurement unit is integrated**.**

Figure 23:
Principle of Ultrasonic ToF Flow Measurement



t_{up}, t_{down}: Signal travel times TOF_{UP} and TOF_{DOWN}
$\Delta t = t_{up} - t_{down}$: $DIFTOF$
L: ultrasonic signal path length parallel to flow
c_0: speed of sound in water
v: flow speed

The flow speed v at a given cross section area is a measure for the actual flow through the spool piece and integrating the flow over time yields the flow volume.

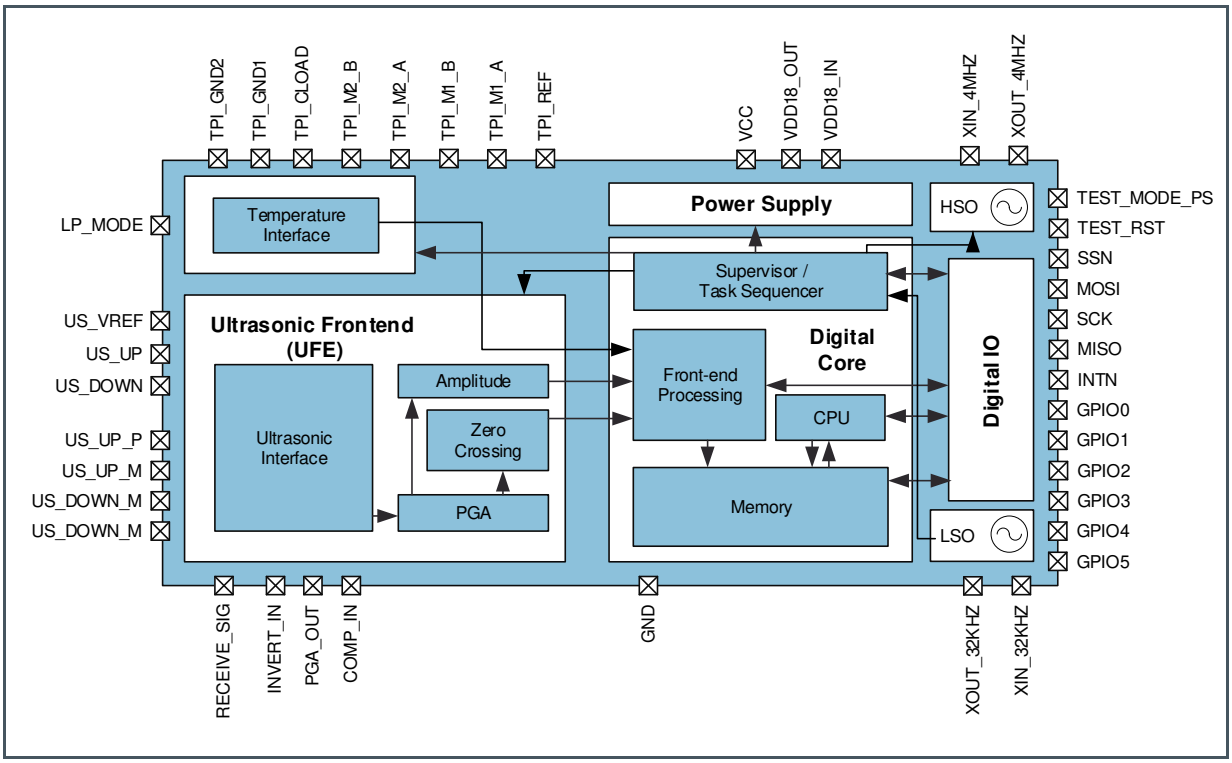
The task of AS6031 is to handle complete process, driving the transmitting piezoelectric transducer, that generate the ultrasonic burst, measuring the receiving transducer with respect to time-of-flight and amplitude, eventually doing a temperature measurement, and optionally do the complete post-processing for flow calculation.

8.2 Functional Blocks Overview

The AS6031 is made of the following major blocks:

- Ultrasonic frontend (UFE) for ultrasonic flow measurement and temperature measurement including programable gain amplifier
- Digital core with the supervisor and task sequencer, the frontend processing unit, the CPU and memory.
- Digital I/O section
- Power supply
- Oscillator drivers

Figure 24:
Major Functional Blocks of AS6031



8.2.1 Operating Modes

The AS6031 is designed for autonomous operation, with all processes, including flow calculation, being managed by the AS6031. But it may be used as pure front end, too.

- **Flow Meter Mode**: In the self-controlled flow meter mode, the supervisor triggers all measurements and the CPU does data processing to deliver processed results, independent from any external control. A bootloader, executed in the ROM of the integrated CPU, takes care of the application setup. A programmed firmware, also executed in the integrated CPU, defines the post processing. The interrupt may wake up an external microcontroller, so that it can read out the data.
- **Time Conversion Mode**: Alternatively, the AS6031 can act as a pure converter that controls the measurement, but provides pure time-of-flight data, without any data processing (time conversion mode, self-controlled). The external microcontroller takes care of the application setup and post processing.

For debugging, an external microcontroller can trigger individual tasks remotely by SPI interface commands (time conversion mode, remote controlled).

Application Setup

The application setup of AS6031 hardware is defined by two sections in the register area:

- Configuration registers (CR) (0x0C0 – 0x0CE)
- System handling registers (SHR) (0x0D0 – 0x0DD)

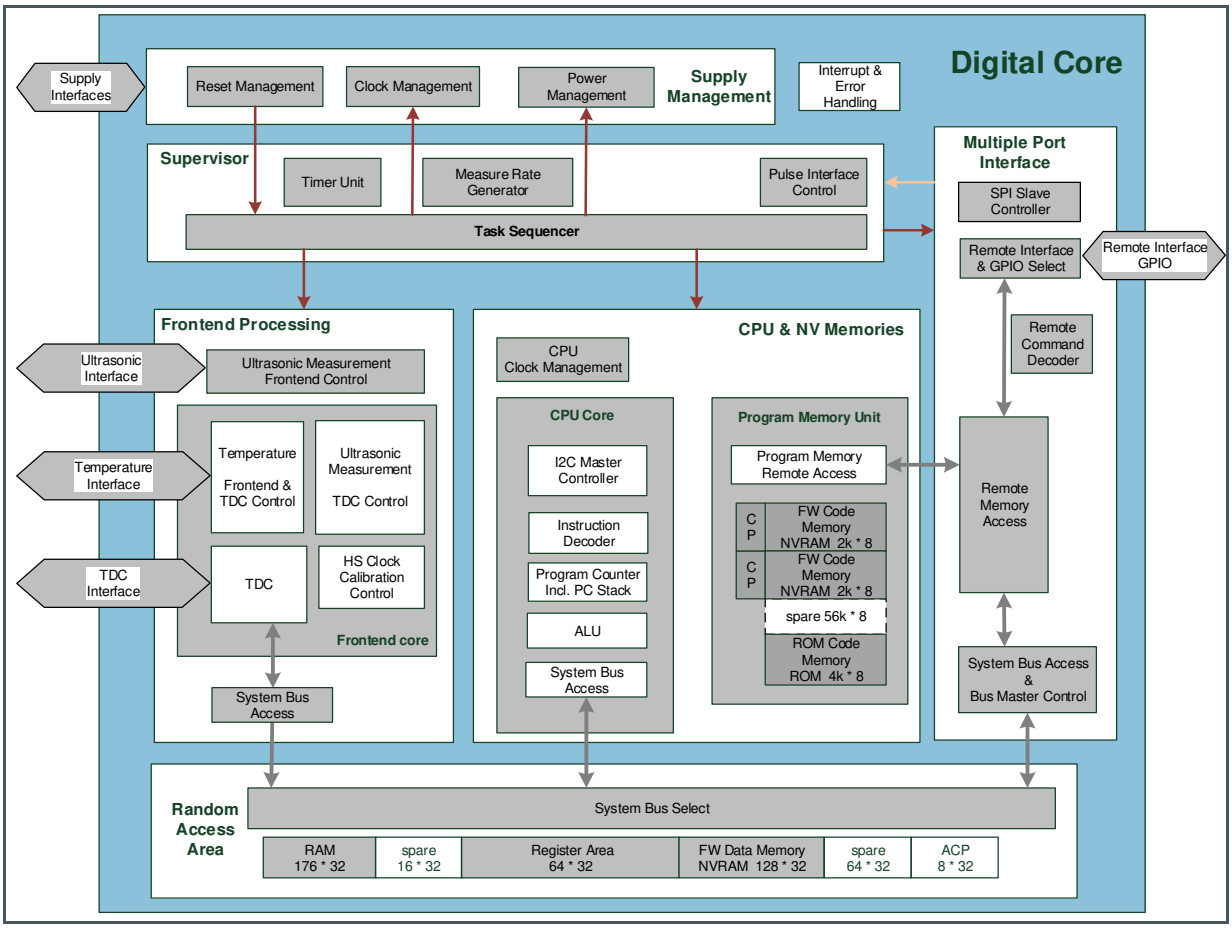
Both register sections will be reset after the execution of a “System Reset”.

8.3 Digital Core

The digital core is made of the following sub-blocks:

- Supervisor with task sequencer, timer unit, measure rate generator and pulse interface control
- Frontend processing with control of the ultrasonic and temperature frontends, the TDC and the high-speed clock calibration
- CPU and memories for optional post-processing
- The multiple port interface for SPI communication, 2-wire master interface and pulse interfaces
- Supply management with reset management, clock management and power management
- Interrupt and error handling
- Common system bus

Figure 25:
Digital Core

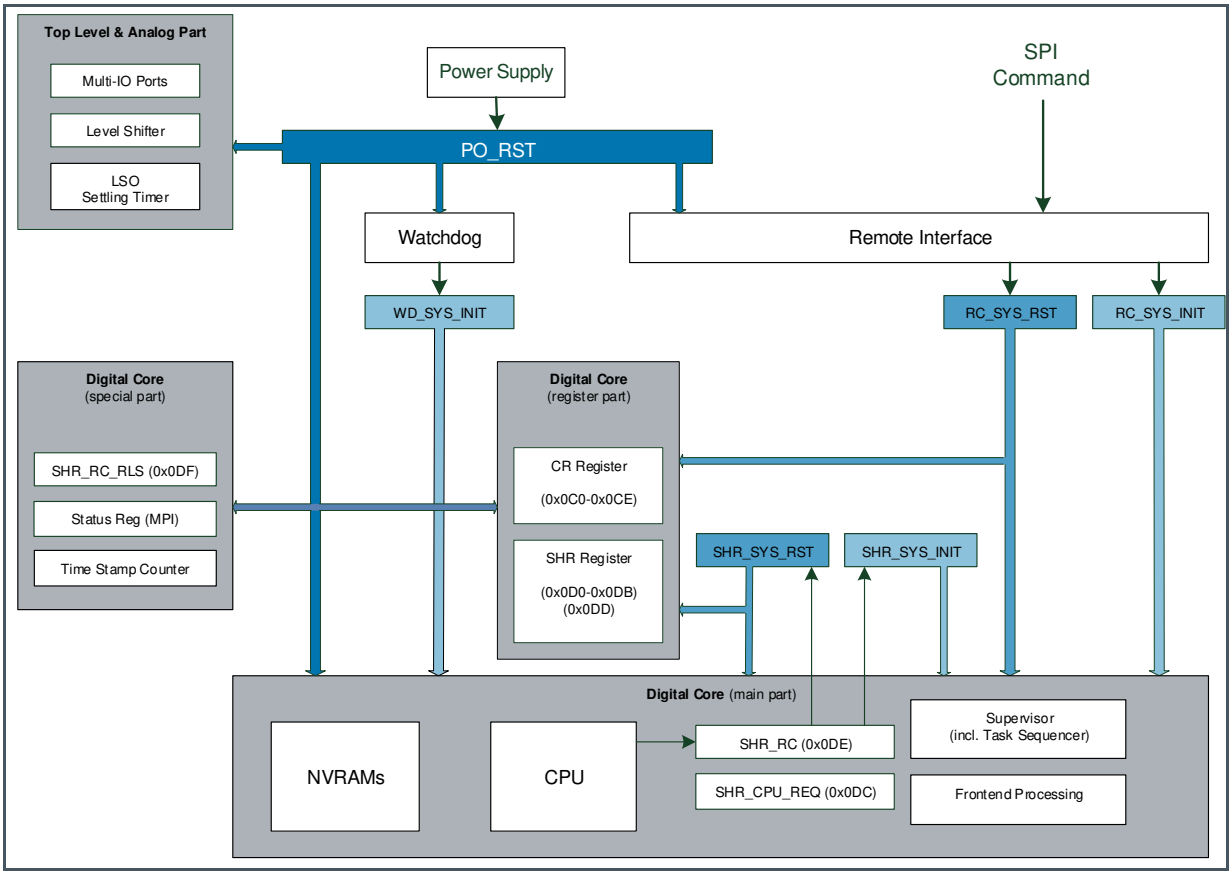


8.3.1 Reset Management

Reset Distribution

Resets in AS6031 are initiated by turning on the power supply, by the watchdog or via remote interface commands.

Figure 26:
Reset Distribution



Following resets can be distinguished:

- PO_RST
  Power-On Reset of AS6031, generated by power supply. Only performed after VDD33 is switched on. Resets complete AS6031 including digital IOs. After a PO_RESET, the measurement cycle timer is disabled for typically 2s until the settling time for the LSO has expired.

- RC_SYS_RST
  Remote Command System Reset, performed after sending a remote command 0x99. Resets the complete digital part of AS6031.
  Note: Applicable only during debugging, not for application
- SHR_SYS_RST
  System Reset performed by writing a '1' to **SHR_RC**[14] (Address 0x0DE) if appropriate release code is written before to **SHR_RC_RLS** (Address 0x0DF). Preferably initiated by CPU to allow a system reset by FW execution.
  Resets main & register part of digital core.

Resetting only the main part of the digital core, without configuration registers, is done the following way:

- WD_SYS_RST
  Watchdog System Reset, performed after the watchdog timer expired. Triggers a SYS_INIT but leaves registers and memory unchanged.
- RC_SYS_INIT
  System Init by remote command, performed after sending the SPI remote command 0x9A. Preferred remote action if register part of digital core is wanted to be untouched.

The different parts of AS6031 are reset as follows:

- Top Level and Analog Part:
  - PO_RST
- Digital Core (special part):
  - PO_RST
- Digital Core (register part):
  - PO_RST
  - RC_SYS_RST
  - SHR_SYS_RST if SHR_RC_RLS == hAF0A_4735
- Digital Core (main part):
  - PO_RST
  - RC_SYS_RST
  - SHR_SYS_RST if SHR_RC_RLS = hAF0A_4735
  - WD_SYS_INIT
  - RC_SYS_INIT

Following registers are separated from register part with different reset behavior

- SHR_RC_RLS (0x0DF): only by PO_RST
- SHR_RC (0x0DE): additionally by WD_SYS_INIT & RC_SYS_INIT
- SHR_CPU_REQ (0x0DC): additionally by WD_SYS_INIT & RC_SYS_INIT

8.3.2 Clock Management

AS6031 normally uses two external clocks and is equipped with pins for two external clock sources.

- **LSO**
  A low-speed clock (typically 32.768 kHz), connecting a quartz crystal at pins XIN_32KHZ & XOUT_32KHZ. This clock is the basis for the supervisor, including measure rate generator and task sequencer. It is running all the time when using normal low power mode.
- **HSO**
  A high-speed clock (typically 4 or 8 MHz), connecting ceramic resonator to pins XIN_4MHZ & XOUT_4MHZ. It is used for the frontend processing and is activated only when needed. Compared to a quartz, a ceramic resonator has the benefit of a short settling time which saves power consumption. On the other hand, the clock needs to be calibrated periodically versus the LSO quartz.
- Alternatively, active external clock can be fed into the XOUT pins (XIN pins need to be grounded then).

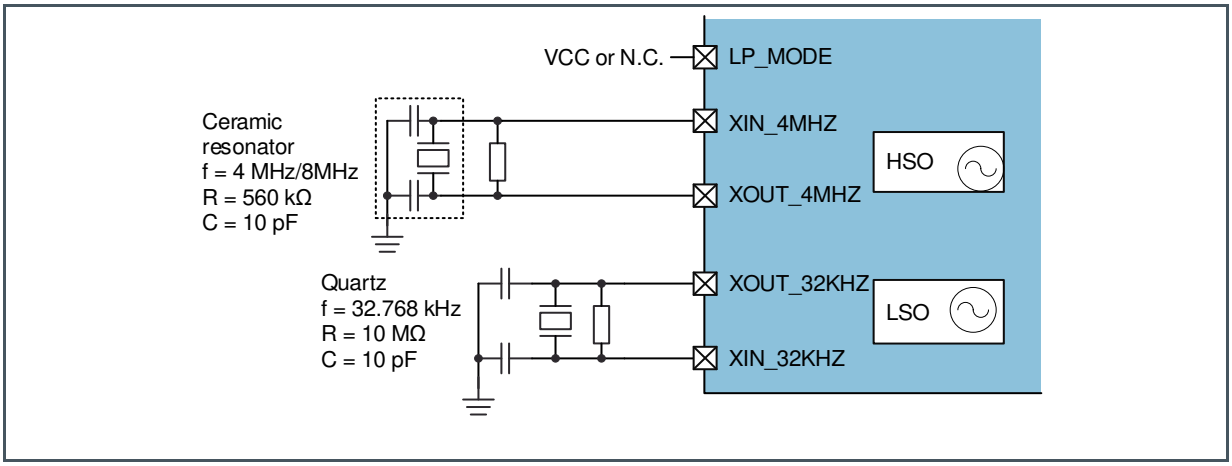
Note: In addition, there is an internal low speed oscillator PS_CLK of typ. 8.7 kHz which is used for the power-up timing to release measurements and for the watchdog.

We distinguish the following clock operation modes:

Low Power Mode

AS6031 is sourced by the external low-speed oscillator (LSO). This is the standard in typical applications.

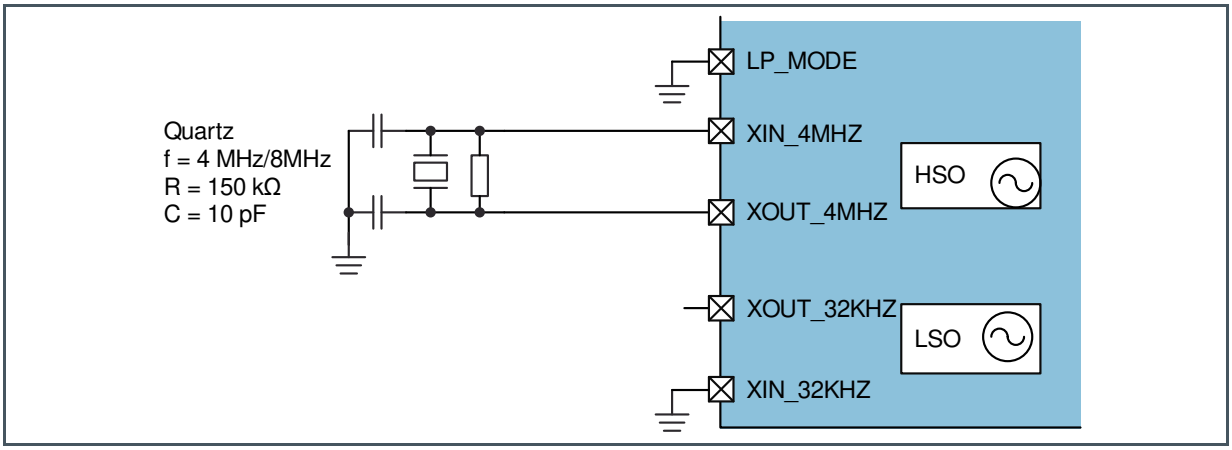
Figure 27:
Connecting Oscillators in Low Power Mode



Single-source Clocking Mode

No external low-speed source is needed. The internal low-speed clock is derived from high-speed clock, which in this case needs to be a quartz. The internal LSO frequency is then 32 kHz. The high-speed clock needs to run all the time. As the HSO needs about 80 µA, this mode is not recommended for applications where low power is needed.

Figure 28:
Connecting Oscillator in Single-source Clocking Mode



Clock Divider Options

Following table shows recommended settings for the clock dividers.

Figure 29:
Useful Caption

HS_CLK	4 MHz	8 MHz
Fire Burst Frequency	40 kHz - 1 MHz	62.5 kHz - 4 MHz
HSC_DIV_MODE	0	1
HSC_DIV	0	1
TDC Result Resolution (* $1/2^{16}$)	250 ns	125 ns

The recommended HS clock frequency and derived clock divider settings are mainly defined by required fire burst frequency.

Clock Calibration

The HSO frequency will vary from device to device but also with temperature. Therefore, it needs to be calibrated frequently. During calibration, four periods of the LSO (122.0703125 µs) are measured by means of the HSO.

The register SRR_HCC_VAL (High-Speed Clock Calibration Value) is updated. The value is eight times the real frequency at the TDC: 8 * f_{HSO} / Hz.

Nominal value with 250ns period: 32,000,000 (0x1E84800)
Nominal value with 125ns period: 64,000,000 (0x3D09000)

Example with real value from SRR_HCC_VAL, e.g. 249.9579ns: 32,005,389 (0x1E85D0D)

Correction factor: 32,000,000/32,005,389 = 0.999832, to be used for all further timing calculations in any post processing.

The status flag **HCC_UPD** in SRR_FEP_STF (Frontend Processing Status Flags) indicates whether the content is updated or not.

Relevant Registers

The table below lists most important registers and parameters for setting the clock management.

Figure 30:
Clock Control & Status Registers

Register	Parameter	Description
CR_CPM (Clock- & Power-Management)	**HSC_CLK_ST**	High-speed clock settling time (77µs, 107µ, …, 5ms) 135µs is a good value for ceramic oscillators.
	HSC_RATE	Sets the calibration rate of HSO versus LSO (off, every 2nd , …,every 100th).
	HSC_DIV	High-Speed Clock Divider 0: HSC_CLK not divided 1: HSC_CLK divided by 2 Optionally with HSC_DIV_MODE = 1: Only applied to low speed clocking & frontend control
	HSC_DIV_MODE	High speed clock divider mode 0: all HSC clock dividers controlled commonly by HSC_DIV 1: HSC clock dividers individually configurable (bit 8:5)
SHR_EXC (Executables)	**HSO_CLR**	Clears the high-speed oscillator (typically used by firmware code for I2C handling)
	HSO_REQ	Requests the high-speed oscillator (typically used by firmware code for I2C handling)
SHR_RC (Remote Control)	**HSO_MODE**	High Speed Oscillator Mode (for debugging only) 00: No change of HSO_MODE state (WO) 01: HSO controlled as configured 10: HSO always on 11: No change of HSO_MODE state (WO)
	HSC_DIV_STATE	Info only, set by HSC_DIV in CR_CPM 00, 11; not possible 01: HSC_DIV = 0 10: HSC_DIV = 1
SRR_FEP_STF (Frontend Processing Status Flags)	**HCC_UPD**	Indicates whether the clock calibration value is updated or not.

Register	Parameter	Description
SRR_HCC_VAL (High-Speed Clock Calibration Value)	**HCC_VAL**	High-speed clock calibration value. = $122.0703125/T_{HSO}*2^{16}$
SRR_MSC_STF (Miscellaneous Status Flags)	**HSO_STABLE**	Flag for the end of the high-speed oscillator settling time, indicating that high-speed oscillator is settled and stable. For CPU handling the flag CPU_SFLAG_HSO_ST_TO is preferred.
SHR_CPU_REQ	**CPU_SFLAG_HSO_ST_TO**	0: High speed oscillator not in timeout condition 1: High speed oscillator in timeout condition Cleared by HSO_CLR in SHR_EXC Same as HSO_STABLE but updated and valid only while CPU is running (synchronized by CPU clock)

8.3.3 Measure Rate Generator

The measure rate generator supplies up to 8 different measure task requests, which can trigger the task sequencer. The task sequencer is a state machine and then manages the processing of the measurement tasks, based on the measure cycle timer and the measure rates. The measure rate cycle time (MR_CT) is the central clock in the measure rate generator.

Figure 31:
Cycle Times for Tasks (example)

Task	Cycle Time(1)	Typical setting		Comment
MR_CT			125	8 Hz base frequency
TOF	TOF_RATE × MR_CT × T_{MCT}	**TOF_RATE**	1	8 Hz flow measurement
AM	AM_RATE × TOF_RATE × MR_CT × T_{MCT}	**AM_RATE**	1	Amplitude measurement performed with every time of flight measurement
AMC	AMC_RATE × AM_RATE × TOF_RATE × MR_CT × T_{MCT}	**AMC_RATE**	50	Amplitude measurement calibration rate
VM	VM_RATE × MR_CT × T_{MCT}	**VM_RATE**	100	Voltage measurement every 12.5s
TM	TM_RATE × MR_CT × T_{MCT}	**TM_RATE**	240	Temperature measurement every 30s
HSC	HSC_RATE × MR_CT × T_{MCT}	**HSC_RATE**	100	High-speed clock calibration every 12.5s
ZCC	ZCC_RATE × MR_CT × T_{MCT}	**ZCC_RATE**	100	Zero-cross calibration every 12.5s

(1) LP_MODE = 1: T_{MCT} = 976.5625 µs

Relevant Registers

The following table lists the most important registers and parameters for setting the measure rates.

Figure 32:
Measure Rate Settings

Register	Parameter	Description
CR_CPM (Clock- & Power-Management)	**HSC_RATE**	High-speed clock calibration rate
CR_TPM (Temperature Measurement)	**TM_RATE**	Defines the number of sequence cycle triggers between sensor temperature measurements [0=off, 1 to 1023].
CR_USM_PRC (Ultrasonic Measurement Processing)	**ZCC_RATE**	Zero-cross calibration rate [0=off, 1, 2, 5, 10, 20, 50, 100].
CR_USM_TOF (Ultrasonic Measurement Time of Flight)	**TOF_RATE_INIT**	Initial value of TOF rate after autoconfiguration of bootload
CR_USM_AM (Ultrasonic Amplitude Measurement)	**AM_RATE**	Amplitude measurement rate [0=off, 1, 2, 5, 10, 20, 50, 100].
	AMC_RATE	Amplitude measurement calibration rate [0=off, 1, 2, 5, 10, 20, 50, 100].
SHR_TOF_RATE (Time Of Flight Rate)	**TOF_RATE**	Rate of flow measurements [0=off, 1 to 63]. In multiples of 976.5625µs/1ms.

8.3.4 Task Sequencer

The task sequencer triggers the various measurement tasks and calibration tasks. It also triggers the post processing. The task sequencer has two operation modes:

- 1-phase mode: All measurements follow the same trigger. Post-processing can be triggered by the flow measurement and the other measurements.
- 2-phase mode: Flow measurement and temperature measurement are triggered by separate triggers, called A and B.

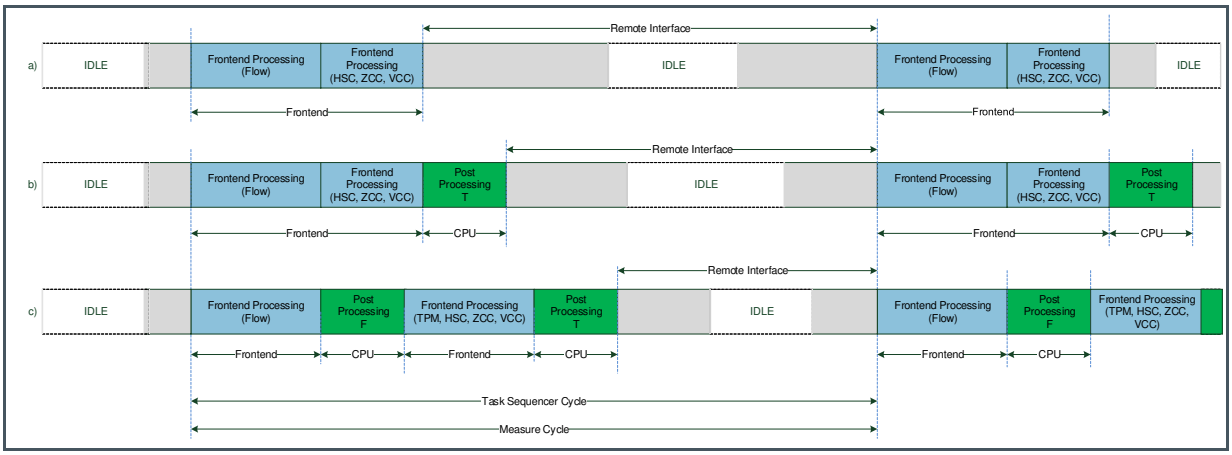
The 1-phase mode is the preferred one when working with CPU post processing or when no temperature measurement is done. The idle time for communication with an external partner is maximized.

The 2-phase mode is needed in time conversion mode (no CPU post processing) with temperature measurement active. The reason is that for flow and temperature the same frontend data buffer is used.

1-Phase Mode

All measurements tasks are triggered by the same single trigger. Following applications are covered:

Figure 33:
1-phase Mode



- a) Time conversion mode without temperature measurement (TPM)
  TS_MCM = 0 / TS_PP_T_EN = 0 / TS_PP_F_EN = 0
- b) Flow meter mode without temperature measurement
  TS_MCM = 0 / TS_PP_T_EN = 1 / TS_PP_F_EN = 0
- c) Flow meter mode with temperature measurement
  TS_MCM = 0 / TS_PP_T_EN = 1 / TS_PP_F_EN = 1

In this mode all measure tasks can be performed in one task sequencer cycle. Doing this, the IDLE time which can be used for remote communication is as large as possible.

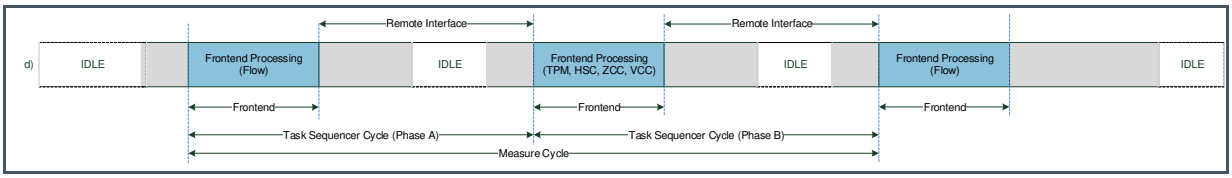
2-phase Mode

This mode is needed for

- D) Time Conversion Mode with Temperature Measurement (TPM)
  TS_MCM = 1 / TS_PP_T_EN = 0 / TS_PP_F_EN = 0

In this mode the measure cycle is divided into two task sequencer cycles. So, it is possible to read out flow and temperature results subsequently via the serial interface. The idle time is split and reduced.

Figure 34:
2-phase Mode



Relevant Registers

The following table lists the most important registers and parameters for setting the measure rates.

Figure 35:
Useful Caption

Register	Parameter	Description
CR_MRG_TS (Measure Rate Generator & Task Sequencer)	**MR_CT**	Task sequencer cycle time. The actual physical measure rate cycle time is t_{cycle} = MR_CT* 976.5625 µs [0, 1…8191]. The measurement rate generator triggers measurements in two alternating channels, one (A) triggering the flow and amplitude measurement, the other one (B) triggering temperature and voltage measurement as well as the high-speed clock (HSO) and the comparator offset calibration. Optionally (TS_MCM = 1), channel B triggers a half cycle time after channel A, to avoid mutual influences among the measurements.
	TS_MCM	Task sequencer measure cycle mode: 0: Trigger A (flow) and Trigger B (temperature) follow sequentially 1: Trigger A (flow) and Trigger B (temperature) are separated by $t_{cycle}/2$
	TS_PP_MODE	Post processing mode (only if post processing is enabled) 0: Post processing requested with every task sequencer trigger 1: Post processing only requested if a measurement task is requested 1 is the recommended setting
	TS_PP_F_EN	Enables Post Processing F (after USM flow and amplitude measurement task) 0: Post Processing F disabled 1: Post Processing F enabled 0 is the recommended setting
	TS_PP_T_EN	Enables Post Processing T (final one after temperature, voltage and calibration measurement tasks or post processing F) 0: Post Processing T disabled 1: Post Processing T enabled

Task Processing Times

The following examples shows how task sequencer cycles are allocated by the different measure and communication tasks. The data show typical times for typical configurations only and likely vary from configuration to configuration.

Figure 36:
Task Duration

Task		Comment	Time[ms]
Refreshing the voltage regulator		Always performed at start of cycle	2.6
Frontend processing	Max. time for ultrasonic measurement with a pause time of 20 ms	For a differentiated calculation please refer to corresponding chapter	21
CPU post processing		Maximum time for a firmware execution with 4000 cycles and recommended CPU speed	0.5
Refreshing the voltage regulator		Always performed at start of cycle	1.7
	Voltage measurement	Maximum time if voltage measurement is performed in this cycle	2

Task		Comment	Time[ms]
Frontend processing Typically not with every measurement	Temperature measurement	Typical time for an internal + 2-ports/2-wire measurement with a pause time of 10 ms	16
	High-speed clock calibration	Maximum time if high-speed clock calibration is performed in this cycle	0.5
	Zero-cross calibration	Maximum time if zero-cross calibration is performed in this cycle	0.5
CPU post processing		Maximum time for a firmware execution with 4000 cycles and recommended CPU speed	0.5
SPI Remote communication		Maximum time for an SPI communication of 100 bytes payload at 8 MHz and an inter-byte gap of 1µs	0.2
		Total	45.5
NVRAM related tasks:			
NVRAM Check		Recommended with CPU processing.	8.6
NVRAM Recall		Timer triggered in larger intervals	0.4

8.3.5 Initial Start / Restart

The initial start sequence is performed after a power-on reset. The restart is performed after sending the remote command RC_SYS_RST or setting SHR_SYS_RST. Figure 38 shows the sequences. Cyclic measurements are released earliest after the start-up timer, based on the internal power supply clock (PS_CLK, typ. 8.7 kHz) has reached its timeout.

Communication is possible as soon as the POR of the analog and digital par have been finished.

Measurement Start in Flow Meter Mode

A measurement start in flow meter mode requires that an executable firmware (FW code & FW data) is programmed to device with enabled “Autoconfig Release Code”. Then the measurement starts automatically without any interaction via remote interface as soon all steps are performed as described in flow diagram above. In case that firmware data enables interrupt for “Bootloader Finished”, this interrupt should be served by remote controller to release interrupt handling for all other kind of interrupt requests which follow.

Measurement Start in Time Conversion Mode

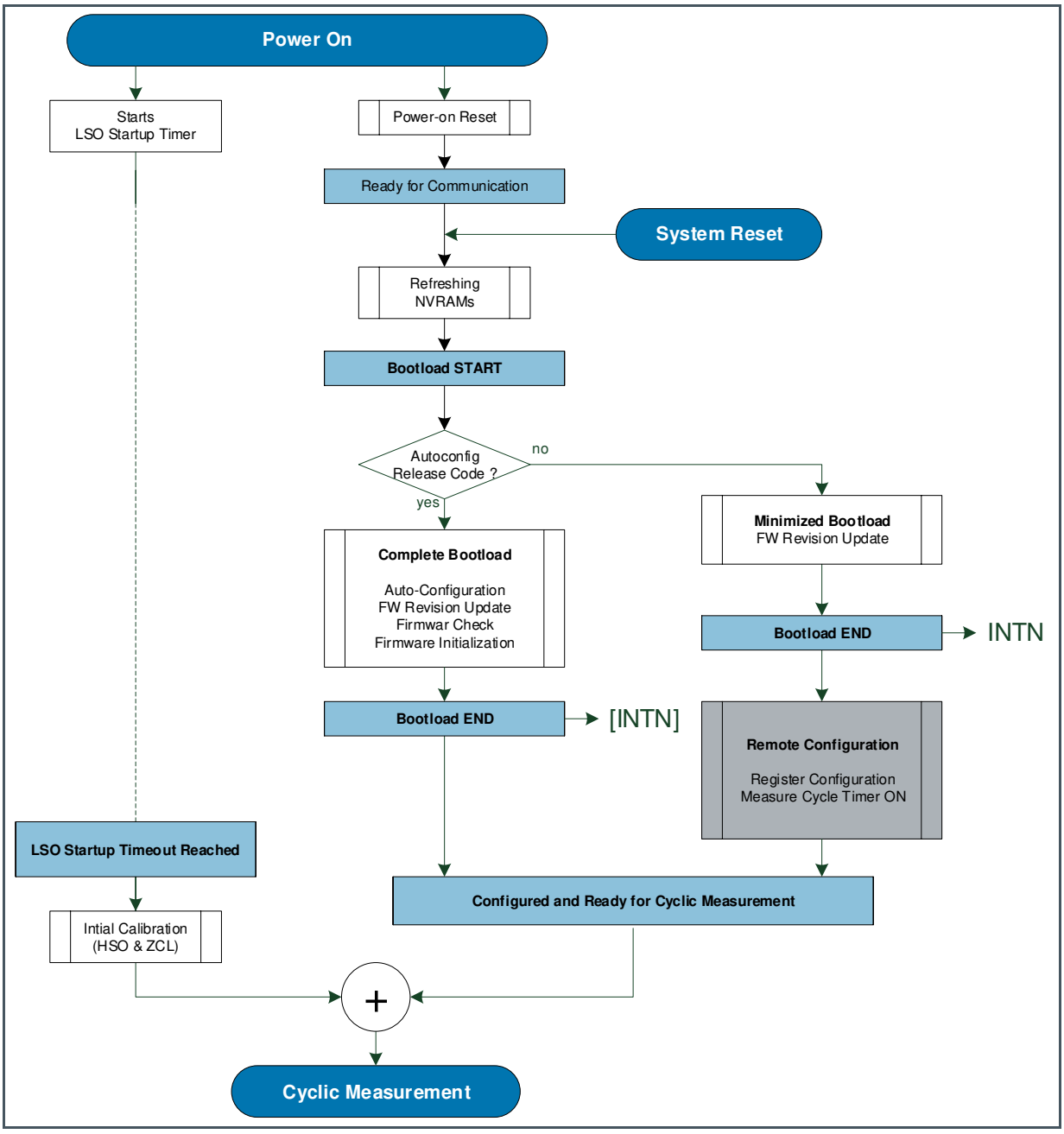
A measurement start in time conversion mode typically requires that “Autoconfig Release Code” is disabled. Further interactions via remote interface have to be performed as follows:

1. Wait on interrupt INTN
2. Check interrupt flag on reading SRR_IRQ_FLAG, bit 2, BLD_FNS
   opcode0x7A 0xE0, read data (Bit 2 of 32)
3. Clear interrupt flag register by sending RC_IF_CLR, opcode 0x8D
4. Write Configuration Data to CR addresses 0x0C0 to 0x0CB and SHR addresses 0x0D0 to 0x0D2 / 0x0DA to 0x0DB
   opcode 0x5A 0xC0, 0xXXXXXXXX … 0x5A 0xD0, 0xXXXXXXXX …
5. Set Measure Cycle Timer On RC_MCT_ON, opcode 0x8B
6. Check if Cycle Timer is on with RC_READ_STATUS 0x8F, bit 4, MCT_STATE

Figure 37:
Start / Restart Timings

Symbol	Parameter	Min	Typ	Max	Unit
t_{POR_RST}	Power-on reset, based on PS_CLK		37	94	ms
t_{NVRAM_RF}	Refreshing NVRAM		0.36	0.36	ms
$t_{bottload\_comp}$	Complete bootload time, FW Init not considered		11.84	19.66	ms
$t_{bottload\_min}$	Minimized bootload time		0.11	0.14	ms
t_{conf_remote}	Remote configuration time, based on SPI:CLK. @ f_{SPI} = 1 MHz, byte gap = 2 µs		0.91		ms
$t_{initial\_calib}$	Initial calibration time (Typical configuration)		4.82	4.88	ms

Figure 38:
Start / Restart Sequence



8.4 Ultrasonic Frontend (UFE)

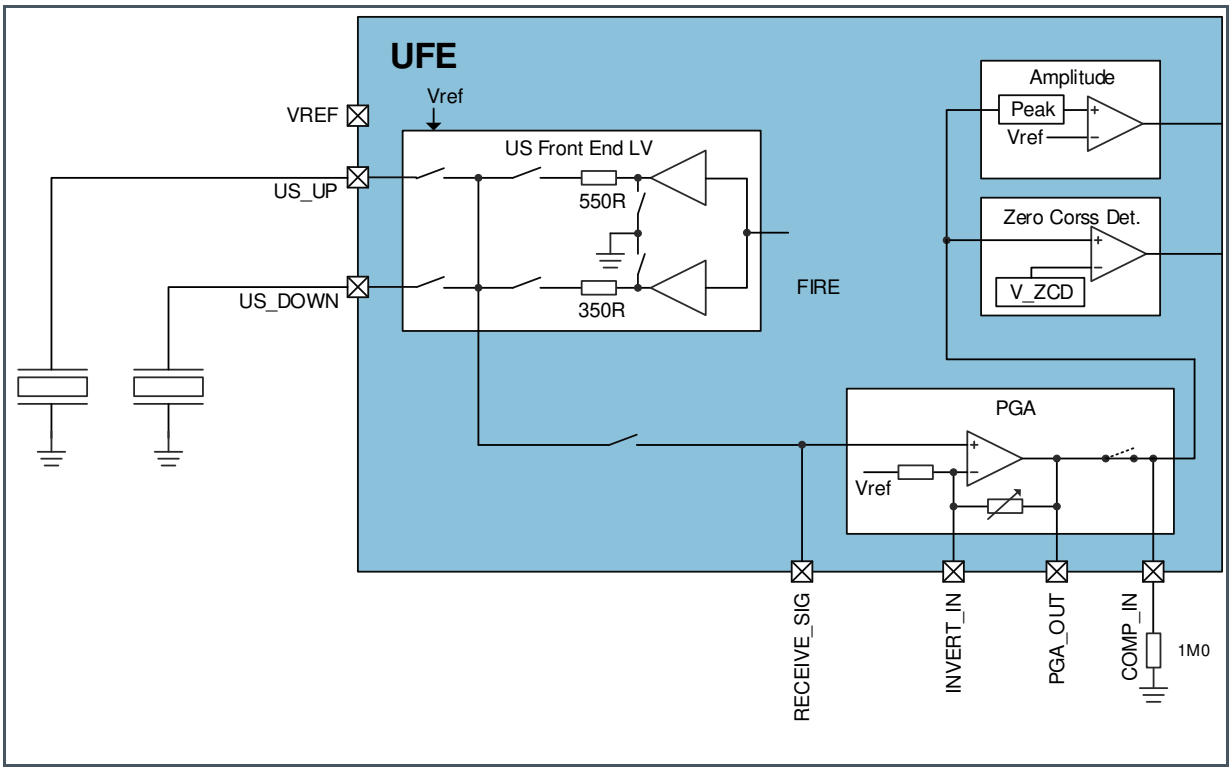
The ultrasonic frontend is made from the following sub-blocks:

- Fire buffers and switches to drive transducers in voltage mode
- Programmable amplifier in the receive path

- Zero-crossing detection circuit
- Amplitude measurement circuit
- Switching network to measure one or two external temperature sensors in 2-wire or 4-wire mode

The AS6031 is designed for autonomous operation. The individual measurement tasks are triggered by the Measure Rate Generator. The individual tasks (like flow, amplitude, temperature) are controlled themselves by individual sequencer units.

Figure 39:
Ultrasonic Frontend Block



The transducers are driven by the voltage applied at VCC. The ultrasonic transducers are directly connected to pins US_UP (against the flow) and US_DOWN (with the flow). The resistors in the transducer driver path are integrated in AS6031.

There are two fire buffers, one with 350 Ohm and one with 550 Ohm. Having both in parallel the resistance is 214 Ohm. Ideally, the impedance is in the same order as the impedance of the transducer to get maximum acoustic power out of the transducers.

8.4.1 PGA (Programmable Gain Amplifier)

AS6031 has an integrated amplifier with programmable gain in the receive path. This allows to amplify the receive signal with a gain depending on the frequency.

Figure 40:
Gains for typical water meter frequencies, simulated @ typical corner: 25 °C and 2.5V VCC

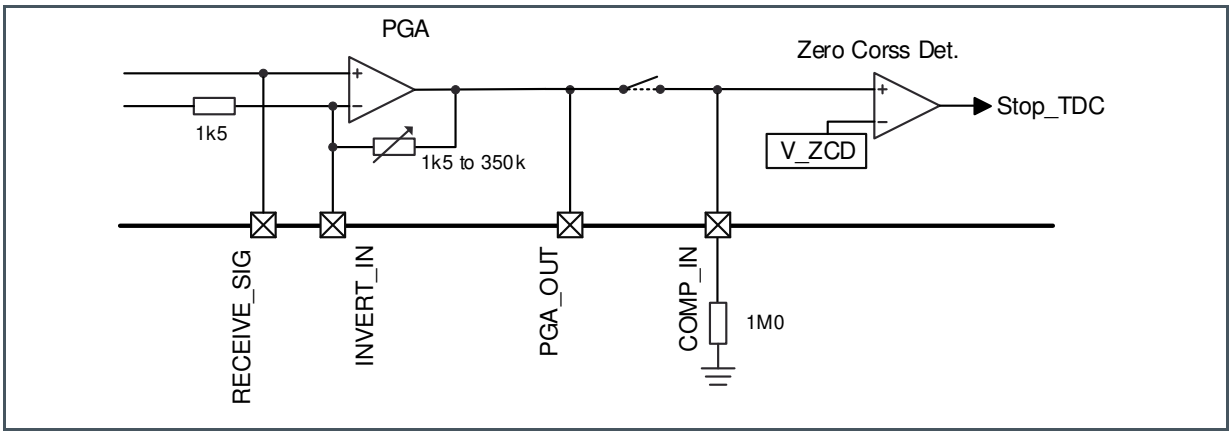
PGA Gain Setting	Gain @ 0 Hz (DC)	Gain @ 1 MHz	Gain @ 2 MHz	Gain @ 4 MHz
0	2	2	2,1	2,2
1	3	3	3	3,3
2	4	4	4,1	4,4
3	5	5	5,1	5,3
4	7	7	7,1	6,7
5	10	9,8	9,5	7,7
6	14	13,3	12,2	8,2
7	19	17,2	14,3	8,3

The gain is set by **PGA_TRIM.**
The PGA is turned on only during the ultrasonic measurement and for the calibration of the zero-cross detection.
These settings are done in register CR_USM_AM (Ultrasonic Amplitude Measurement).

It's recommended to set the amplification in a way that the amplified signal amplitude is between ±200 and ±400 mV.

Note: PGA current consumption can increase with high signal levels, getting close to the rails.

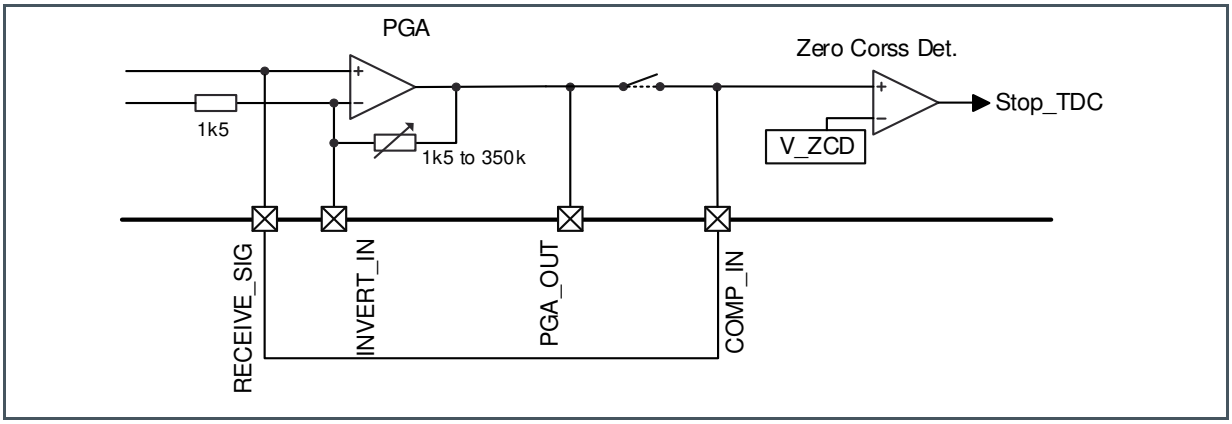
Figure 41:
Wiring with PGA enabled



PGA Disabled

In case the receive signal is strong enough, the PGA may be disabled to save current, setting **PGA_MODE** = 0 and bit 29 in register 0x0CE = 1 (to make RECEIVE_SIG accessible). With the PGA being disabled, it is necessary to connect the RECEIVE_SIG and COMP_IN pins.

Figure 42:
Wiring with PGA disabled



Relevant Registers

Figure 43:
PGA Registers

Register	Parameter	Description
CR_USM_AM (Ultrasonic Amplitude Measurement)	**PGA_TRIM** **PGA_MODE**	0 to 7: 2V/V to 19V/V PGA dis-/enabled

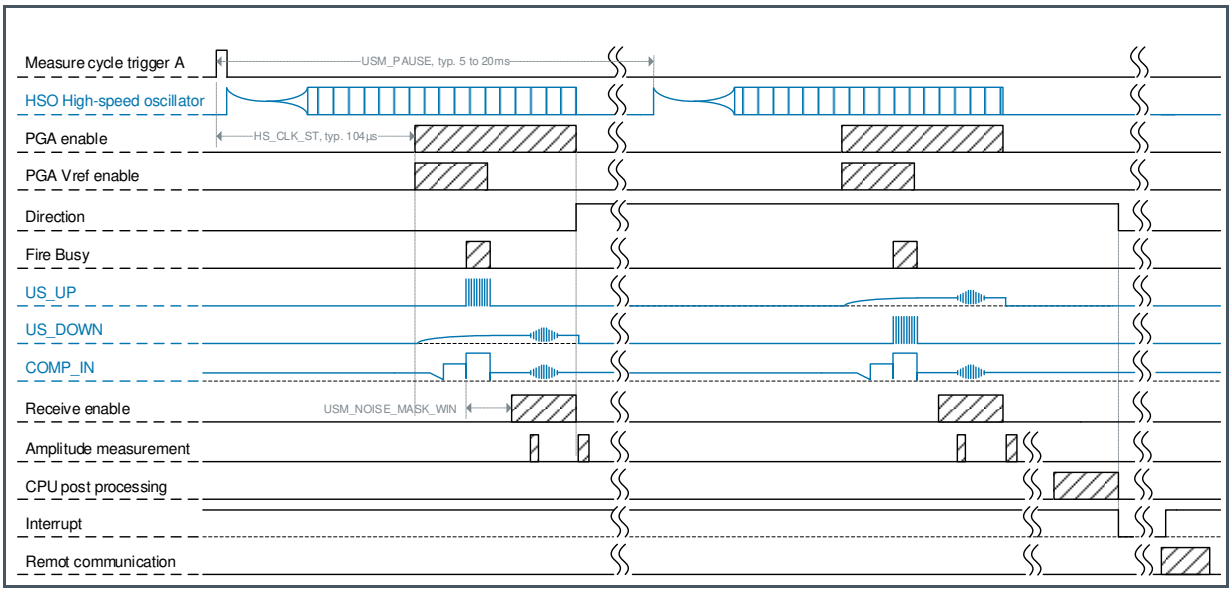
8.5 Ultrasonic Flow Measurement

8.5.1 Flow Measurement Sequence

The AS6031 manages the complete process of flow measurement, with special control of power consumption. All relevant elements, like high-speed oscillator, PGA, comparator and TDC as well as CPU, are active only for the period they are needed. This brings down the typical power consumption to the lowest possible level.

The following graph shows a complete flow measurement task with its sub-processes.

Figure 44:
Timing Diagram Flow Measurement



The ultrasonic flow sequence is made of the following steps:

Figure 45:
Flow Measurement Sequence & Parameters

Step	Register	Parameter
0	General settings:	
	CR_CPM (Clock- & Power-Management)	**BF_SEL** sets the base frequency for the time interval between up and down measurement, as used by USM_PAUSE. According to the power net frequency, it is set to 0 := 50 Hz or 1 := 60Hz. Having a period in multiples of this frequency helps to suppress ripple voltage. **TI_PATH_SEL** sets transducer fire buffer impedance: 01:= 550 Ohm; 10:= 350 Ohm; 11:= 214 Ohm
	CR_MRG_TS (Measure Rate Generator & Task Sequencer)	**MR_CT** defines the measure rate cycle time. In low-power mode, this is in multiples of 976.5625 µs, as derived from the 32768Hz LSO. 0 := disabled 1 to 8191 := Cycle time = MR_CT x 976.5625 µs (LP_MODE = 1) The cycle time needs to be as short as the period of the most frequent task. In water this would be e.g. 125ms for an 8 Hz flow measurement. **TS_PP_F_EN** turns on post-processing directly after the US flow measurement (flow meter mode). With TS_MCM = 1, set TS_PP_F_EN = 0 and use TS_PP_T_EN only, With TS_MCM = 0, use TS_PP_F_EN = 1 to evaluate flow results before they are overwritten by an (optional) subsequent sensor measurement **TS_PP_T_EN** turns on post-processing as last state of the task sequencer.
	CR_USM_PRC (Ultrasonic Measurement Processing)	**USM_DIR_MODE** defines which fire buffer fires first in a flow measurement sequence. 00 := Always starting firing via UP-buffer (against the flow), 01 := Always starting firing via DOWN-buffer (with the flow) 1x := Toggling sequence with every ultrasonic measurement. This option is best to compensate for temperature drift within a measurement. A temperature drift and therefore a speed of sound change between up and down measurements will give an error. If the following measurement the error has the opposite sign when the sequence has opposite order. So, in average, the error is compensated.
1	Turn-on HSO and wait until the HSO has settled (start-up time)	

Step	Register	Parameter
	CR_CPM (Clock- & Power-Management)	**HSC_CLK_ST** defines the turn-on or settling time for the high-speed oscillator. To save current, the HSO turns on only for the TOF measurement. As the resonator needs time to establish the oscillation, there is a delay before the fire buffer send. The delay can be set in steps of 77µs, 104µ, 135 µs, 196 µs, 257 µs, 379 µs, 502 µs, ~5000 µs. b010 := 135 µs is a good setting for ceramic resonators. Bigger delays will increase current. 5ms would be needed only for quartz oscillators.
2	Turn-on PGA and PGA Vref, turn on the comparator and set the zero-cross offset voltage, typically 700mV plus maybe the offset. Regulate the offset and apply .	
	CR_USM_PRC (Ultrasonic Measurement Processing)	**USM_RLS_MODE** distinguishes between “First hit only” and “Release delay”: 0 := Start hit release condition by First hit only 1 := Start hit release condition by Release delay
	CR_USM_AM (Ultrasonic Amplitude Measurement)	**PGA_MODE** enables the PGA. 0 := disabled, 1 := enabled. IN case the internal PGA is disabled, RECEIVE_SIG and COMP_IN need to be connected. Alternatively, an external amplifier can be connected. **PGA_TRIM** sets the gain of the PGA via trim bits in steps of (DC gain) 0 := 2 V/V, 2 := 4 V/V, 4 := 7 V/V, 6 := 14 V/V, 1 := 3 V/V, 3 := 5 V/V, 5 := 10 V/V, 7 := 19 V/V The final gain depends on the fire frequency due to the limited bandwidth of the PGA. At 1 MHz the gain is about 17V/V. **ZCD_FHL_INIT** FWD copy of initial value for first hit levels Only important if autoconfig release code is set. Then, during the boot process, the according FWD cell content is copied into this register (with no further effect) but also into the register SHR_FHL_U & SHR_FHL_D from which first hit levels are supplied for dynamic operation
3	Send fire burst	
	CR_USM_FRC (Ultrasonic Measurement Fire & Receive Control)	**TOF_HIT_MODE**: The multi-hit mode defines whether the individual TOF data in the frontend buffer are according to a uniform receive burst (1, GP30 compatible) or a split burst. When set = 0 the 10 individual ToF data will be subsequent or split in three blocks. **FBG_CLK_DIV** is the clock divider for fire burst generator. Frequency = High speed clock divided by **FPG_CLK_DIV** 0,1: not allowed, 2 to 127: divided by 2 to 127 **FBG_PHASE_INS** defines the phase shift in case **MH-MODE** is set to 0. 1 LSB: 1/(2 * f(HS_CLK)), 0 := 2 LSBs, 1 := 2 LSBs, 2 to 255 := 2 to 255 LSBs **FBG_MODE** defines whether the inserted phase is low (0) or high (1) **FBG_BURST_PRE** / **FBG_BURST_POST** define the number of pulses in the initial sequence /ending sequence of the fire burst (0 to 63)
		Further details can be seen in section Multi-hit Modes
4	Receive enable. The receive path is not open instantly but after a defined window, relative to the fire burst. This allows to suppress any noise in between fire burst and receive signal.	
	CR_USM_PRC (Ultrasonic Measurement Processing)	**USM_NOISE_MASK_WINOW** defines the window as long any signal (e.g. noise) is masked on the receive path. The start time refers to the rising edge of 1st fire pulse. Offset: -0.4 µs, 1 LSB := 1 µs Typically the mask window ends close to the receive burst, but with enough margin to not end in the receive burst over the whole operating range. Note that, especially at high gain, the end of the noise mask window can cause distortions which should be masked by using a correspondingly configured start hit delay window.

Step	Register	Parameter
5		Receive burst detection: The receive burst detection includes several tasks. • Identification of the Start hit in the burst by means of first-hit level and / or release window • Do the ToF measurements according to the multi-hit mode setting • Pulse-width measurement of the 1st hit and Start hit • Sample & hold the amplitude before ToF measurement ends, then measure after the ToF measurement the discharge time to get a measure of the amplitude. For details about start hit modes and multi-hit modes please see the separate sections Start Hit Modes and Time-of-Flight Measurement
	CR_USM_PRC (Ultrasonic Measurement Processing)	**USM_TO** defines the timeout limit for the ToF measurement. In case there is no water, or one of the transducers is broken, no receive signal will be seen. To limit the time and also the current, a timeout window can be set accordingly. It should be as short as possible due to the geometry and operating range. 00 := 128 µs 01 := 256 µs 10 := 1024 µs 11 := 4096 µs
	CR_USM_TOF (Ultrasonic Measurement Time of Flight)	**TOF_HIT_START** defines the number of hits, including the first hit, before a TOF measurement is done by the TDC 0 is not allowed, 1 is not recommended 2 to 31
	SHR_USM_RLS_DLY_U (Ultrasonic Release Delay Up)	**USM_RLS_DLY_U** defines the delay window in up direction, after which start condition for multi-hits is released. The start time of the delay window refers to rising edge of the 1st fire pulse 1 LSB: 7.8125 ns (**TYP**) This window could be set e.g. in the part of the receive burst where the amplitude reached a stable value. It can also be used to mask distortions before the receive burst appears.
	SHR_USM_RLS_DLY_D (Ultrasonic Release Delay Down)	**USM_RLS_DLY_D** does the same in down direction
6		Pulse width measurement: The pulse width ratio is an indication of how close the initial offset of the comparator is to the peak of the first hit. If active, the pulse width of the first hit (offset V_{FHL} > 0) is measured as well as the pulse width of the Start hit ((offset V_{ZCD} = 0mV) are measured, the ratios for up and down are calculated and stored as ultrasonic pulse width ratios in data buffers **FDB_US_PW_U** and **_D**. This information can be used to do a first-hit level regulation. A good value is in the order of 0.6 to 0.7. In case the ratio exceeds the limits the first-hit level should be adjusted. Wrong readings of the pulse width are often caused by misinterpretation of noise or distortions before the receive burst as first hit. In such cases, use the delay windows to filter.
	CR_USM_AM (Ultrasonic Amplitude Measurement)	**PWD_EN** enables the pulse width detection (1 := active). For further details see section First Hit Detection.

Step	Register	Parameter
7		Amplitude measurement. The amplitude measurement starts with sample & hold of early amplitudes, beginning with the first wave. The number can be configured and is set relatively to the first hit. The conversion then, a discharge time measurement, starts after the end of the TOF measurement. For reference, in a calibration task the discharge times of two fixed voltages are measured. The features are: ● True peak amplitude measurement with every TOF (configurable) ● Highly reliable bubble and aging detection ● Very good consistency check in comparison to first hit detection ● Easy quality check in production and development ● Configurable number of hits to stop the amplitude measurement – this allows to measure the peak amplitude of each single wave at the start of the burst signal (but only one single value in each TOF measurement) The raw data are stored in the front-end data buffer in nanoseconds format: **FDB_US_AM_U** ≡ $AMUp[ns]$ **FDB_US_AM_D** ≡ $AMDown[ns]$ **FDB_US_AMC_VH** ≡ $AMChigh[ns]$ **FDB_US_AMC_VL** ≡ $AMClow[ns]$ Note: during operation, the relative time information is fully sufficient for amplitude comparisons and there is no need to calculate the voltage. For the calculation of the amplitude in mV see appendix Amplitude Calculation.
	CR_USM_AM (Ultrasonic Amplitude Measurement)	**AM_PD_START_MODE** defines when the amplitude peak detection starts. 0 := 10 us after the noise mask window expired. 1 := after the ultrasonic release delay expired. This is recommended, as the noise mask window also switches from fire to receive, which may generate a peak itself. **AM_PD_END** defines the end of the peak detection in number of hits after ultrasonic release condition. 0, 31 := not allowed 1 to 30 := after 1 to 30 detected hits
8		Measure in opposite direction With the end of the amplitude measurement, the measurement in one direction is finished. The measurement in the opposite direction follows after some time. This period is ideally in multiples of the power net frequency to suppress noise. With **USM_PAUSE** = 0 only one measurement will be done. The full actions from above apply to this second measurement.
	CR_USM_PRC (Ultrasonic Measurement Processing)	**USM_PAUSE** defines the pause time between two ultrasonic measurements. The typical setting is 1.0 * T(BF_SEL)= 20ms for 50Hz power net and 16.66ms for 60Hz power net, for optimal noise suppression 000 := no pause, only 1 measurement is done, in the direction set in **USM_DIR_MODE** 001 := not allowed 010 := 0.25 * T(BF_SEL) 011 := 0.5 * T(BF_SEL) 100 := 1.0 * T(BF_SEL) 101 := 1.5 * T(BF_SEL) 110 := 2.0 * T(BF_SEL) 111 := 2.5 * T(BF_SEL)
9		In flow meter mode, the CPU will start post processing as soon as the measurement has finished. Therefore, the chip needs a firmware. Users can write their own firmware, based on our sample code. Or they may use AS6031 with the ScioSense flow calculation code for water meters. At the end of the processing, the firmware has to set back the watchdog and to set the interrupt.
	CR_MRG_TS (Measure Rate Generator & Task Sequencer)	**TS_PP_F_EN**:= 1 enables the post-processing after an ultrasonic flow measurement use only with TS_MCM = 0 to evaluate results before they get possibly overwritten **TS_PP_T_EN**:= 1 enables the final post-processing at the end of the task sequence
		Set interrupt Depending on configuration, the interrupt is set by end of the task sequencer (time conversion mode) or by the firmware (flow meter mode).

Step	Register	Parameter
	CR_IEH (Interrupt & Error Handling)	**IRQ_EN_TSQ_FNS** :=1 sets the interrupt pin with the end of the task sequencer (time conversion mode or flow meter mode) **IRQ_EN_FW_S** :=1 sets the interrupt pin by the firmware, synchronized with firmware. (flow meter mode only; can be used to get an interrupt only on request, see below) An external controller can send command RC_COM_REQ to AS6031 to request a remote An external controller can send command RC_COM_REQ to AS6031 to request a remote communication at an arbitrary time. This causes that **COM_REQ** will be set in register **SRR_MSC_STF**. By polling this status flag, the firmware is able to trigger remote communication by setting **FW_IRQ_S** as described below.
	SHR_EXC (Executables)	**FW_IRQ_S,** 1:= Requests a firmware-triggered interrupt, synchronized with the task sequencer
	Remote communication With the interrupt pin being set, the remote controller can start communication. For details on the remote communication see section Remote Communication (Opcodes)	

Note: Avoid TOF timeout lower than fire burst length and avoid TOF timeout lower than Noise_Mask_Win. Otherwise TOF Timeout error is set and the USM sequence is not aborted after TOF timeout. TOF timeout has to take as much time as fire pulses or Noise Mask time is set.

8.5.2 Time-of-Flight Measurement

The AS6031 is based on TDC (time-to-digital converter) technology and uses a precise zero-crossing detection of the individual waves of the receive pulse to determine the travel time of the ultrasonic burst. To trigger not on any noise peaks in between fire and send pulse, AS6031 has three methods implemented:

- A first hit level detection implemented, which identifies the receive burst by setting a trigger level for the receive signal. Once the signal surpasses this level, the comparator level is set back to zero. This works fine with fast rising receive signals. It is supported by pulse width measurement and amplitude measurement.
- A delay window can be set to suppress any triggers. This window needs to be adjusted continuously to handle temperature drifts. This method may be sufficient in stable systems like heat meters, with slow amplitude variations and stable flow.
- New: Phase insertion in the fire burst. The phase jump can be detected in the receive data and gives a unique identification of the position within the receive burst.
- When signal frequencies approach half of the HSO frequency (e. q. 2 MHz signals when using a 4 MHz HSO) it is mandatory to introduce ignored hits. In such cases, the internal arithmetic unit is not fast enough to do all necessary calculations for each single hit, so at least every second hit must be ignored.

8.5.3 Multi-hit Modes

For the zero-cross detection itself, AS6031 has two multi-hit modes:

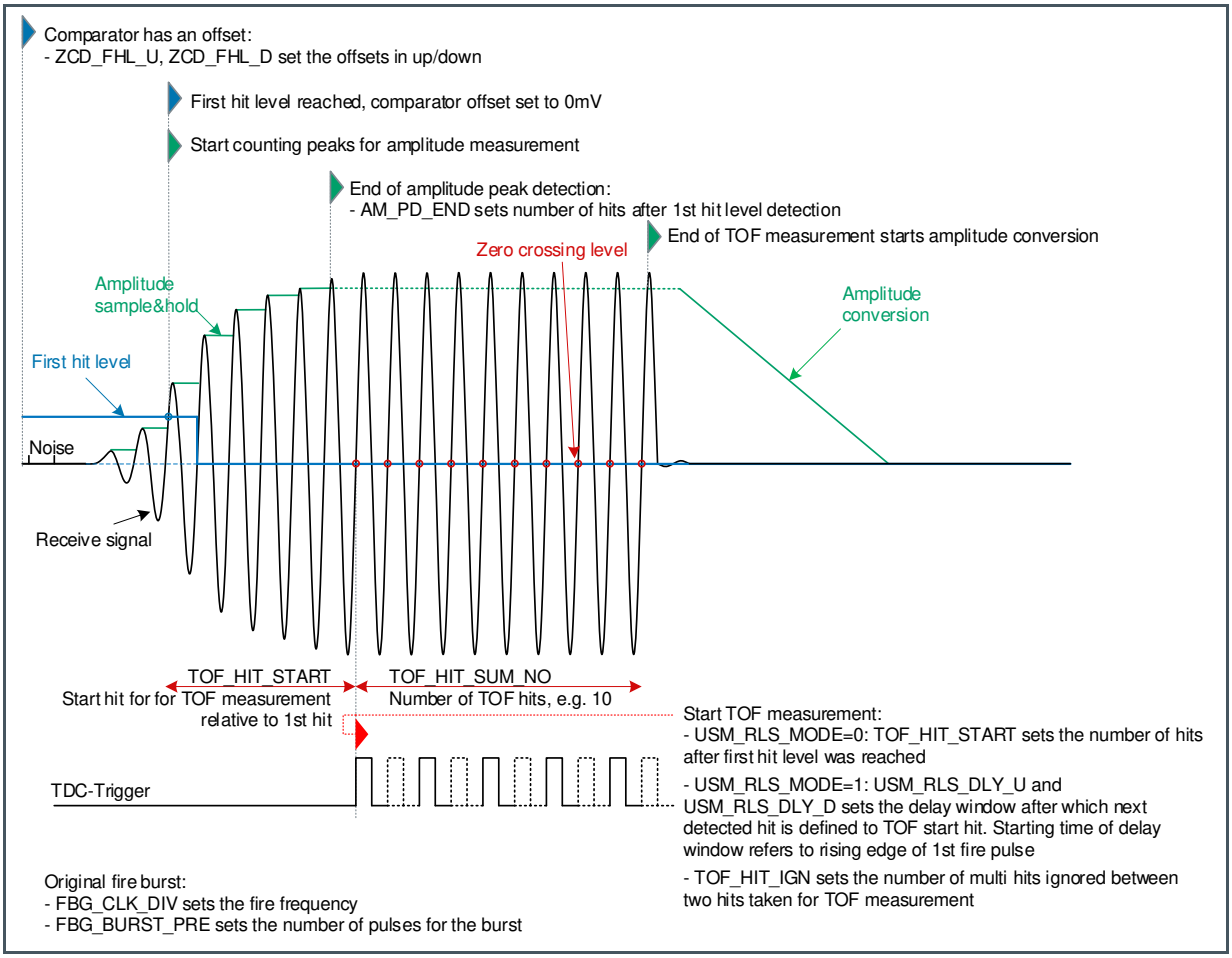
- **Uniform burst:** This is compatible to TDC-GP30. After first hit-level detection or release delay window opening, a set number of subsequent zero-crossings is measured.
  TOF_HIT_MODE = 1 adjust the TOF data output in the FDB up to 10 subsequent ToF data.
- **Split burst:** In this mode, in addition a phase shift is introduced. The fire burst is made of two sequences, separated by a phase shift defined through **FBG_PHASE_INS**.
- **TOF_HIT_MODE** = 0 The ToF data are also split, in three segments to identify the phase jump.

Uniform burst (GP30 compatible)

The following diagram shows in detail the receive signal and in multi hit mode TOF_HIT_MODE = 1, which is compatible to TDC-GP30.

Note: In this mode, TOF_HIT_MODE =1, it is mandatory to set TOF_HIT_END =127.

Figure 46:
Time-of-flight measurement



Split burst (phase insertion)

In AS6031 a new additional method for burst identification is implemented, called phase insertion. The fire burst splits in two parts, an initial sequence and an ending sequence. Each part has its number of pulses, and the inserted phase can be set to a phase shift defined through **FBG_PHASE_INS**

While the piezo oscillation transits from the imposed initial sequence to the ending sequence, a deviation in the periods of the receive burst can be observed. The beginning of this deviation has a fixed relation to the phase shift in the fire burst and therefore can be used to identify the position of the hit numbers.

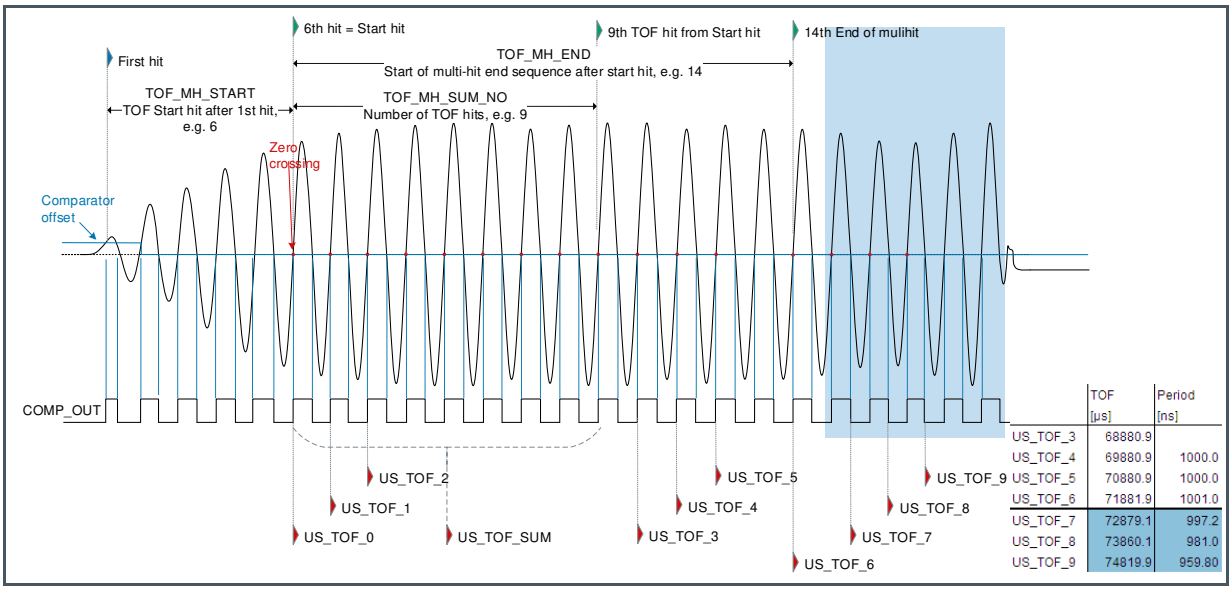
With TOF_HIT_MODE = 0, the TDC measures the set number of TOF hits, three additional hits following, and additional four hits starting with the end of multi-hit setting. In the front-end data buffer, the user can read

- US_TOF_SUM_OF ALL _U/D, Sum over all hits as set in TOF_HIT_SUM_NO
- The first three hits, US_TOF_0_U/D, US_TOF_1_U/D and US_TOF_2_U/D
- Additional three hits after the number of hits set for multi-hit, US_TOF_3_U/D, US_TOF_4_U/D and US_TOF_5_U/D
- Four additional hits, beginning with the hit as set in multi-hit end, TOF_HIT_END.

Following figures show a typical sequence.

Figure 47:
Split Fire Burst

Figure 48: Receive with Split Burst, TOF_HIT_MODE = 0



Relevant Registers

The table below lists most important registers and parameters for setting this multi-hit mode.

Figure 49:
Multi-hit Mode Relevant Registers

Register	Parameter	Description
CR_USM_FRC (Ultrasonic Measurement Fire & Receive Control)	**FBG_CLK_DIV**	Clock divider for the fire burst generator frequency, 2 to 127
	FBG_MODE	0 := Insertion of low phase 1 := insertion of high phase
	FBG_PHASE_INS	Size of inserted phase in multiples 2 to 255 of 1 LSB = $1/(2\times f_{HS\_CLK})$ 0, 1 and 2 set all 2 LSB Recommended is 3×LSB or 5×LSB
	FBG_FIRE_BURST_PRE	Sets 1 to 63 pulses for the initial sequence (pre-burst). This number shall take into account TOF_HIT_START + TOF_HIT_SUM_NO plus a small number for the very first waves before 1st-hit level detection
	FBG_FIRE_BURST_POST	Sets 1 to 63 pulses for the ending sequence (post-burst). This number plus the pre-burst number shall take into account TOF_HIT_START + TOF_HIT_END plus minimum 3
	TOF_HIT_MODE	= 0 sets the TOF data output in the FDB to multi-hit mode
CR_USM_TOF (Ultrasonic Measurement Time of Flight)	**TOF_HIT_START**	Defines the number of hits, including the first hit, before a TOF measurement is done by the TDC 0 is not allowed, 1 is not recommended 2 to 31
	TOF_HIT_SUM_NO	Number of hits taken for sum value of TOF measurement 1 to 31 Note: The sum of TOF values may not exceed 16 ms @ 4 MHz, 8 ms @ 8 Mhz. Individual TOF values shall not exceed 4 ms @ 4 MHz, 2 ms @ 8 MHz

Register	Parameter	Description
	TOF_HIT_END	Defines the hit, counted from the Start hit, that triggers the multi-hit end sequence 1 to 127
	TOF_HIT_IGN	Number of ignore hits between two hits taken for TOF measurement. Must be set to 1 or higher if incoming multihits has a shorter distance than 400ns (> 2.5 MHz receive burst frequency)

Calibration of Zero Crossing Detection

The zero line of the receive signal is structurally given by the hard-coded Vref level (typically 0.7 V). The zero cross detection level V_{ZCD} is the corresponding reference level of the comparator and is defined in register **SHR_ZCD_LVL**. To ensure that the comparator correctly detects zero crossings of the signal, V_{ZCD} has to be calibrated to V_{ref} regularly. Basically, this compensates the offset of the comparator. The calibration is automatically done once after power-on, and then at a rate defined in register **CR_USM_PRC**, **ZCC_RATE**. In typical applications it is sufficient to do the zero-cross calibration with every 100^{th} cycle, having **ZCC_RATE** = 7.

The calibration automatically updates the value in **SHR_ZCD_LVL**, such that the user does not need to take any action. Note that the value in **SHR_ZCD_LVL** may be changed by the user, but such changes are overwritten by the next comparator offset calibration, except ZCC_RATE = 0.

8.5.4 Start Hit Modes

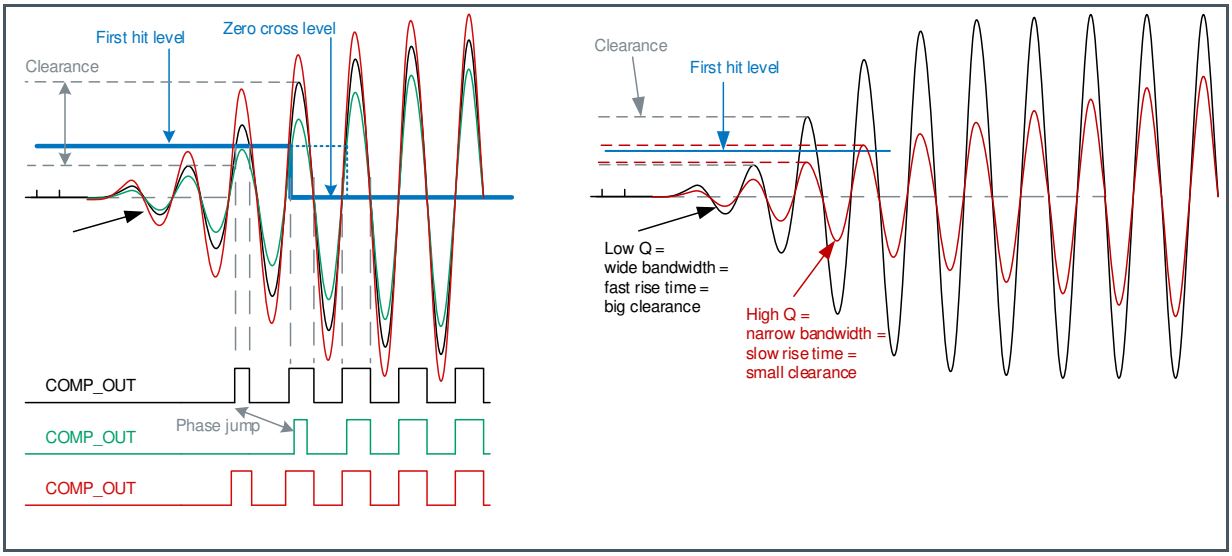
First Hit Detection Only

To do a time-of-flight measurement, the received signal needs to be identified and its arrival time needs to be measured thoroughly. This can be done by defining a first wave which results in a first hit, and then counting subsequent hits and storing the relevant arrival times. This is elaborated in the following manner: The receive signal, typically a burst-like signal, is converted into a digital signal using an internal comparator. While receiving, the reference voltage of the comparator most of the time equals the zero line of the receive signal to identify zero crossings (Actually, the zero line is the overlaid reference voltage V_{ref}, and the comparator's reference is set to the zero cross detection level V_{ZCD}, which is calibrated to V_{ref}). This way, received wave periods are converted into digital hits. To determine an absolute numbering of the hits, a so-called first **hit** is defined by adding a well-defined voltage level, the first hit level (V_{FHL}), to the comparator's reference. This first **hit** detection, at a comparator level which differs from the zero-cross level, is implemented to make the time-of-flight measurement independent from temperature and flow. The offset level V_{FHL} practically represents the level of receive signal at which the first hit is detected, which generates the first hit. After the first hit was detected, the comparator's reference is brought back to zero cross detection level (V_{ZCD}) at the 2nd hit, and the subsequent hit measurements are done at zero crossing. The following parameters define the first hit detection and the TOF hits:

- The trigger level ZCD_FHL, which defines the comparator offset level V_{FHL}
- The count number of the first subsequent TOF hit (Start hit) which is actually measured
- The number of measured TOF hits

- The interval between measured TOF hits
- The ultrasonic release delay: This delay disables hit detections for some defined lead time. It enables to suppress noise, in addition to the noise mask

Figure 50:
First Hit Level Detection



Starting the measurement with the comparator offset V_{FHL} different from zero, e.g. 100 mV, helps suppressing noise and allows the detection of a dedicated wave of the receive burst that can be used as reference. Once this first wave is detected, the offset is set back to the zero cross detection level V_{ZCD}. It is recommended to start actual TOF hit measurements after at least two more wave periods. The maximum value for the V_{FHL} is 175 mV.

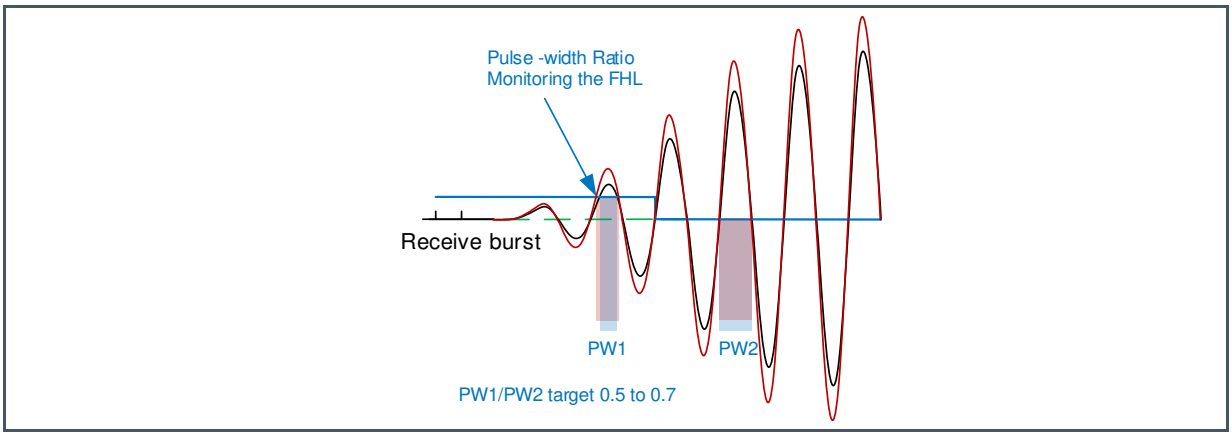
AS6031 allows to set different first hit levels for up and down measurement. This allows to handle transducer pairs with different amplitudes. This e.g. appears if the sensors are pressure sensitive and there is some pressure loss across the measurement path. The combination will lead to different amplitudes in up and down direction with increased flow.

In general, the first-hit level method needs a fast increase of the receive burst's amplitude. In other words, it is not suited for high-Q transducers that take a long time for settling, because then the clearance (amplitude difference from wave to wave) is not sufficient.

Pulse Width Measurement

The information of pulse width is used to monitor how close the first hit level is to the margins. Therefore, the device measures the pulse width of the first hit, with the comparator being at the first-hit level. This is compared with the pulse width of the Start hit, where the comparator is at zero cross level. By nature , the ratio is less than one. If the amplitude decreases at a given first hit level, the pulse width will decrease, too. At the point when the amplitude of the current wave falls below the first hit level, the next wave will be taken for first hit. The pulse width ratio will jump from a very small value to a high value.

Figure 51:
Pulse Width Ratio



One method of regulating the first-hit level would be to set upper and lower limits for the pulse width ratio, e.g. 0.5 to 0.7. Then, when the pulse width ratio reaches those limits, the FHL level is increased or decreased accordingly.

The result of the pulse width measurement is stored in the frontend data buffer.

Figure 52:
FDB for Pulse Width Measurement

Addr	Name[1]	Description
0x081	**FDB_US_PW_U**	Ultrasonic Pulse Width Ratio Up
0x085	**FDB_US_PW_D**	Ultrasonic Pulse Width Ratio Down

This data format is an 8-bit fixed point number with 7 fractional bits, ranging from 0 to 1.992. For further details on first hit level regulation please refer to our application note.

Relevant Registers

The table below lists most important registers and parameters for setting this first wave detection.

Figure 53:
Relevant Registers for First Wave Detection

Register	Parameter	Description
CR_USM_AM (Ultrasonic Amplitude Measurement)	**PWD_EN**	Enables pulse width detection 0: off 1: on
	ZCD_FHL_INIT	FWD copy of initial value for first hit levels Only important if autoconfig release code is set. Then, during the boot process, the according FWD cell content is copied into this register (with no further effect) but also into the register SHR_FHL_U & SHR_FHL_D from which first hit levels are supplied for dynamic operation

Register	Parameter	Description
SHR_ZCD_LVL (Zero Cross Detection Level)	**ZCD_LVL**	Zero Cross Detection Level 1 LSB: ~ 0.88 mV
SHR_FHL_U (Zero Cross Detection Level)	**ZCD_FHL_U**	First Hit Level Up 1 LSB ~ 0.88 mV, maximum 175mV
SHR_FHL_D (First Hit Level Down)	**ZCD_FHL_D**	First Hit Level Down 1 LSB ~ 0.88 mV, maximum 175mV

Release Window Only

Some applications may use transducers with high Q = narrow bandwidth, with the disadvantage of a very slow settling time. This ends with very small amplitude differences from peak to peak, which makes it difficult to use first-hit level detection. In this case, a fixed but programmable window can be used that releases the TDC input after a fixed but programmable time.

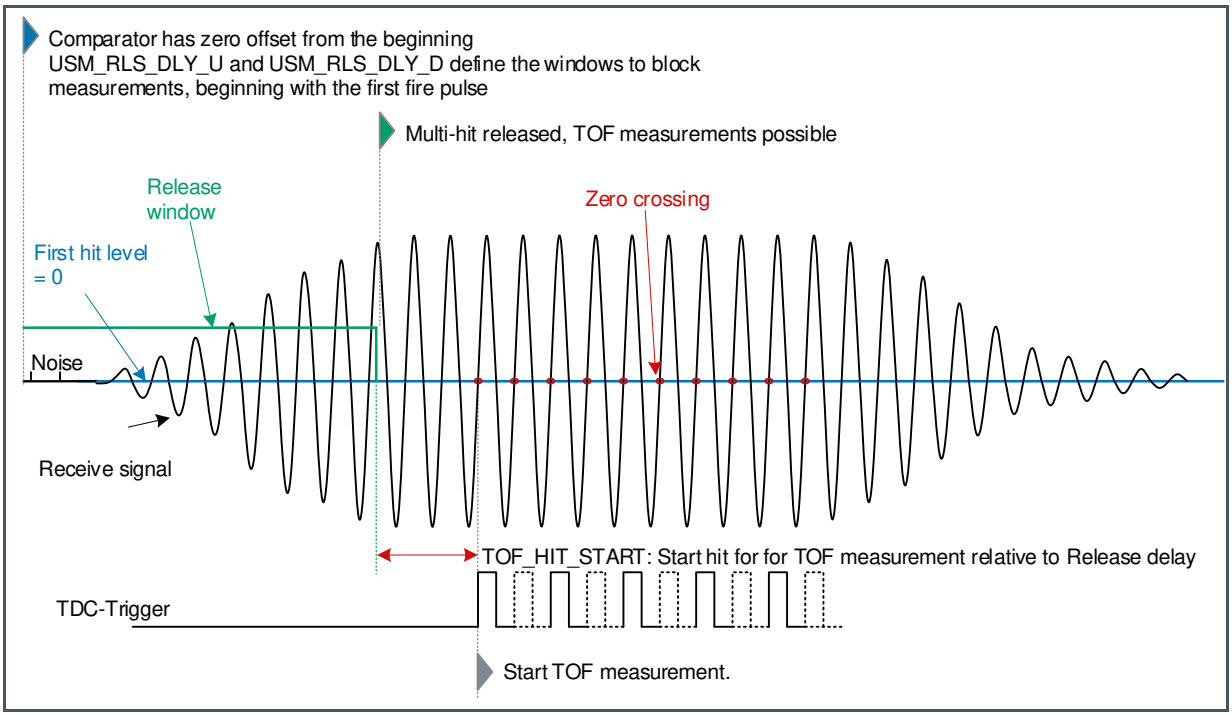
The time starts with the first fire pulse and can be programmed for up and down direction independently. The precision of the 19-bit values is 1LSB = 7.8125 ns with 4MHz.

Register **SHR_USM_RLS_DLY_U** Parameter **USM_RLS_DLY_U**
Register **SHR_USM_RLS_DLY_D** Parameter **USM_RLS_DLY_D**

The challenge here is to track and control the window by software to cover drifts over temperature (speed of sound).

Following figure shows the situation with release window instead of first-hit level detection.

Figure 54:
Release Window



Combination First Hit Level plus Release Window

Ideally, the first hit level detection is combined with release window. This combination can be activated in configuration register CR_USM_PRC (Ultrasonic Measurement Processing), setting bit **USM_RLS_MODE** = 1 and having a first hit level FHL ≥ 0. **TOF_HIT_START** is counted after the first hit level detection before the start hit is taken.

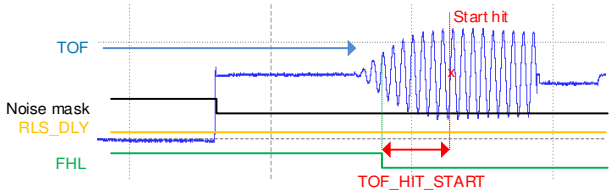
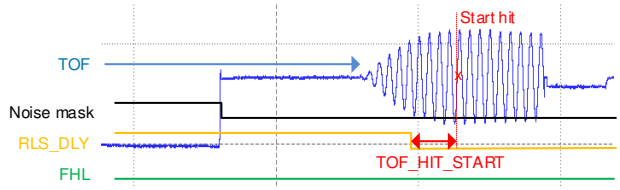
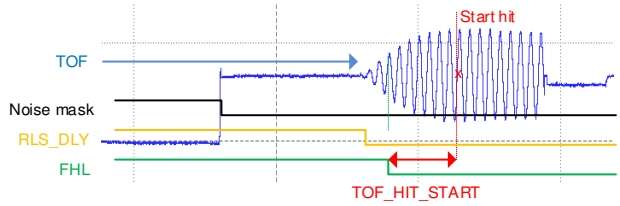
The release window should be set to the lowest time-of-flight possible over the whole temperature range. This window will suppress noise in advance to the receive burst. Note that the noise mask window switches between send and receive path. Switching the capacitive load will generate noise or oscillations with the noise mask opening. By means of the release delay the first hit level detection will work properly.

The right choice for TOF_HIT_START depends on which zero crossing is the first to be sufficiently stable and low noise for taking it into account.

Summary

Global settings:
TOF_HIT_START ≥ 2 (in CR_USM_TOF[5:1])

Figure 55:
Start Hit Modes

USM_RLS_MODE	USM_RLS_DLY_U/D	FHL	Start Hit Mode / Release Event	Comment
0	= 0	> 0	First hit detection only / First hit	Start hit = TOF_HIT_START hits after first hit detected Independent from temperature, but sensitive to noise from noise mask opening. 
1	> Noise mask, >TOF	= 0	Release delay only / Release delay	RLS_DLY within the receive burst, needs to be adjusted with temperature Start hit = TOF_HIT_START hits after release window opening 
1	> Noise mask, < TOF	> 0	Combined / First hit	RLS_DLY shortly before the receive burst (to suppress noise as generated by the noise mask switching) Start hit = TOF_HIT_START hits after first hit detected No need to adjust the RLS_DLY with temperature due to FHL 

8.5.5 Amplitude Measurement

The amplitude measurement is done by a single slope AD-conversion of a stored peak amplitude value. In practice, this means a sample & hold detector stores the amplitude peak value during the measurement interval (between the first wave and the configured end of the measurement) in a capacitor. This capacitor is then discharged at a constant current down to Vref, which yields a discharge time measured by the internal TDC.

The amplitude measurement has its limitation especially at high fire frequencies (> 1MHz) because the sample circuit can't follow the amplitude fast enough. Therefore, the measured amplitude for the first

rising waves will be lower than the real one. Especially at high fire frequencies (2 MHz +) the amplitude measurement is close to the real value only when the sample ends in the plateau of the burst or even the end of the TOF measurement.

The amplitude measurement is calibrated against two reference level measurements at nominal offset levels of Vref and Vref /2, respectively. From these two reference time measurements, slope and offset of the calibration curve can be calculated, which permits to calculate actual values for the measured peak amplitudes. The rate and interval length of amplitude measurements, and the rate of calibrations can be configured.

Figure 56:
Amplitude Measurement

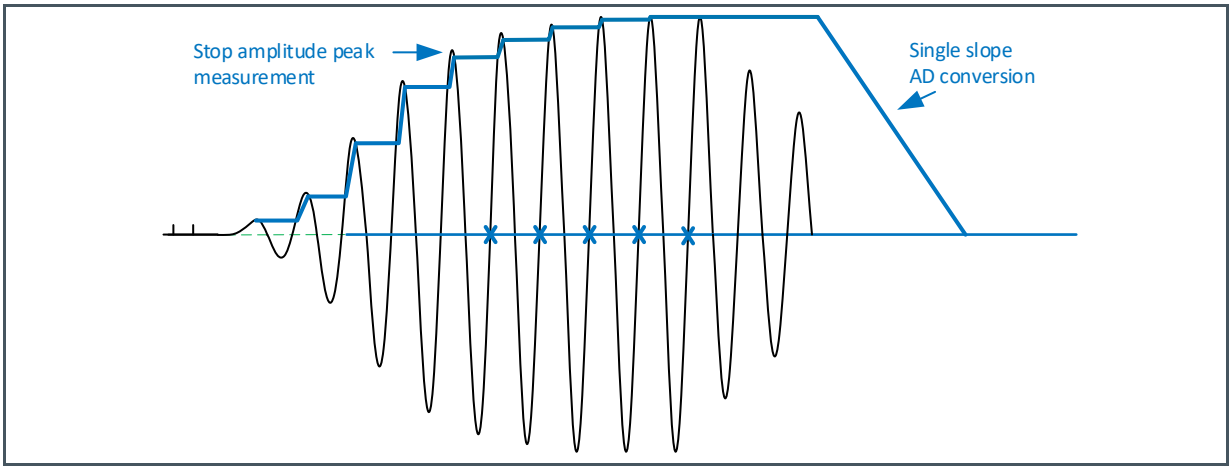


Figure 57:
Relevant Registers Amplitude Measurement

Register	Parameter	Description
CR_USM_AM (Ultrasonic Amplitude Measurement)	**AM_RATE**	Amplitude measurement rate, maybe off or every single, 2nd, 5th, 10th, 20th, 50th, or 100th TOF measurement. Usually it should be done with every TOF measurement. So, **AM_RATE** = 1 is recommended.
	AM_PD_END	End of peak detection, defined by number of detected hits 0, 31: not allowed 1 to 30: after 1st to 30th detected hit It depends on the rise time of the receive signal, which wave gives a reasonable amplitude information. With suitable transducers, having a fast rise time, a setting between 5 and 8 is a good choice. Necessary condition to avoid a time condition: AM_PD_END ≤ End of TOF measurement
	AMC_RATE	Defines the repetition of the amplitude calibration. Settings off, or every single, 2nd, 5th, 10th, 20th, 50th, or 100th amplitude measurement. In typical applications it is sufficient to have the amplitude measurement calibration with every 50th amplitude measurement, setting **AMC_RATE** = 50

Register	Parameter	Description
	AM_PD_START_MODE	0: AM peak detection starts 10 µs after noise mask window expires 1: AM peak detection starts after ultrasonic release delay expires Suitable only for combined start hit mode, when ultrasonic release delay is configured between end of noise mask window and ultrasonic receive burst

The resulting measurements are then stored as raw TDC values in the frontend data buffer.

The format of the TDC values is 32-bit data with 1 LSB: $1/ 2^{16} * t_{period(HSO)}$, $t_{period(HSO)}$ = 250 ns at 4 MHz, = 125 ns with 8 MHz.

Figure 58:
FDB in case of ToF

Addr	Name[1]	Value	Description
0x082	**FDB_US_AM_U**	AM_{up}	Ultrasonic Amplitude Value Up
0x083	**FDB_US_AMC_VH**	AMC_{high}	Ultrasonic Amplitude Calibrate Value High
0x086	**FDB_US_AM_D**	AM_{down}	Ultrasonic Amplitude Value Down
0x087	**FDB_US_AMC_VL**	AMC_{low}	Ultrasonic Amplitude Calibrate Value Low

Those time data are converted into voltage by means of the formulas given in appendix 15.6.

It is, however, not necessary to calculate actual amplitudes in mV since the measured time values themselves can be used for relative amplitude comparison. In this case, the calibration values are used in reverse way to derive time values for amplitude comparison, for example from given limits.

While the amplitude measurement is repeatable and stabilized through calibration, it is still not a high-precision measurement. In the final measurement result an offset of a few mV typically remains, that also depends on the fire frequency. Since amplitude measurement always starts at the first wave, it should be clear that the result can never be smaller than the first hit detection level V_{FHL}.

Note that the peak detection can be configured to end at any hit after start condition (first hit level or hit release delay). This way it is possible to measure the peak amplitudes of receive burst hits during its rise individually. Of course, this requires several separate measurements with different configurations. As an alternative, the amplitude measurement can be stopped after the first wave already, and the first hit level is then increased in steps of 1mV. The resulting data can be used as basis for the selection of the right first hit level.

8.6 Temperature Measurement

AS6031 has a highly accurate interface for resistive temperature sensors. All these resistive measurements are based on discharge time measurements, using a fixed load capacitor and discharging this one sequentially through the sensor resistors and a reference resistor. The sensors' resistance is defined by calculating the ratio of sensors' discharge time and reference resistor's discharge time.

The unit can handle various modes, based on following possible configurations:

- An internal temperature sensor with 3000ppm/K and an internal reference with100ppm/K
- 1 or 2 external temperature sensors in 2-wire or 4-wire connection

For precision temperature measurement with mK precision, as needed in heat meters, platinum resistors (PT500 or PT1000) are recommended. The external load capacitor should be of C0G material. X7R will need more fake measurement to get into a micromechanical stable mode. Heat meters will need two sensors, for hot water (incoming) and cold water (outcoming). Water meters will need a temperature sensor only in case of hot water meters with temperatures above 60°C. In cold water meters the temperature can be calculated from the sum of time-of-flight.

The mode selection will define the sequence of the activation of the internal switches. This includes also compensation measurements that correct for $R_{ds(on)}$ (switch resistance) and gain (comparator delay).

2-wire measurements are good for sensor temperature measurements or other resistor measurements with medium accuracy demands. They require calibration for their offset resistances. 4-Wire measurements are more accurate but require the according 4-Wire sensor cabling.

4-wire measurements compensate for cable and connection resistance. This is preferred in case of screwed or plugged sensor connections. This mode achieves a mathematical precision of 10mK.

Attention

Note. Both methods cannot compensate for variations in cable capacitance. Therefore, the AC-based measuring unit cannot be used for long cables (> 1.5m).

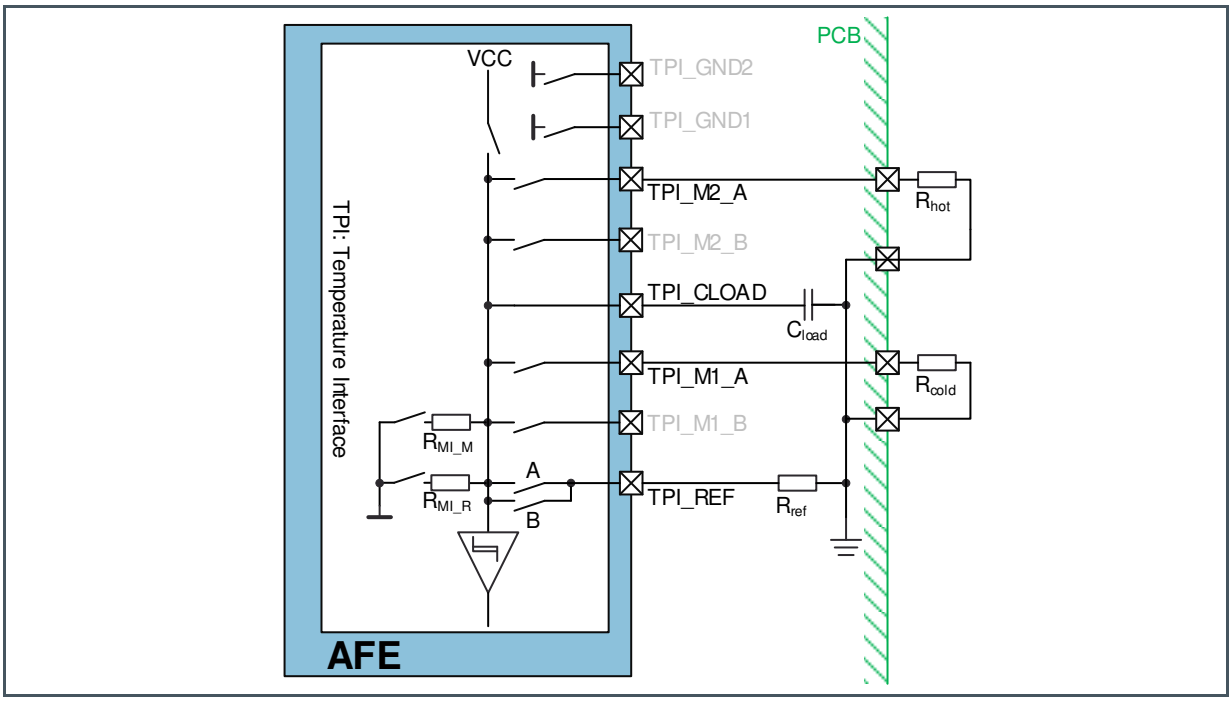
Information

In case the temperature measurement unit is not used, we recommend connecting a standard 100nF capacitor to pin TPI_CLOAD. All other pins may be left not connected.

8.6.1 2-Wire Temperature Mode

In 2-wire temperature mode, the temperature interface does a sequence of resistor measurements at the external ports, for 1 or 2 sensors and for the reference resistor. The sensors are connected to the chip as drawn in the next figure:

Figure 59:
Temperature sensor, 2-wire connection



A measurement sequence for 2 sensor looks like the following:

1. 2 fake measurements with C0G capacitors (8 for X7R capacitors). The capacitor is discharged but without any time measurement. This is to stabilize the load capacitor. They are followed by the first measurement **M1**.
2. t_{M1A} is measured by closing the switch for the resistor at pin TPI_M1_A
3. t_{M2A} is measured by closing the switch for the resistor at pin TPI_M2_A
4. t_{RAB} for the reference at pin TPI_REF by closing both switches.
5. t_{Rdson} for the $R_{ds(on)}$ compensation, measuring at pin TPI_REF by closing one switch only.
6. t_{gain} for the gain compensation, with the switches at TPI_REF and TPI_M1A being closed
7. Repetition of sequence 2. To 6. In reversed order.

Results of these five measurements are stored in the data buffer:

Figure 60:
FDB in case of Temperature Measurement

Addr	Name	Unit	Seq.	Time	Description
0x080	**FDB_TPM1_M1AB_RAB_G12**	C	1	t_{gain}	Gain compensation
0x081	**FDB_TPM1_RAB_G12**			t_{RAB}	Reference port REF-AB
0x082	**FDB_TPM1_M1A_G12**	T		t_{M1A}	Temperature port M1-A

Addr	Name	Unit	Seq.	Time	Description
0x083	**FDB_TPM1_M2A_G12**			t_{M2A}	Temperature port M2-A
0x084	**FDB_TPM1_RA_G12**	C		t_{Rdson}	RDSON compensation
0x08E	**FDB_TPM2_M1AB_RAB_G12**	C	2	t_{gain}	Gain compensation
0x08F	**FDB_TPM2_RAB_G12**			t_{RAB}	Reference port REF-AB
0x090	**FDB_TPM2_M1A_G12**	T		t_{M1A}	Temperature port M1-A
0x091	**FDB_TPM2_M2A_G12**			t_{M2A}	Temperature port M2-A
0x092	**FDB_TPM2_RA_G12**	C		t_{Rdson}	RDSON compensation

The raw results given here are raw TDC values. They can be converted to actual times by multiplying with $t_{HSO}/2^{16}$. But since in the final calculation only ratios of values will be used, this conversion into time is not needed.

t_{M1A}, t_{M2A} and t_{RAB} correspond to the total resistance values of the measured network, including internal switch resistances. To remove the influence of switch resistances and comparator delay, measurements t_{Rdson} "and" t_{gain} should be used according to the following equations(1):

$R_{ds(on)}$ correction (the correction of switch resistances) $t_{RO} = t_{Rdson} - t_{RAB}$

Schmitt trigger delay compensation(2) $\Delta t = 2t_{gain} - 2\frac{t_{M1A}\,t_{RAB}}{t_{M1A}+t_{RAB}}$

Note that the Schmitt trigger delay compensation requires a measurement of the cold sensor. In case one sensor may be optional, always use the hot sensor for the optional one.

Reference: $t_R = t_{RAB} - t_{RO} - \Delta t$

Sensor Cold: $t_C = t_{M1A} - t_{RO} - \Delta t$ Hot: $t_H = t_{M2A} - t_{RO} - \Delta t$

The sensor to reference ratios are used in the following for the temperature calculation:

Sensor resistance vs. reference: Cold: $\frac{R_C}{R_{REF}} = \frac{t_C}{t_R}$ Hot: $\frac{R_H}{R_{REF}} = \frac{t_H}{t_R}$

(1) The calculation assumes that all double switches are identical, and that the measurements are linear and repeatable. Under these conditions, the sensor network resistances are measured to the accuracy of the reference resistor, with an uncertainty through added noise of ±0.001 % of full scale. Additional line resistances in the sensor networks can't be calibrated out by 2-Wire measurements. If the line resistances are known from different measurements, they may simply be subtracted from the result in a separate calibration.

(2) The calibration measurement for comparator delay uses the cold sensor. If the cold sensor path is unusable, for example due to some damage, also the hot sensor path can't be fully calibrated. So, if only one sensor is used, always use the cold sensor pins, and if one sensor result is needed with high accuracy, even in case the other sensor may be damaged, connect the more important sensor to the cold sensor ports

The calculation of temperatures from resistance values is done as usual through the sensor's characteristic T(R) curve. For convenience, there are ROM routines (15.5.4) implemented to do the calculation by means of a polynomial of second degree or simply by a linear approach. The polynomial resembles the inverted R(T)-polynomial for PT (according to IEC 60751:2008) within 0°C and 100°C.

Equation ROM_TEMP_POLYNOM:

$$Temperature\ T[°C] = 10.115 \times \left(\frac{t_C}{t_R}\right)^2 + 235.57 \times \left(\frac{t_C}{t_R}\right) - 245.683$$ Format: fd16

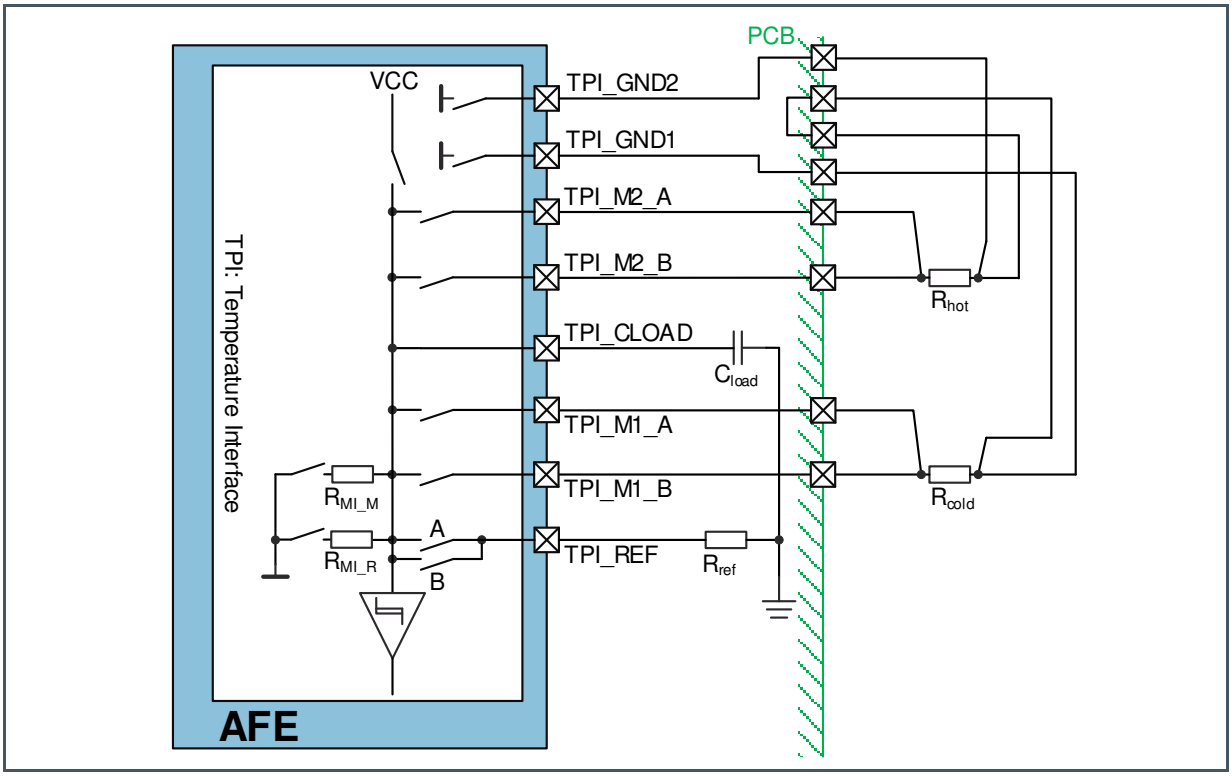
8.6.2 4-Wire Temperature Mode

In 4-wire temperature mode, the temperature interface does a much more complex sequence of resistor measurements in different configurations. The set of measurements allows an accurate determination of the sensor resistance of one or two 4-wire sensors.

In contrast to the 2-wire measurement, every single switch resistance of different pins can be calculated from these measurements. This is important, since every pin may be connected to the sensor over a different line resistance. There is no assumption about similar switches or similar line lengths. This makes this measurement method particularly suitable for high-accuracy measurements using connectors or other network elements that may suffer from aging.

The sensors should be connected to the chip as drawn in then following figure:

Figure 61:
Temperature sensor, 4-wire connection



In case only one 4-wire sensor is used, connect its two ground cables separately to TPI_GND1 and TPI_GND2.

The complex mathematics of the 4-wire measurements is fully covered by the applied firmware. Therefore, we do not provide a more detailed description. The applied firmware writes the results of the temperature measurement to RAM cells 0x020 to 0x024:

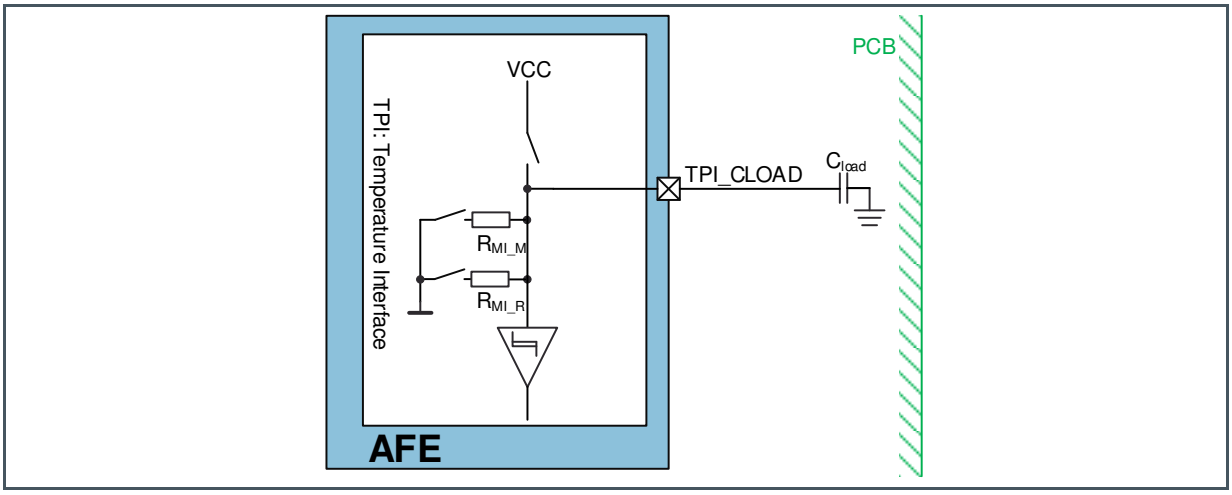
Figure 62:
Applied Firmware Temperature Result Registers

Addr	Name	Description	Format
0x020	**RAM_R_PTC_TEMPERATURE**	Cold sensor temperature [°C]	fd16
0x021	**RAM_R_PTH_TEMPERATURE**	Hot sensor temperature [°C]	fd16
0x022	**RAM_R_PTC**	Cold sensor resistance	fd16
0x023	**RAM_R_PTH**	Hot sensor resistance	fd16

8.6.3 Internal Temperature Measurement

A simple temperature measurement is possible through a build-in temperature-sensitive resistor and a temperature independent resistor.

Figure 63:
Internal Temperature Sensor



The internal temperature measurement can be used solely or in combination external temperature sensors in 2-wire connection.

A measurement sequence looks like the following:

1. 2 fake measurements. The capacitor is discharged but without any time measurement. This is to stabilize the load capacitor. They are followed by the first measurement **M1**.
2. t_{MI_M} is measured by closing the switch for the internal measurement resistor
3. t_{MI_R} is measured by closing the switch for the internal reference resistor
4. t_{MI_g} for the gain compensation is measured by closing both switches.
5. Repetition of sequence 2. To 5. In reversed order.

The results of the 3 measurements are stored in the frontend data buffer:

Figure 64:
FDB in case of Internal Temperature Measurement

Addr	Name	Unit	Seq.	Description
0x085	**FDB_TPM1_MI_R_G12**	I	1	Internal temperature reference
0x086	**FDB_TPM1_MI_RM_G12**			Internal temperature compensation
0x087	**FDB_TPM1_MI_M_G12**			Internal temperature measurement
0x093	**FDB_TPM2_MI_R_G12**	I	2	Internal temperature reference
0x094	**FDB_TPM2_MI_RM_G12**			Internal temperature compensation
0x095	**FDB_TPM2_MI_M_G12**			Internal temperature measurement

The internal temperature measurement utilizes an internal reference resistor with a nominal value of 1.265 kΩ at room temperature and a temperature coefficient of -0.1 Ω/K, and a sensor resistor with the same nominal resistance value, but a different temperature coefficient of 3.7 Ω/K. Due to chip tolerances, this measurement will not be very accurate. Under the assumption that the combined temperature coefficient of the resistance ratios is known as 3.8 Ω /K, a simple calculation can be done when a measurement at known chip temperature is performed:

$$T = \left(\frac{t_{MI\_M}}{t_{MI\_R}} - \frac{t_{MI\_M}(T_0)}{t_{MI\_R}(T_0)}\right) * \frac{1.265\text{k}\Omega}{3.8\Omega/\text{K}} + T_0$$

Due to chip tolerances, at least one calibration measurement at some temperature T_0 is recommended. It is actually sufficient to assume

$$\frac{t_{MI\_M}(T_0)}{t_{MI\_R}(T_0)} = 1$$

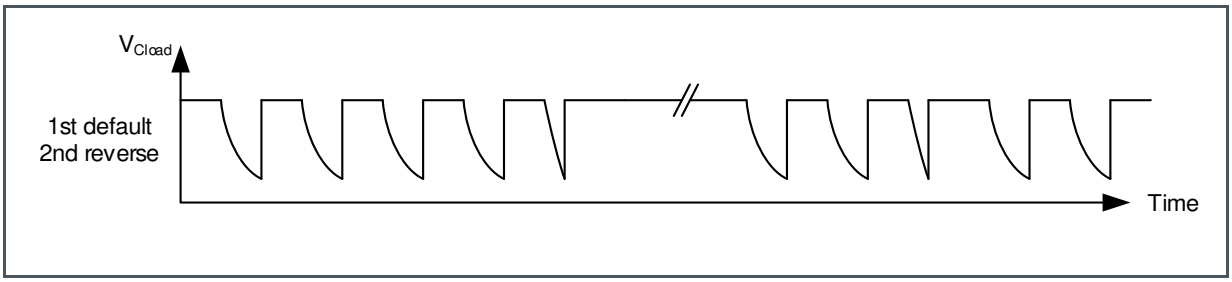
and to adjust the constant T_0 in the upper equation for the correct result.

More elaborate calibrations and calculations are possible. However, the internal temperature sensor is not accurate enough to justify that effort. It would also be possible to find an individual value for the internal sensor's temperature coefficient, but that would require a measurement at different temperatures. Nevertheless, with the simple calibration described above the internal temperature sensor can reach accuracies down to a few centigrade, which is good enough for some applications.

8.6.4 Technical Properties of the Temperature Interface

The temperature interface performs resistance measurements by discharging a capacitor, which was loaded to the supply voltage VCC, over the unknown resistor network, down to some fixed comparison voltage. Each temperature interface pin contains a double switch which connects this pin to C_{load} (the temperature measurement load capacitor on pin TPI_CLOAD). Through different switch settings, all necessary measurements are done in a well-controlled measurement sequence. The measurement sequences are hard-coded for 2-wire and 4-wire sensor case. Details on the sequences have been discussed above.

Figure 65:
Voltage C_{load} with Internal Temperature Measurement



In idle state the CLOAD port is connected internally to VCC. Externally the pin may be not connected.

Measurement range and selection of load capacitor

One discharge cycle takes a time of about

$$\tau = 0.7 \times R \times C_{load}$$

This makes e.g. 70 µs for a 1 kΩ resistor R and a 100 nF capacitor C_{load}.

The value of C_{load} and the range of R has to be chosen such that actual measurement times are between 10 µs (limited by the internal TDC calibration time) and the discharge cycle time minus the capacitor recharge time. The discharge cycle time t_{dct} can be chosen to 512 µs (recommended) or 1024 µs.respectively (**TM_CYCLE_SEL** in **CR_TM** set to 0 or 1, respectively). The capacitor recharge time can be estimated as

$$\tau_{re} = 10 \times 10\Omega \times C_{load}$$

for highest accuracy permit $3 \times \tau_{re}$. Typically, C_{load} = 100 nF is used, which permits to measure resistances between 142 Ω and about 5 kΩ (maximum measurement time 502 µs).

Considerations on measurement accuracy

The short times of the measurements, corresponding to high signal frequencies, have to be considered when using the temperature interface:

- Long lines can be problematic through their parasitic inductance and capacitance. When using lines in the range of meters or even above, the quality of the measurements must be checked carefully.
- In addition, long lines may introduce problems through coupled noise and EMI. It is possible to reduce such problems by using filtering elements like ferrites or small capacitors, but the possible influence of such filters on the measurement has to be considered.
- It is also recommended to use a capacitor of C0G or other class-1-type. For example, X7R-types can suffer from memory effects and, of course, from high temperature coefficients, which may reduce measurement accuracy dramatically.

It is in any case recommended to control the quality of such resistance measurements in your actual measurement environment.

Measurement accuracy is achieved by a suitable calibration and based on the measurement of a well-known reference resistor. The equation for τ has no guaranteed accuracy and can only be used for resistance estimations. Still, high measurement accuracy is reached by comparisons, making use of the high short-term repeatability and linearity of the single resistance measurements. Thus, a typical measurement sequence includes the measurement of a well-known reference resistor, such that actual resistance values can be calculated from the ratios of measured times τ to τ_{REF}, the time result for the reference resistor measurement. The accuracy of such measurements depends on the following factors:

- Absolute accuracy, temperature dependence and aging of the reference resistor is of course fully traced to the measurement result. For high quality measurements, use a reference resistor with low tolerance and low TC.
- Offset line and switch resistances. Such offset values are unavoidable, their removal requires additional calibration measurements. The following sections describe how such measurements are setup and used in case of 2-wire and 4-wire measurements. Switch resistances also add to the reference resistor measurement, typical values for switch resistances in AS6031 are below 10 Ohms.
- Linearity and repeatability of measurements: Due to a stable comparison voltage, linearity can be considered ideal. The good short-term stability of the temperature interface can even be further improved:
    - All measurements needed are done in a well-controlled measurement sequence. The measurement sequences are hard-coded but can be configured as described below.
    - Typically, 2 fake measurements are done before each measurement sequence. This makes sure that each relevant measurement starts in a similar condition (all measurement taken into account did have at least two similar measurements before).
    - The measurement sequence can be repeated (configurable) after a fixed time of, for example, 1.5 times the base frequency period. Setting the base frequency to the local mains frequency (50 or 60 Hz) results in suppression of power line noise in temperature measurements.
    - The repeated measurement sequence can be configured to be in reverse order. This removes any linear deviation trend that may appear during measurements, for example changing sensor temperature during a measurement cycle.

- Finally, the noise of the TDC measurements, which are utilized for the temperature ports as well, set an absolute limit on measurement accuracy. With a typical single-shot peak noise level of about ±2.5ns, the equivalent peak noise in measured resistance ratios can be estimated as ±0.001 % of full-scale values (assuming 500us maximal measurement time). Note that this is an absolute error, such that the relative error increases for low measurement times or low resistances, respectively.

Measurement run time and current consumption

The total runtime t_{ti} of a temperature interface measurement sequence depends on the chosen configuration and can be calculated as

$$t_{ti} = t_{HSO} + t_{dct} * \left(n_{fake} + n_{meas}\right) + t_{pause}$$

Here, n_{meas} is the number of actual measurements (four with 1 sensor in 2-wire, five with two sensors in 2-wire, always fourteen for 4-wire case and 3 for internal measurements). t_{HSO} is the configured HSO settling time, $t_{dct}$the discharge cycle time (512 or 1024 µs), n_{fake} =2 or 8 the number of fake measurements and t_{pause} the configured pause time (could even be 0, then the measurement sequence is not repeated).

Example: The total runtime of a 4-wire 2-sensor measurement with 2 fake measurements, at 135 µs HSO settling time and 512 µs discharge cycle time and with 30 ms pause time is 38.33 ms. If a firmware is used to evaluate the results, about 0.5 ms runtime is added. It should be clear that the total measurement time is dominated by the pause time. Note that the pause time starts together with the first measurement sequence, such that the first measurement sequence takes place during pause time.

Of course, the average current consumption depends strongly on the total runtime t_{ti} as well as on the temperature measurement rate or its frequency f_{tm}, respectively. A reasonable estimation of the additional average current consumption for the temperature measurement interface I_{tm} is

$$I_{tm} = 0.6\ \mu A * \left(n_{fake} + n_{meas}\right) * f_{tm} * \frac{C_{load}}{100\ \mathrm{nF}}$$

This formula gives an upper limit for repeated measurement sequence and 512 µs discharge cycle time, the value for 1024 µs is about 15% higher. At only one single measurement sequence (t_{pause} = 0), the actual current consumption is half of the calculated value.

Example: Average current consumption for external 2-wire 2-sensor measurements with 2 fake measurements and repeated ($t_{pause} \neq 0$), at a cycle time of 125 ms and **TM_RATE** = 240 (f_{tm} = 1/30 Hz) is estimated to 0.14 µA for 512 µs discharge cycle time or 0.161 µA for 1024 µs. At

(t_{pause} = 0), the result is 0.07 µA for 512 µs. Note that the current consumption does not depend on the measured resistance. Note also that a low repetition rate, as in the given example, the current consumption comes in shape of a peak at the temperature measurement frequency. It is recommended to use a 100 µF blocking capacitor on supply voltage for high quality temperature measurements.

8.6.5 Relevant Registers

The table below lists most important registers and parameters for setting this multi-hit mode.

Figure 66:
Multi-hit Mode Relevant Registers

Register	Parameter	Description
CR_TPM (Temperature Measurement)	**TM_RATE**	Temperature Measurement Rate 0: disabled 1 to 1023:Rate related to sequencer cycle trigger
	TPM_PAUSE	Pause time between 2 temperature measurements for 50Hz/60Hz suppression 00x: no pause, only one measurement 010: Pause = 0.25 * T(BF_SEL) ms 011: Pause = 0.5 * T(BF_SEL) ms 100: Pause = 1.0 * T(BF_SEL) ms 101: Pause = 1.5 * T(BF_SEL) ms 110: Pause = 2.0 * T(BF_SEL) ms 111: Pause = 2.5 * T(BF_SEL) ms
	TPM_MODE	Temperature Measurement Mode 000: Off 001: Internal only 010: Internal & 2-wire/1 port 011: Internal & 2-wire/2 ports 100: 2-wire/1 port 101: 2-wire/2 ports 110: 4-wire/1 port 111: 4-wire/2 ports
	TPM_PORT_MODE	Temperature Measurement Port Mode 0: Inactive ports pulled to GND while measurement (recommended setting)
	TM_PORT_ORDER	Temperature Measurement Port Order 10: 1. measurement: default order / 2. measurement: reversed order (recommended setting)
	TPM_CLOAD_TRIM	Temperature Measurement Load Trim 10: = recommended value
	TPM_CYCLE_SEL	Temperature Measurement Cycle Select 0: 512 µs (recommended value) 1: 1024 µs
	TPM_FAKE_NO	Number of Fake measurements 0: 2 fake measurements (recommended value)

8.6.6 Error messages

The temperature interface generates error messages, which can be read from register SRR_ERR_FLAG. There are three different error flags that may be set by the temperature interface:

- **EF_TM_SQC_TMO** (bit 8): Temperature Sequence Timeout. This flag is set when the first temperature measurement sequence did not finish before the end of the configured pause time. Note that this flag must be ignored when no second measurement block is configured, since then there is no pause time and the flag is always set.

- **EF_TM_SC_ERR** (bit 4): Temperature Measurement Short Circuit. This flag is set when the load capacitor is discharged very quickly, such that within the first 1 µs of the measurement the capacitor voltage drops below VCC /2. This indicates a too low resistance, which typically happens in case of a short circuit of a sensor.
- **EF_TM_OC_ERR** (bit 3): Temperature Measurement Open Circuit. This flag is set when the load capacitor is not discharged to the comparison voltage level during a measurement cycle at all. This indicates a too high resistance, which typically happens in case of an open circuit, for example a loose sensor

9 Special Functions

9.1 Time Stamp (RTC)

AS6031 has a simple timestamp function with a resolution of 1 sec. The current values for hours, minutes and seconds are latched in result registers. The time stamp can be updated automatically every measure cycle trigger. But it is possible to trigger an update as well as to clear the content, both by setting executables in the special handling register SHR_EXC.

Time stamp is cleared only by “Power On Reset” and therefore not influenced by any other system reset or system init.

Relevant Registers

The table below lists most important registers and parameters for setting the clock management.

Figure 67:
Relevant Registers for Time Stamp (RTC)

Register	Parameter	Description
CR_CPM (Clock- & Power-Management)	**TSV_UPD_MODE**	Time stamp update mode 0: updated by TSV_UPD in SHR_EXC 1: automatically updated with every measure cycle, use this setting
SHR_EXC (Executables)	**TSV_UPD**	Time stamp value update 0: No action 1: Update time stamp value from time stamp counter
	TSC_CLR	Time stamp counter clear 0: No action 1: Clears time stamp counter
SRR_TS_HOUR (Time Stamp Hours)	**TS_HOUR**	Timestamp hours, 18-bit values, 1 LSB: 1h
SRR_TS_MIN_SEC (Time Stamp Minutes & Seconds)	**TS_MIN**	Timestamp minutes, 8-bit values, 1 LSB: 1min, range 1 to 59
	TS_SEC	Timestamp seconds, 8-bit values, 1 LSB: 1sec, range 1 to 59

9.2 Backup

Backup handling in AS6031 can be realized by connecting an external I2C EEPROM to the GPIO Unit of the AS6031. Backup handling can be triggered by the integrated general-purpose timer, which defines the cycle time for the backup. It’s enabled if the CPU request enable **CPU_REQ_EN_GP** is enabled and triggered by the general-purpose timer.

The backup data has to be written by firmware code to the 2-wire interface, where backup data is directly transferred to an external I2C EEPROM.

For details on backup handling or different usage of the EEPROM interface please contact support.

Figure 68:
EEPROM Interface

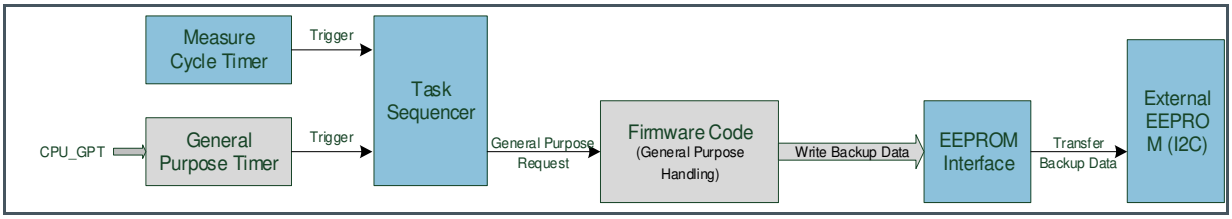


Figure 69:
Relevant Registers for Backup

Register	Parameter	Description
CR_IFC_CTRL Register (Address 0x0C1)	**I2C_MODE**	2-wire master interface mode, I2C like 00 & 11: I2C disabled 01: I2C enabled on GPIO 0/1 10: I2C enabled on GPIO 2/3
	I2C_ADR	2-wire master interface slave address

9.3 Watchdog

After a system reset the watchdog of AS6031 is enabled. The nominal value of the watchdog time is 15.2 seconds, based on the internal oscillator clock source of 8.7 kHz.

For operation in time conversion mode, it could be useful to disable the watchdog of AS6031. For that a disable code has to be written to register CR_WD_DIS.

When AS6031 operates with firmware, it is good practice to keep the watchdog enabled. The watchdog timer must then be reset before the watchdog time elapsed. Otherwise the watchdog issues a system init (please compare to chapter “Reset Management”). Typically, the watchdog timer is reset in each post processing cycle by the firmware command clrwdt. The firmware can then use the watchdog for any safety mechanism where a system init is required to resolve problems, just by not resetting the watchdog timer. And of course, the system init will be triggered when the firmware does not run at all. Of course, it must be made sure that the chip starts operating as desired after a system reset, for example by a suitable configuration and setting the autoconfig release code in FWD.

Figure 70:
Relevant Registers for Watchdog

Register	Parameter	Description
CR_WD_DIS Register (Address 0x0C0)	**WD_DIS**	Code to disable Watchdog: 0x48DB_A399, Write only register. Status of watchdog can be checked in **WD_DIS** in register **SRR_MSC_STF**

9.4 Supply Voltage Measurement

The voltage measurement is the only measurement task which is performed directly by the supervisor and not by frontend processing. It's automatically executed if VM_RATE > 0. The value of VCC is measured and can be compared to a low battery threshold.

Figure 71:
Relevant Registers for Voltage Measurement

Register	Parameter	Description
CR_CPM (Clock- & Power-Management)	**VM_RATE**	Repetition rate, every Nth measure cycle trigger
	LBD_TH	Threshold for low-battery detection 1 LSB: 25 mV LBD_TH = 0: 2.15 V LBD_TH = 63: 3.725 V
SRR_VCC_VAL (VCC Value)	**VCC_VAL**	Measured value of VCC voltage 1 LSB: 25 mV, default 0x2F VCC_VAL = 0: 2.15 V VCC_VAL = 63: 3.725 V
SRR_ERR_FLAG (Error Flags)	**EF_LBD_ERR**	Error flag, indicates if low battery is detected

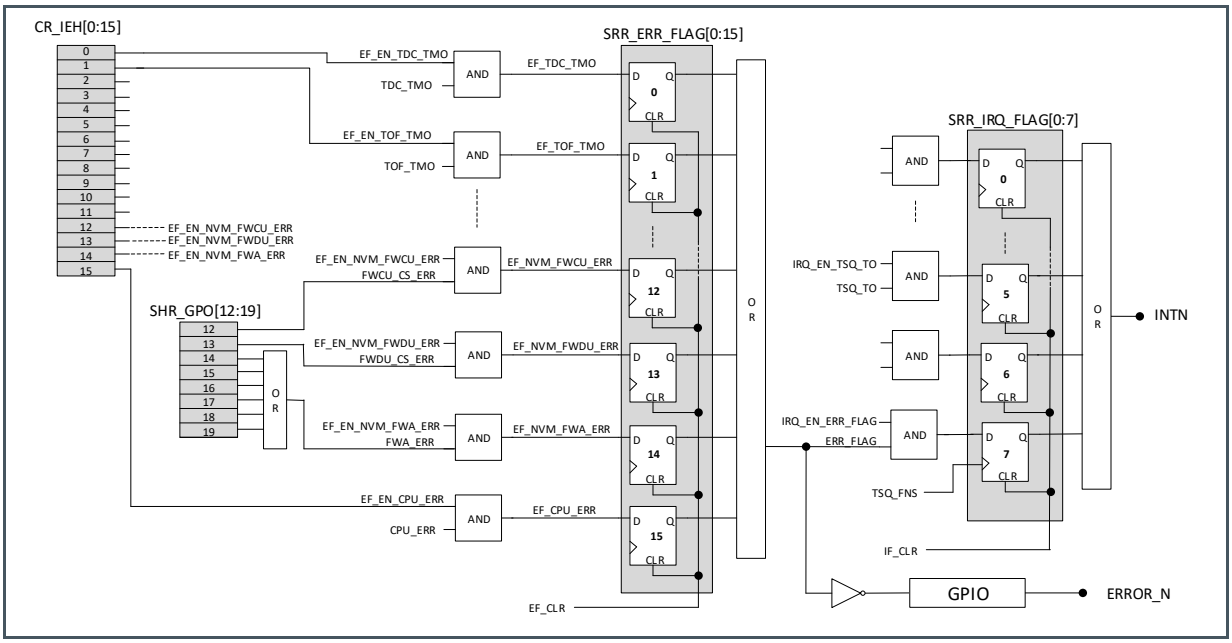
9.5 Error Handling

AS6031 features a number of hardware flags to indicate measurement errors. Some of the possible errors are related to wrong configurations, others to measurement problems. Typical error cases are TOF timeouts, when for example a water flow meter pipe ran dry and no signal was received, or a task sequencer timeout when the configured cycle time did not leave enough time to finish all tasks. Other examples would be short cuts or lost connections on temperature sensors.

It is in general recommended to check error flags, with or without firmware usage, and to implement processes to resolve the indicated problems.

Figure 72 sketches the relations between error flag enabling and signaling as implemented in AS6031 hardware. It is possible to generate an interrupt after an error (set IRQ_EN_ERR_FLAG in CR_IEH), and it is possible to configure which errors should be ignored (bits 15.0 in CR_IEH). As ERR_FLAG is updated when task sequencer cycle is finished (TSQ_FNS), the task sequencer timeout (TSQ_TO) should be additionally enabled too (set IRQ_EN_TSQ_TO in CR_IEH).

Figure 72:
Hardware Error Flags



Relevant Registers

The table below lists most important registers and parameters for error handling.

Figure 73:
Relevant Registers for Error Handling

Register	Parameter	Description
CR_IEH (Interrupt & Error Handling)	**EF_EN_XX_XX_XX**	Bits 0 to 15 enable error flags corresponding to register **SRR_ERR_FLAG**
	IRQ_EN_ERR_FLAG	Interrupt request enable for error flags
	IRQ_EN_TSQ_TO	Interrupt request enable for task sequencer timeout
SHR_GPO (General Purpose Out)	**FWCU_CS_ERR**	Set by NVRAM check routine in case of checksum error in FWCU
	FWDU_CS_ERR	Set by NVRAM check routine in case of checksum error in FWDU
	FWA_CS_ERR	Set by NVRAM check routine in case of any checksum error in applied FW (FWCA or FWDA)
	FW_ERR	Typically set by customized FW
	IF_CLR	Clears interrupt flag

Register	Parameter	Description
SHR_EXC (Executables)	**EF_CLR**	Clears error flag
SRR_IRQ_FLAG (Interrupt Flags)	**ERR_FLAG**	Error flag has been set
SRR_ERR_FLAG (Error Flags)	**EF_XX_XX_XX**	Bits 0 to 15 indicate Error flags corresponding to error flag enable bits in CR_IEH

9.6 Cyclic NVRAM Recall

It's recommended to perform a cyclic recall of NVRAMs, independent if operating in flow meter or time conversion mode. NVRAM recall means that memory is refreshed by transferring contents from non-volatile to volatile part of NVRAM.

Figure 74:
Relevant Registers for Cyclic NVRAM Recall

Register	Parameter	Description
CR_MRG_TS (Measure Rate Generator & Task Sequencer)	**TS_NVR_RATE**	NVRAM Recall Timer Rate 0000: disabled 0001: 1 sec 0010: 2 sec 0011: 5 sec 0100: 10 sec 0101: 30 sec 0110: 1 min 0111: 2 min 1000: 5 min 1001: 10 min 1010: 30 min 1011: 1 h 1100: 2 h 1101: 6 h 1110: 24 h 1111: 48 h

10 Interfaces

The UFC is able to operate in flow meter mode or in time conversion mode.

In flow meter mode a remote port interface is needed to program the UFC. In time conversion mode a remote port interface is needed to configure and for measurement related communication with the UFC. The remote port interface operates as an SPI interface.

Figure 75:
SPI Interface

Pin Name	Description	Comment
SSN	Input	low active (1)
MOSI	Input	(1)
SCK	Input	(1)
MISO	Output	3-state
INTN	Output	low active

(1) For standalone operation without external controller, the unconnected inputs should be configured with internal pull up by SPI_INPORT_CFG in CR_IFC_CTRL.

10.1 Serial Interface

The SPI interface is able to operate as a slave in a multi-slave SPI bus working in SPI mode 1. Pin MISO_TXD is in high Z state when the chip is not communicating.

SPI mode 1 (CPOL = 0, CPHA = 1) is defined as follows:

- Idle State of SCK is LOW
- Data is sent in both directions with rising edge of SCK.

Data is latched on both sides with falling edge of SCK.

Slave select (SSN) and slave interrupt (INTN) are low active.

It's strongly recommended to keep SSN = HIGH when SPI interface is in IDLE state.

10.1.1 Remote Communication (Opcodes)

Remote communication can be started within a task sequencer cycle, after frontend and post processing finished. AS6031 signalizes this instant to a remote controller by an interrupt via pin INTN.

In general, it is preferred to have an interrupt-based communication. Even though there is an asynchronous communication port (ACP), any communication during a measurement can have a negative impact on the measurement quality.

Generating an interrupt request for remote communication is served in following ways:

- Interrupt is automatically sent with every task sequencer cycle by enabling IRQ_EN_TSQ_FNS in CR_IEH. This is typically used in time conversion mode to allow the remote controller reading raw measurement values from frontend data buffer after each measurement.
- Interrupt is controlled by firmware. Therefore IRQ_EN_FW_S in CR_IEH has to be set. Then a synchronous interrupt can be triggered by firmware by setting FW_IRQ_S in SHR_EXC. The interrupt will appear after post processing finished. Typically used in flow conversion mode where a remote communication need not be requested with every task sequencer cycle.

In case of a running firmware, it is possible to avoid permanent communication and to keep the external controller in sleep mode over long periods. Since the firmware can evaluate and store results, it does not need to communicate after each measurement. It can then be advantageous to let the external controller trigger communication: An external controller can send command RC_COM_REQ to AS6031 to request a remote communication at an arbitrary time. This causes that COM_REQ will be set in register SHR_CPU_REQ. By polling this status flag, the firmware is able to trigger remote communication as described above on request by the external controller.

A remote control always starts communication with the UFC by sending a remote command RC_xx_xx as the first byte of a remote request. The following acronyms will be used:

Figure 76:
Acronyms

Acronym	Remote Command	Length
RC_	Remote Command	1 Byte
RAA_ADR	Random Access Area Address	1 Byte
RAA_WDx_Bx	Random Access Area Write Data	≥ 4 Bytes (4 bytes wise)
RAA_RDx_Bx	Random Access Area Read Data	≥ 4 Bytes (4 bytes wise)
FWC_ADR	FW Code Memory Address	2 Bytes
FWC_WDx_Bx	FW Code Memory Write Data	≥ 1 Byte

10.1.2 Reset & Inits

Figure 77:
Reset & Inits

Remote Command	Code	Description
RC_SYS_RST	0x99	Resets main part of digital core including register part and triggers bootloading process **Note**: Applicable only during debugging, not for application
RC_SYS_INIT	0x9A	Resets main part of digital core without register part and triggers bootloading process
RC_SV_INIT	0x9C	Resets Supervisor, Frontend Processing and CPU in main part of digital core but without a bootload trigger

10.1.3 Memory Access

Figure 78:
Memory Access

Remote Command	Code	Description
RC_RAA_WR	0x5A 0x5B	Write to RAM or register area Write to FW data area (NVRAM)
RC_RAA_WRS	0x5E 0x5F	Write to RAM or register area with read system status before write Write to FW data area (NVRAM) with read system status before write
RC_RAA_RD	0x7A 0x7B	Read from RAM or register area Read from FW data area (NVRAM)
RC_RAA_RDS	0x7E 0x7F	Read from RAM or register area with read system status before read Read from FW data area (NVRAM) with read system status before read
RC_FWC_WR	0x5C	Write to FW code area (NVRAM)
RC_RD_STATUS	0x8F	Read system status only

The least significant bits of remote commands RC_RAA_WR and RC_RAA_RD correlate to the most significant bit of the RAA address RAA_ADR[8]. RAA_ADR[7:0] are defined in a separate address byte.

In general, it is possible to do blockwise data transfer and this is the preferred operation.

Command details

Figure 79:
RC_RAA_WR (RAA Write in blocks)

Remote Request		Answer	
Command	RC_RAA_WR		
Address	RAA_ADR		
Write Data	RAA_WD0_B3		
	RAA_WD0_B2		
	RAA_WD0_B1		
	RAA_WD0_B0		
	RAA_WD1_B3		
	…		
	RAA_WDx_B0		

Figure 80:
RC_RAA_WRS (RAA Write in blocks with status first)

Remote Request		Answer	
Command	RC_RAA_WRS		
Address	RAA_ADR		
		Status	SYS_STATUS
Write Data	RAA_WD0_B3		
	RAA_WD0_B2		
	RAA_WD0_B1		
	RAA_WD0_B0		
	RAA_WD1_B3		
	…		
	RAA_WDx_B0		

Figure 81:
RC_RAA_RD (RAA Read in blocks)

Remote Request		Answer	
Command	RC_RAA_RD		
Address	RAA_ADR		
		Read data	RAA_RD0_B3
			RAA_RD0_B2
			RAA_RD0_B1
			RAA_RD0_B0

Remote Request		Answer	
			RAA_RD1_B3
			…
			RAA_RDx_B0

Figure 82:
RC_RAA_RDS (RAA Read in blocks with status first)

Remote Request		Answer	
Command	RC_RAA_RDS		
Address	RAA_ADR		
		Status	SYS_STATUS
		Read data	RAA_RD0_B3
			RAA_RD0_B2
			RAA_RD0_B1
			RAA_RD0_B0
			RAA_RD1_B3
			…
			RAA_RDx_B0

Figure 83:
RC_RAA_STATUS (RAA system status only)

Remote Request		Answer	
Command	RC_RAA_STATUS		
Address	RAA_ADR		
		Status	SYS_STATUS

The system status bit gives quick access to important system information with read operations.

Figure 84:
SYS_STATUS

Bit	Bit Name	Reset	Format	Bit Description
0	**RAA_BUSY**	b0	BIT	Random access area busy: Occupied system bus
1	**NOT USED**			
3:2	**MT_REQ_CTR**	b0	BIT	Measure task request counter Implemented as gray counter: 00 -> 01 -> 11 -> 10 -> 00 ->
4	**MCT_STATE**	b0	BIT	Status of measure cycle timer
5	**COM_FAIL**	b0	BIT	Communication Failed

Bit	Bit Name	Reset	Format	Bit Description
6	**RST_FLAG**	b0	BIT	Reset Flag
7	**ERR_FLAG**	b0	BIT	At least one error flag is set

Figure 85:
RC_FWC_WR (FWC Write in blocks)

Remote Request		Answer	
Command	RC_FWC_WR		
Address	FWC_ADR_B1		
	FWC_ADR_B2		
Write Data	FWC_WD0_B3		
	FWC_WD0_B2		
	FWC_WD0_B1		
	FWC_WD0_B0		
	FWC_WD1_B3		
	…		
	FWC_WDx_B0		

10.1.4 Measure Task Request

Figure 86:
Measure Task Request

Remote Command	Code	Description
RC_MT_REQ	0xDA	Measure Task Request

The Measure Task Request is followed by an extended command EC_MT_REQ, which defines the requested measure task(s):

Figure 87:
EC_MT_REQ

Extended Command	Description
EC_MT_REQ	Measure Task Request EC_MT_REQ [Bit 0]: VCC Voltage Measurement EC_MT_REQ [Bit 1]: not used EC_MT_REQ [Bit 2]: Time Of Flight Measurement EC_MT_REQ [Bit 3]: Amplitude Measurement EC_MT_REQ [Bit 4]: Amplitude Measurement Calibration EC_MT_REQ [Bit 5]: Temperature Measurement EC_MT_REQ [Bit 6]: High Speed Clock Calibration EC_MT_REQ [Bit 7]: Zero Cross Calibration

10.1.5 Debug & System Commands

Figure 88:
Debug & System Commands

Remote Command	Code	Description
RC_TSC_CLR	0x86	Time stamp counter clear
RC_BM_RLS	0x87	Bus master release
RC_BM_REQ	0x88	Bus master request
RC_RF_CLR	0x89	Reset flag clear (RST_FLAG in SYS_STATUS)
RC_MCT_OFF	0x8A	Measure cycle timer off
RC_MCT_ON	0x8B	Measure cycle timer on
RC_GPR_REQ	0x8C	General purpose request
RC_IF_CLR	0x8D	Interrupt flags clear
RC_COM_REQ	0x8E	Communication request
RC_FW_CHKSUM	0xB8	Builds checksum of all FW memories

10.2 General Purpose I/O Unit

The general-Purpose IO unit supports up to 6 GPIOs which can be used for different internal signals and/or interfaces.

The assignment of the GPIOs has to be configured by

- GPx_DIR & GPx_SEL (x = 0 to 5) in CR_GP_CTRL
- I2C_MODE in CR_IFC_CTRL

10.2.1 General Purpose Out

Each GPIO can be configured individually as an output signal of digital part as defined below:

Figure 89:
GPIO as Output, GPx_DIR = b00, I2C_MODE = b00 or b11

Pin	GPx_SEL			
	00	01	10	11
GPIO5	GPO[5]	(1)	TI_PGA_VREF	(1)
GPIO4	GPO[4]	TI_VR_EN	TI_PGA_EN	(1)
GPIO3	GPO[3]	PI_DIR	(1)	TI_FIRE_EN
GPIO2	GPO[2]	PI_PULSE	TI_FP_PCH	(1)
GPIO1	GPO[1]	PI_DIR	ERROR_N	USM_DIR
GPIO0	GPO[0]	PI_PULSE	LS_CLK	TI_FIRE

(i) For internal use only

- GPO[5:0]: General purpose outputs writable via SHR_GPO
- PI_DIR: Pulse interface direction (for more details, see section below)
- PI_PULSE: Pulse interface out (for more details, see section below)
- TI_VR_EN: Enable signal of ultrasonic transducer interface
- TI_PGA_VREF: VREF signal for PGA
- TI_PGA_EN: Enable signal for PGA
- TI_FP_PCH: Pre-Charge state of transducer interface
- ERROR_N: Error signal (low active)
- LS_CLK: Low speed clock
- TI_FIRE_EN: Busy signal of TI_FIRE
- USM_DIR: Direction signal (Down/up) of ultrasonic measurement
- TI_FIRE: Fire signal of ultrasonic measurement

10.2.2 General Purpose In

Figure 90:
GPIO as Input, GPx_DIR = b01/10/11 (in), GPx_SEL != 0, 0 I2C_MODE = b00 or b11

Pin	Function
GPIO5	GPI[5]
GPIO4	GPI[4]
GPIO3	GPI[3]
GPIO2	GPI[2]
GPIO1	GPI[1]
GPIO0	GPI[0]

- GPI[5:0]: General purpose inputs, readable via SRR_GPI

10.3 Pulse Interface

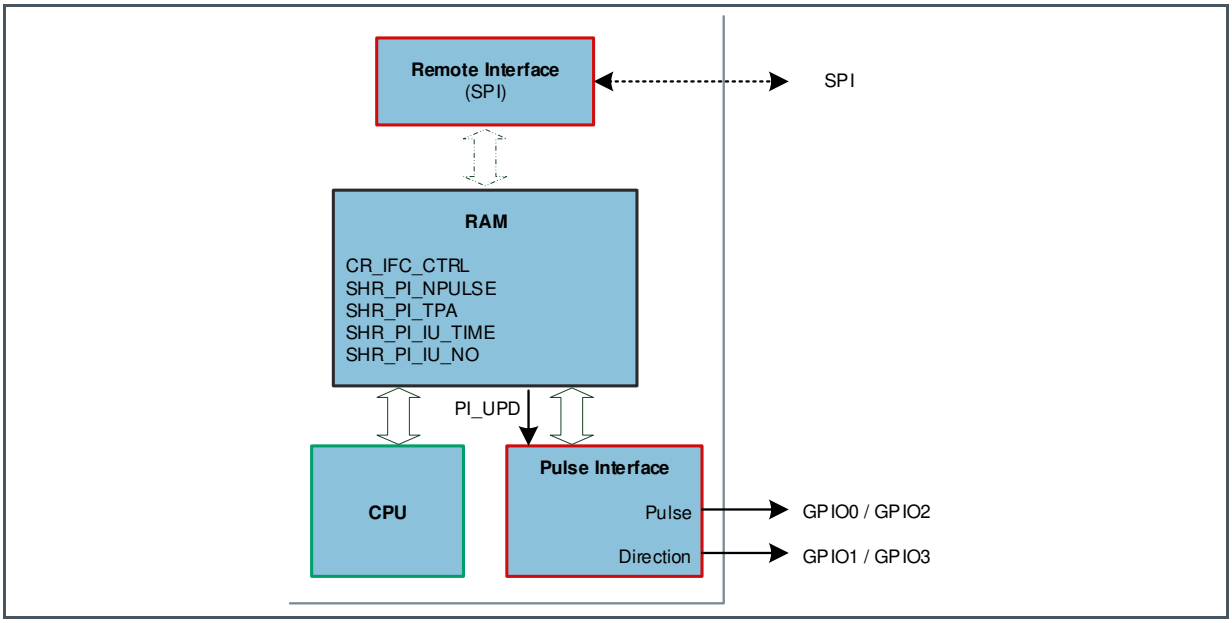
The pulse interface is a separate and independent unit connected to the GPIO block. The CPU can take any of the data, e.g. the original DIFTOF, but also the finally calculated flow information and translate this to a pulse stream. With flow as basis, this will be fully compatible to typical pulse interfaces of mechanical flow meters. Such a system might be a one-to-one replacement for a mechanical flow meter.

The pulse interface generates pulses, where each pulse corresponds to a configurable flow volume (pulse valence, for example one pulse per 100 ml). The parameters of the pulse interface are configured in register CR_IFC_CTRL. The interface then operates at the configured update rate, independent of measurement interface and CPU, by generating pulses according to the actual flow volume. The flow volume must be signaled and updated, typically by a firmware running on the CPU, by updating the register SHR_PI_NPULSE or, simpler, by using the ROM routine ROM_PI_UPD. The ROM routine calculates the necessary input variables for the pulse interface from a given flow volume.

If you plan to configure and update the pulse interface via remote interface by an external µController please contact support for details.

The following figure gives on overview of the relevant units and variables.

Figure 91:
Pulse Interface: Functional Blocks and Variables



10.3.1 Configuration of the pulse output

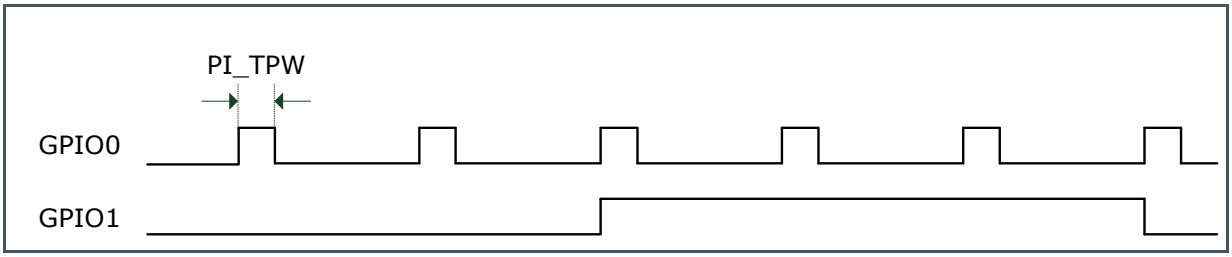
The pulse interface outputs can be provided via GPIO0/GPIO1, optionally via GPIO2/GPIO3 (register **CR_GP_CRTL**).

The basic configuration of the pulse interface is done in register CR_IFC_CTRL Register (Address 0x0C1), with the most important configuration variables having the following meaning:

PI_OUT_MODE Selects the two possible output formats.

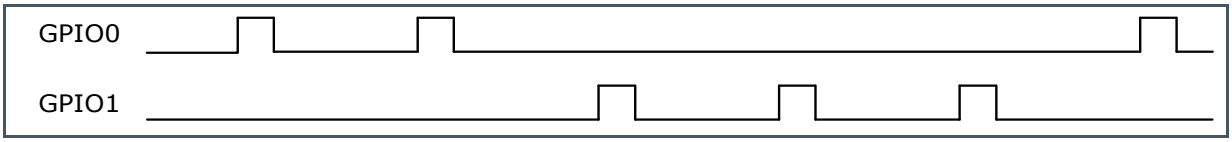
- **PI_OUT_MODE** = 0
  - GPIO0 / GPIO2 = Pulse output, provides the pulses, indicating flow
  - GPIO1 / GPIO3 = Direction output, provides the direction of the measured flow rate, indicating positive/negative flow

Figure 92:
PI_OUT_MODE = 0



- **PI_OUT_MODE** = 1
  - GPIO0 / GPIO2 = Pulse output, positive direction, issued for flow direction forward
  - GPIO1 / GPIO3 = Pulse output, negative direction, issued for flow direction reverse

Figure 93:
PI_OUT_MODE = 1



- **PI_TPW**: Pulse width in multiples of 0.97656 ms (= period of 1024 Hz generated by 32.768 kHz clock), configurable from 1 to 255 (0.97656 ms to 249 ms).
- **PI_OUT_MODE** and **PI_TPW** are initial parameters, which are typically configured once.

The general FW library of UFC provide subroutines for pulse interface initialization, dependent on following application parameters:

- TOF measure cycle time

- Pulse valence (ratio pulses/liter)
- Maximum flow

10.4 2-wire Master Interface

The 2-wire master interface for an external memory extension (e.g. for backup purpose) or an external pressure sensor is a separate, independent unit, connected to the GPIO unit, which can be controlled by firmware of the integrated CPU. It is a master interface, suited for a single two-wire connection to an I^2C compatible device. It works in standard mode up to 100 kHz or in fast mode up to 400 kHz (but no Schmitt-trigger inputs). It supports spike suppression on the SDA input. Clock stretching is not supported.

SCL: Serial clock line
SDA: Serial data line (bidirectional)

The assignment of the signal lines to GPIOs can be configured by **I2C_MODE** in **CR_IFC_CTRL** as follows:

Figure 94:
I2C Modes

I2C_MODE	00	01	10	11
GPIO0	(1)	SCL	(1)	(2)
GPIO1	(1)	SDA	(1)	(2)
GPIO2	(1)	(1)	SCL	(2)
GPIO3	(1)	(1)	SDA	(2)

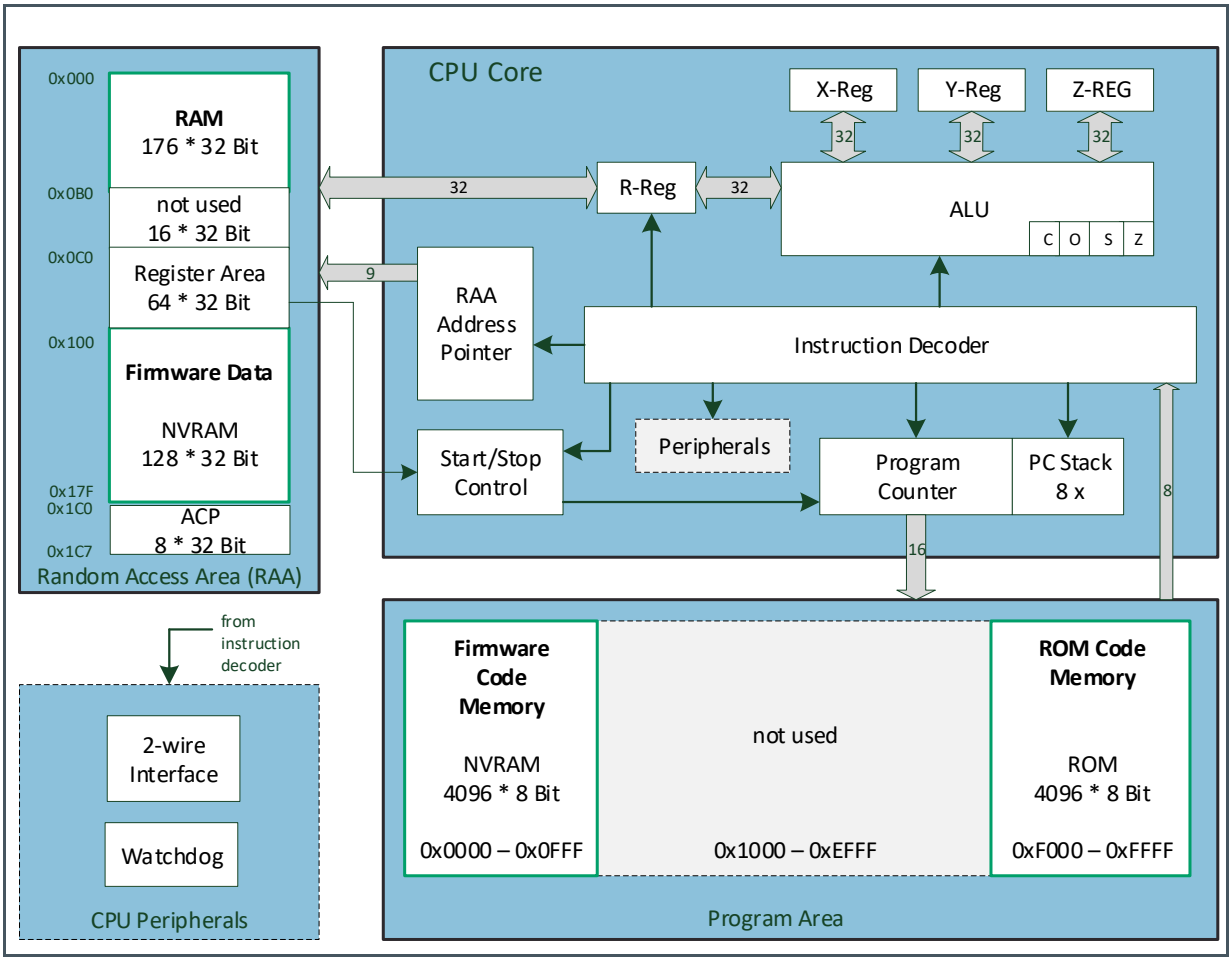
(1) As configured by GPx_DIR & GPx_SEL (table at beginning of this chapter)
(2) Not allowed

The general FW library of UFC provide subroutines for 2-wire interface communication.

11 CPU

The CPU is structured and implemented as shown in the following figure. It has access to the full RAM, including the result registers and the status registers.

Figure 95:
CPU Environment



11.1 Registers and Accumulators

The 32 bit-CPU operates on three internal registers, the X, Y, and Z-accumulators, and on one register or RAM cell, addressed by the CPU's RAM address pointer. The latter register is denoted with R, it can be any accessible cell within the RAA address range. R is handled in the same way as an accumulator by most commands. One specialty of R is the byte coding and decoding by the bytesel and bytedir command, which only acts on R in read direction (details see below). This function is built in for simplified and accelerated byte operations.

11.2 CPU Flags

The CPU uses four flags to classify the results of operations: Carry (C), equal Zero (Z), Sign (S) and Overflow (O). Zero and Sign flags are set with each CPU write access to any register, RAM or accumulator. Additionally, the Carry and Overflow flags are set in case of a calculation, shift or rotation. Flags which are not actively changed by an operation remain in their former state. It is possible to query each flag in a jump or skip instruction.

11.2.1 Carry (C)

Shows the carry over in an addition or subtraction. Note that the carry flag is calculated assuming unsigned binary numbers, in contrast to the overflow flag. Thus, it may produce confusing results, refer to the detail description of instructions for usage. With shift operations (shiftL, shiftR, rotL, shiftR.), the carry flag is set to the (last) bit that has been shifted out.

11.2.2 Overflow (O)

Indicates an overflow during an addition or subtraction of two numbers in two‘s complement representation. This is strictly an overflow for positive numbers, underflow in case of negative numbers is not indicated. If the eventuality of a negative underflow can’t be avoided, additional calculations to indicate the underflow are required.

11.2.3 Zero (Z)

The zero flag indicates if the last number written into a register (by add, sub, move, swap, etc.) was zero or not equal to zero.

11.2.4 Sign (S)

The sign flag indicates if the last number written into a register (by add, sub, move, swap, etc.) has the highest bit (MSB) set to 1 or to 0. It thus indicates the sign of this number, with zero indicated positive. The sign flag assumes a two’s complement number representation.

11.3 Arithmetic Operations

An arithmetic command processes two of the registers X, Y, Z or R, and writes back the result into the first mentioned register (or, for commands with 64-bit results, into both). These operations also affect flags of the CPU. In particular, the carry (C) and overflow (O) flags should be checked to ensure correctness of the last operation.

All arithmetic operations process a 32-bit wide input, (mostly) based on the common two’s complement operations. This means that the MSB (the most significant bit of the binary word, here bit

31) defines the sign of the binary number, with negative signs having MSB=1. Number values of positive numbers are as usual, while the value of a negative number A follows the rule |A| = NOT(A) +1 , in words: negative numbers are converted into positives by bitwise inversion, and then adding 1 (see the instructions “compl” and “invert”)

11.3.1 Branch Instructions

There are 3 principles of jumping within the code:

- Goto: Jumps with relative or absolute addressing. Within an address vicinity of –128 to +127, the assembler automatically uses relative addressing (“Branch”). For wide distances, absolute addressing within the whole address space of 64 kB is automatically used (“Jump”). The latter is more flexible but needs one code byte more.
- Jsub: Absolute or relative jump, used to call a subroutine. The difference to goto is that the code returns to the calling address at jsubret (for example at the end of the subroutine). It is possible to handle 7 nested jsub. An overflow of the stack counter is indicated in CPU_ERR. When no return to the calling address is desired, it is better (and of course possible) to use goto instead of jsub.
- Skip: Suppress the execution of the next 1, 2 or 3 instructions. Note that the skipped instructions are in fact processed, but they produce no result or further activity. Thus, skip does not save processing time of the skipped instructions, in contrast to goto or jsub. However, the skip command itself is only one byte short, and in addition it is highly suitable for structured programming.

Goto and skip come in different flavors, as unconditional command as well as controlled by some bit or CPU flag. Refer to the detail instruction list below for details.

11.4 Instruction Set

The complete instruction set of the AS6031 consists of 70 core instructions that have unique op-codes decoded by the CPU. The following table gives an overview of all available expressions, details are given further below.

Figure 96:
Instruction Set Overview

Logic	Simple arithmetic	Complex arithmetic	Flags	Register-wise	Jsub
and	abs	div	clrC	clear	jsub
eor	add	divmod	getflag	move	jsubret
eorn	compare	mult	setC	swap	
invert	compl				
nand	decr				
nor	incr				

Logic	Simple arithmetic	Complex arithmetic	Flags	Register-wise	Jsub
or	sign				
Logic	**Simple arithmetic**	**Complex arithmetic**	**Flags**	**Register-wise**	**Jsub**
	sub				
RAM access	**Jump**	**Skip**	**Miscellaneous**	**Shift & Rotate**	**Bitwise**
bytedir	goto	skip	clkmode	rotL	bitclr
bytesel	gotoBitC	skipBitC	clrwdt	rotR	bitinv
decramadr	gotoBitS	skipBitS	equal	shiftL	bitset
getramadr	gotoCarC	skipCarC	equal1	shiftR	
incramadr	gotoCarS	skipCarS	i2crw		
ramadr	gotoEQ	skipEQ	mcten		
	gotoNE	skipNE	nop		
	gotoNeg	skipNeg	stop		
	gotoOvrC	skipOvrC			
	gotoOvrS	skipOvrS			
	gotoPos	skipPos			

For a detailed description of the individual instructions see appendix 15.4 CPU Commands.

11.5 Libraries and pre-defined routines

AS6031comes with a number of predefined routines in its ROM. Some of them are ready-to-use and freely available. The ROM routines are organized in a library, defined by a so called header file which relates routine and variable names to their call addresses and memory addresses, respectively:

- common.h General purpose routines

File “common.h” that comes with the assembler must be included in codes that use any of these routines (use the “include” statement in the main *.asm file). The routines are called using their ROM routine name after jsub or any goto statement. The ROM routine name is a synonym of the call address, as defined in the header file. The call address may be used alternatively.

Some routines come in different alternative versions or with alternative start addresses. To some extent, this allows the user to select different RAM cells for data storage. A typical example would be a routine which needs some cells of usual RAM, and an alternative version where cells in the firmware data (FWD) range are used instead – this second one frees up RAM space and could make use of automated non-volatile storage, at the cost of firmware data space. Another reason for alternative start addresses is to skip a part of the routine if some part of the preparation work is not needed or undesired (for example when some numbers calculated at the start of the routine are already known). The differences between the versions are explained for each routine in detail in the subsequent sections.

Number format: As usual in fixed decimal-point arithmetic, care has to be taken to set values in the right format. Unless differently noted, all numbers are in two's complement (MSB determines sign). The binary representation B_bin of a fractional number is defined with a fixed number N of fractional binary digits, such that the corresponding decimal number B_dec is calculated as: B_dec = Bin_into_decimal(B_bin)/2^N. Throughout this document, such a format will be labeled "fd N" – N fractional digits. A typical value format is fd 16, covering a fractional number range from about -32768.0 to 32768.0 (when using 32 bit RAM cells).

The second factor to be considered in calculations is the unit, which in many cases comes with a fixed factor, for example whenever values are related to a particular physical value. A typical example is measured TOF time, which is always given as fd 16 in HSC periods (250 ns for 4 MHz operation). This means, the measured Time-of-flight value in time units TOF relates to the measured number TOF_bin as: TOF = (Bin_into_decimal(TOF_bin)/2^16) * 250 ns. Another example is the first hit level FHL, which is given as an integer binary number FHL_bin with an LSB of about 0.88mV: FHL = Bin_into_decimal(FHL_bin) * 0.88 mV.

Due to the internal calculation processes, the range of values which generate correct results in some calculation is limited and depends on the format definition. For example, a multiplication of two 32 bit numbers always generates a correct 2*32 bit result (in two words, Y and X register). But if this result is formatted into one single word in fd 16 format (for example using ROM_FORMAT_64_to_32BIT) for further calculations, the result can only be right when the leading 16 bit of the original result where 0 (and of course, some accuracy is lost by cutting the lowest 16 bit, too). Such effects have to be considered in any routine that deals with actual calculations. Wherever applicable, number formats and additional range limitations are given in the subsequent routine descriptions.

11.5.1 common.h

The general purpose routines defined in common.h are listed in the following. Note that not all of them can be used in the same code, depending on memory allocation. Some routines are included in alternative versions, to enable optimized memory usage. In the following sections, the ROM routines defined in common.h are grouped according to their usage. The table gives an overview:

Figure 97:
ROM-routines for common usage

Name	Description	Remarks
Filtering		
ROM_INIT_FILTER ***ROM_INIT_FILTER1***	Routine to initialize the RAM cells for any filter (rolling average) with a given value	
ROM_ROLL_AVG	Routine to filter the ***FILTER_IN*** values using a rolling average filter	Filter length can be configured
ROM_ROLLAVG_2OUTLIER	Routine to filter the ***FILTER_IN*** values using a rolling average filter. One value which deviates most is always ignored.	Filter length can be configured
ROM_FILTER_FLOW	Routine to filter flow values using the standard rolling average filter, including initialization.	Filter length can be configured
Error detection and handling		

Name	Description	Remarks
ROM_EH	This routine checks all error flags and suppresses processing of wrong results.	many RAM cells fixed
ROM_PP_AM_MON ***ROM_PP1_AM_MON***	Monitor the amplitude values and check limits to identify bad measurements	alternative calls exist
ROM_PP_AM_CALIB ***ROM_PP1_AM_CALIB***	This routine gets the Amplitude Calibration values (H & L) and evaluates the gradient and offset that can be used for calculating the actual amplitude.	alternative calls exist
Pulse interface and flow volume		
ROM_CFG_PULSE_IF	This routine configures the pulse interface with the parameters calculated from the given configuration.	
ROM_PI_UPD	Pulse Interface Update Routine	
ROM_PP_PI_UPD	Pulse Interface Update Routine with input from RAM	
ROM_SAVE_FLOW_VOLUME ***ROM_SAVE01_FLOW_VOLUME*** ***ROM_SAVE1_FLOW_VOLUME*** ***ROM_SAVE11_FLOW_VOLUME*** ***ROM_SAVE2_FLOW_VOLUME*** ***ROM_SAVE21_FLOW_VOLUME***	This routine is used to store the converted flow (in LPH), cumulatively to flow volume in cubic meter.	alternative versions exist
Sensor temperature measurement		
ROM_TEMP_POLYNOM	Calculates the temperature of a PT sensor using a polynomial approximation	
ROM_TEMP_LINEAR_FN	This routine is used to calculate the temperature of any sensor as a linear function of sensor resistance using the nominal resistance and sensor slope.	
ROM_TM_SUM_RESULT	Sums up the results of double temperature measurements. The double measurements are performed to eliminate the 50/60 Hz disturbance.	
Interface communication		
ROM_I2C_ST	I2C Start Byte Transfer	Low-level routines, covered by the ones following
ROM_I2C_BT	I2C Byte Transfer	
ROM_I2C_LT	I2C Last Byte Transfer	
ROM_I2C_DWORD_WR	Write 4 bytes of data to a specified address through the I2C interface	
ROM_I2C_BYTE_WR	Write a single byte of data to a specified address through the I2C interface	
ROM_I2C_DWORD_RD	Sequentially read 4 data bytes from the I2C interface	
ROM_I2C_BYTE_RD	Sequentially read a single data byte from the I2C interface	
Housekeeping		
ROM_CPU_CHK	Check kind of CPU request: This routine is called by hardware design after any Post Processing (PP) request, it is the starting point of any CPU activity, including the firmware call at **MK_CPU_REQ**.	automatically started
ROM_USER_RAM_INIT	Initialize the entire user RAM with 0	
High speed oscillator		
ROM_HSC_CALIB	This routine evaluates the high speed clock scaling factor for the 4MHz / 8 MHz clock	
ROM_SCALE_WITH_HSC	Routine to scale the input parameter with the HS Clock Calibration factor	
Configuration		

Name	Description	Remarks
ROM_RECFG_TOF_RATE	Routine to reconfigure TOF_RATE generator to a lower rate, depending on the parameter N	
Mathematics		
ROM_FORMAT1_64_TO_32BIT	Routine to format a 64-bit value (in Y and X) into a 32 bit result with 16 integer + 16 fractional bits. Useful for formatting 64 bit multiplication results with 32 integer + 32 fractional bits	alternative version: faster, but needs temporary RAM
ROM_DIV_BY_SHIFT	Perform the division of a value Y by X, where X=2^N is an integer power of two	
ROM_SQRT	Evaluate the square root accurately for values in the range (196 <= X <= 5476)	
ROM_LINEAR_CORRECTION ***ROM_LINEAR1_CORRECTION***	Linear interpolation of a coefficient between two sampling points	alternative version is fixed to inter-polation over THETA
ROM_FIND_SLOPE	Used to find the slope between two points, given the coefficient values and parameter values at the two points.	

For a detailed description of the ROM routines refer to the appendix.

11.6 CPU Handling

The CPU starts with handling a request as soon as one of the bits in the system handling register **SHR_CPU_REQ** is set. All bits are typically triggered by the task sequencer, the error handling, a general-purpose pin or the remote control.

Post processing is enabled in register **CR_MRG_TS**, bits **TS_PP_F_EN** and **TS_PP_T_EN**. Error handling and general purpose handling is enabled in register **CR_IEH**, bit **CPU_REQ_EN_GPH**.

- CPU_REQ_BLD_EXC: Bootloader, triggered by task sequencer
- CPU_REQ_CHKSUM: Checksum Generation, triggered by task sequencer
- CPU_REQ_PP_F: Post Processing F, triggered by task sequencer
- CPU_REQ_PP_T: Post Processing T, triggered by task sequencer
- CPU_REQ_GPH: General Purpose Handling, triggered by task sequencer
- CPU_REQ_FW_INIT: Firmware Initialization, triggered by bootloader, cleared when CPU stops

In general, sending any of these requests by task sequencer the corresponding bit in **SHR_CPU_REQ** lets the chip start the CPU at the appropriate position within the task sequencer cycle. CPU operation then always starts within the ROM code by checking the request.

Bootloader and checksum generation requests are handled directly in ROM routines.

For Firmware initialization, post processing and General purpose handling the program counter is directed to firmware code memory, starting at address

- 0x0024: With applied Empty FW

- FWCU_RNG: With applied Flow FW

After the request is processed, the firmware must clear it in **SHR_CPU_REQ** (or simply clear the whole register after all is done), else the request remains.

The following figures show the basic structure of code. Program code in white color has to be defined and programmed by customer, whereby public subroutines in the ROM code can also be used by customer.

Figure 98:
Code Structure, Empty Firmware

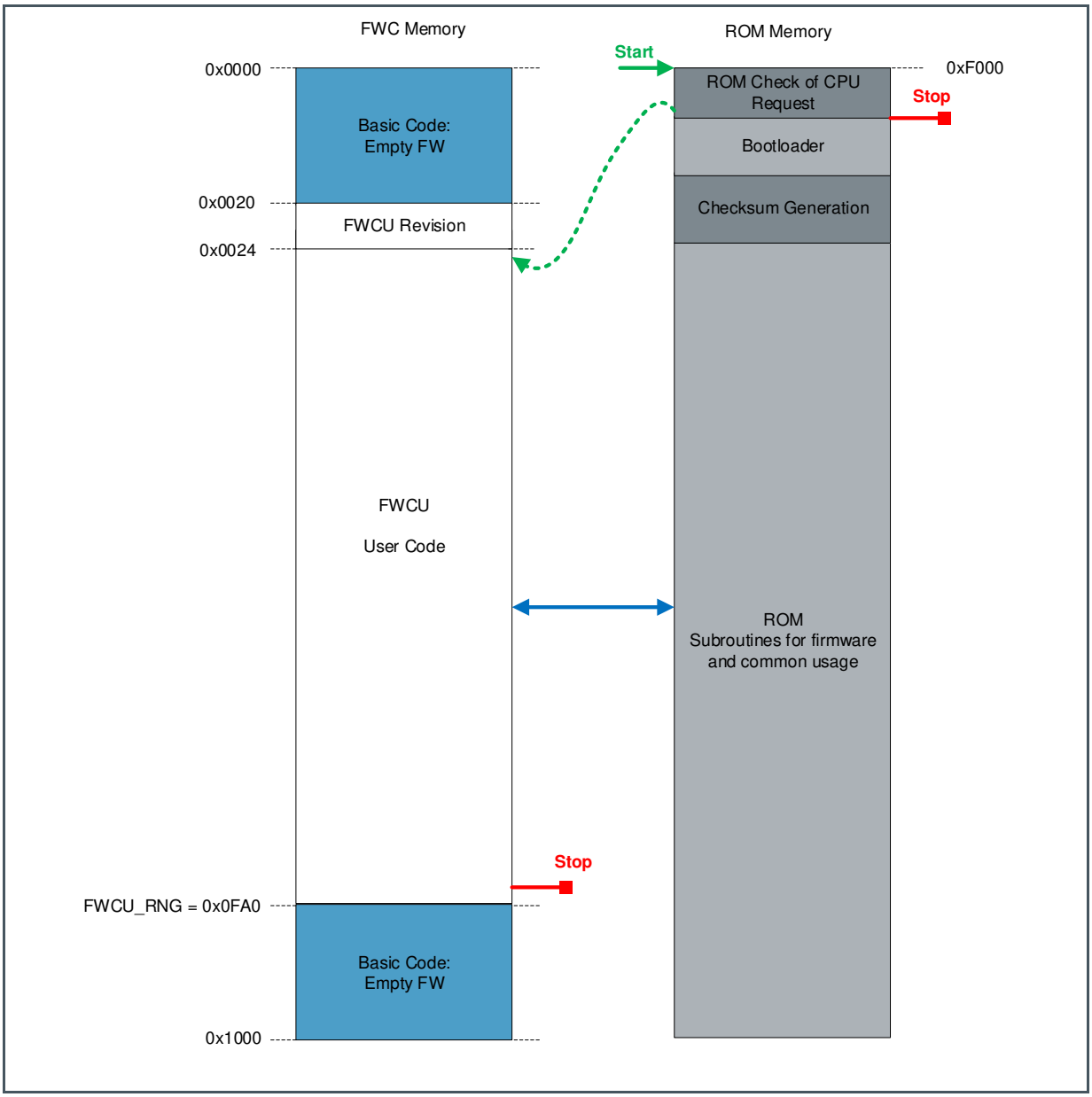
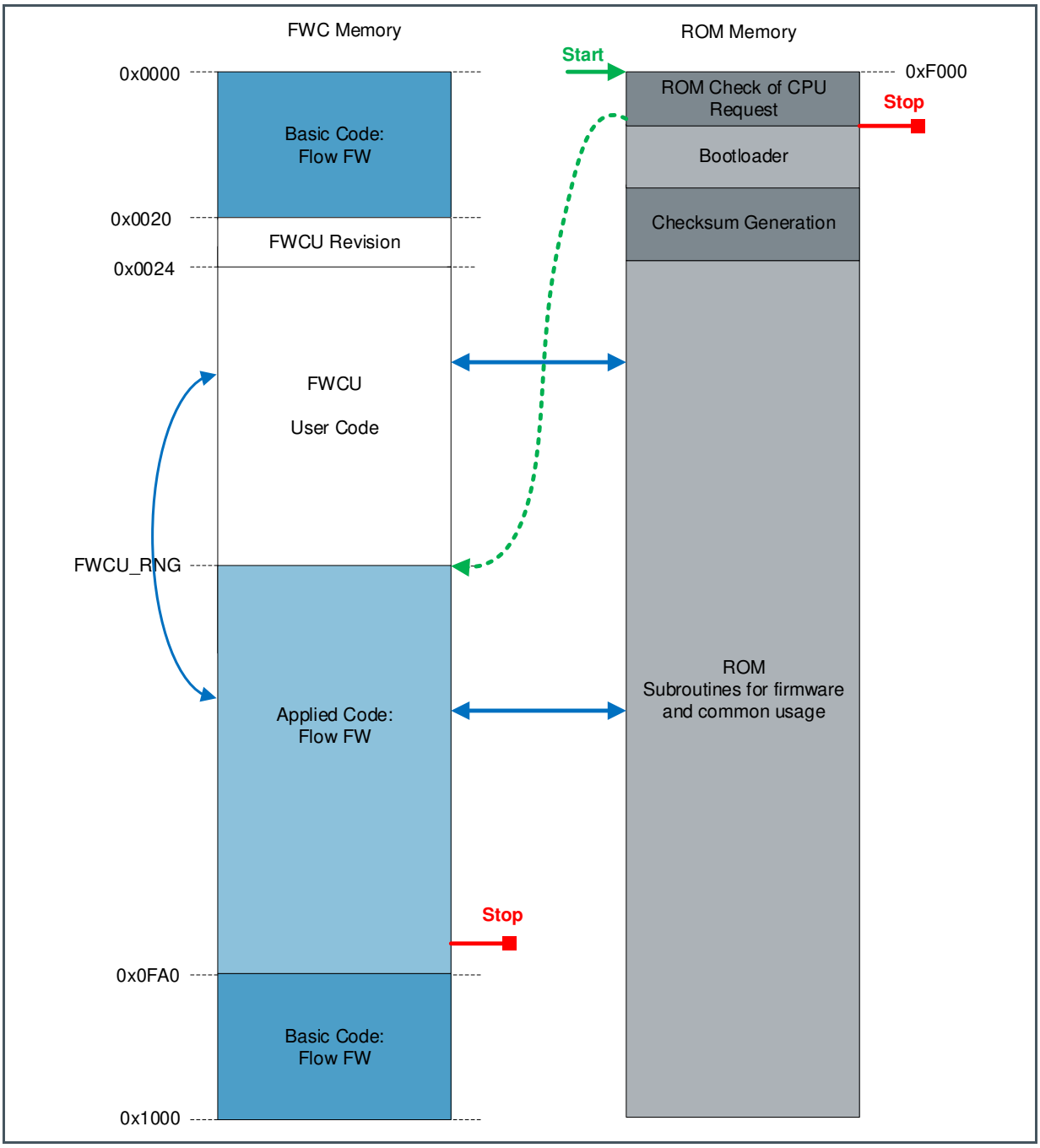


Figure 99:
Code Structure, Flow Firmware

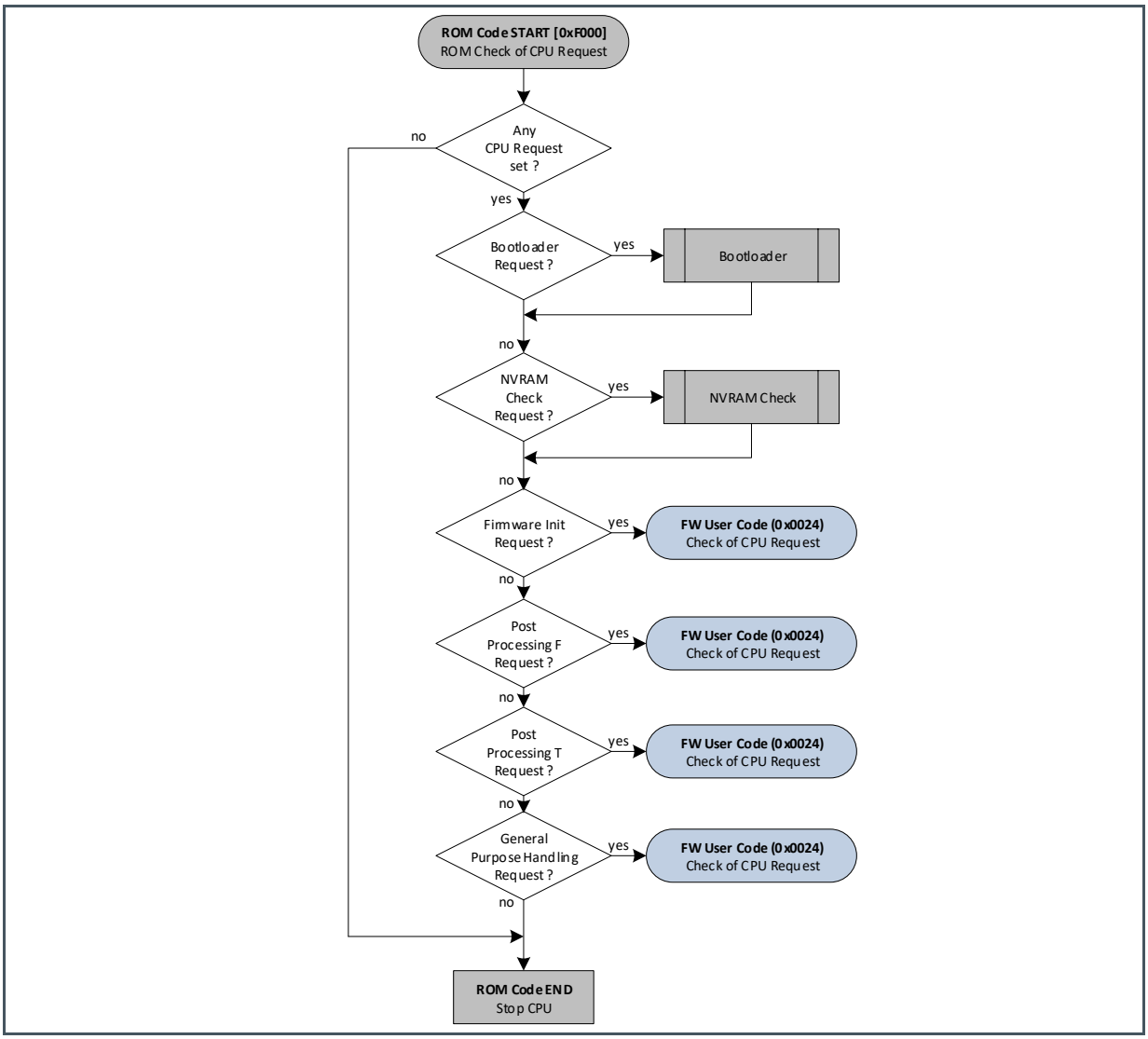


The device will be offered as an option with integrated firmware, that does the complete flow calculation, but still offers the possibility to add custom made code. After a ROM check of the CPU request, the program counter is directed via basic code to the address of FWCU_RNG from where applied flow FW is executed.

11.6.1 Check CPU Request

In case that any of the request bits is set in **SHR_CPU_REQ** the CPU starts at first with code in the ROM that checks the type of request.

Figure 100:
CPU Request Handling(1)



(1) With empty firmware

In case of a post processing request, a general purpose request or a firmware initialization request the CPU is directed into firmware user code, starting from address 0x0024. This means that the user has to implement in his firmware also a CPU request check.

- Firmware initialization (FW code): Besides the configuration done by the boatloader some additional configurations can be performed, which typically are some initializations of the SHR register.
- Post Processing F (FW code): This will be the most common request, namely for data post processing of flow in case that flow and temperature measurement is enabled.
- Post Processing T (FW code): This will be the most common request, namely for data post processing like flow (if temperature measurement is not enabled) or temperature calculation.
- General purpose request handling (FW code)
  General Purpose Request can be triggered either triggered by GP Timer, by SHR_EXC or by CPU_REQ_GPH. For any of these actions CPU_REQ_EN_GPH has set before.

In addition, CPU Request Handling processes two further requests which are performed in ROM code and are described in following sections:

- Bootloader (ROM code)
- NVRAM check (ROM code)

11.6.2 Bootloader

The bootloader is always requested after any system reset or a system INIT occurred. However, complete bootloader actions are only performed if the autoconfiguration release code is set.

The Register Configuration is performed if Autoconfig Release Code (RAA address 0x16B) is set.

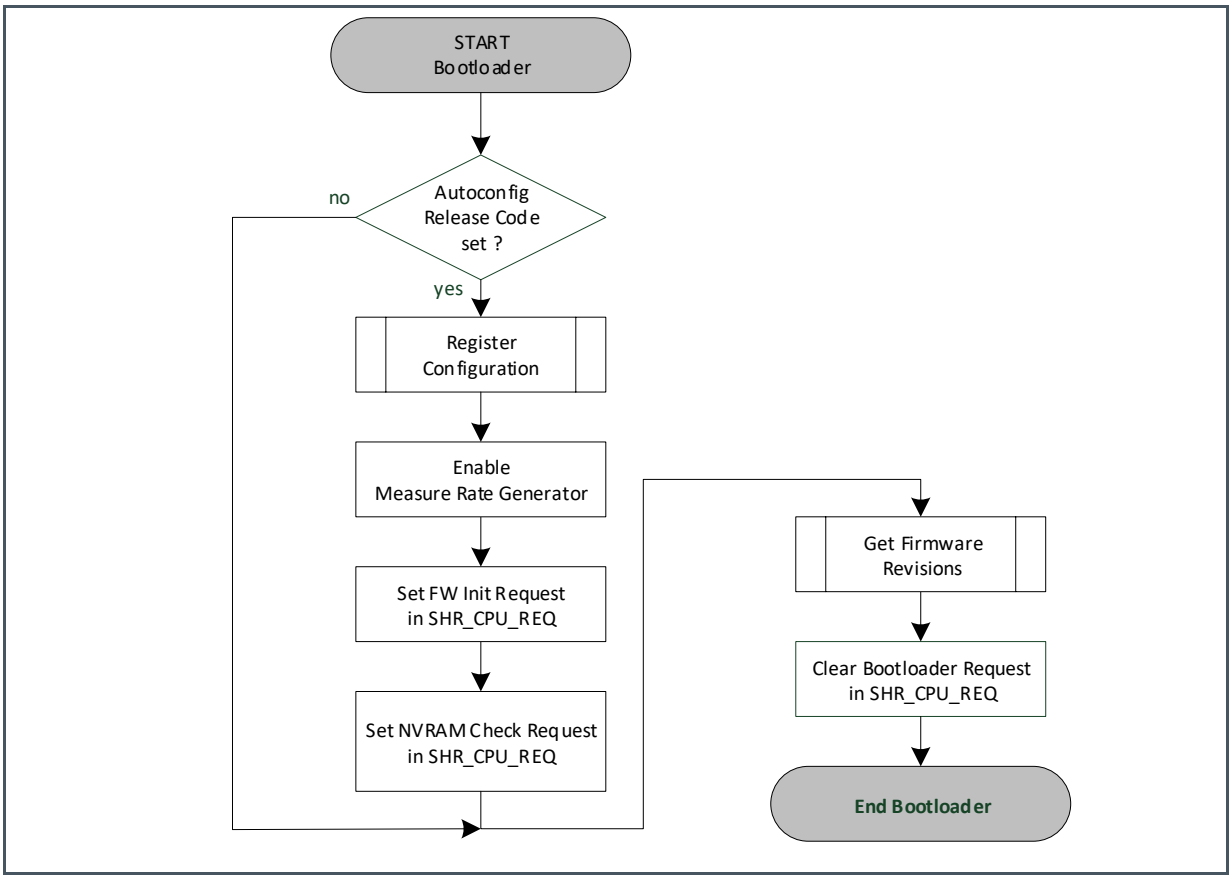
It copies the user configuration data and the ScioSense configuration data (RAA addresses 0x16C-017B) into CR register area (0x0C0-0x0CF). Also, the FHL is copied to SHR_FHL_U and SHR_FHL_D.

Bootloader actions are:

- “Get Firmware Revisions” which set FW revisions in register **SRR_FWU_REV** & **SRR_FWA_REV**
- Check whether the release code for autoconfiguration is set (0x16B). If yes then
  - Transfer of configuration data to register area
  - Enabling Measure Rate Generator
  - Setting “FW Init” request in **SHR_CPU_REQ** which is performed after bootloader sequence has been finished
  - Setting NVRAM check request in **SHR_CPU_REQ**
- Finally, the bootloader clears the bootloader request in **SHR_CPU_REQ** and jumps back to ROM code for checking CPU requests.

The bootloader does not set USM_RLS_DLY. This needs to be set manually or per initialization in the firmware.

Figure 101:
Bootloader actions



11.6.3 NVRAM Check

NVRAM check can be requested by remote command RC_FW_CHKSUM, by the checksum timer or, if autoconfiguration release code is set, it's automatically performed after bootloader.

Then the checksums of all FW areas are generated and compared to checksums which can be stored to FW Data memory.

Then different FW areas are checked:

- FWCU: compared to checksum stored on address 0x100
- FWDU: compared to checksum stored on address 0x101
- Applied FW: Different areas compared to applied checksums internally

Finally, the checksum generation clears its request in **SHR_CPU_REQ** and jumps back to ROM code for checking CPU requests. The checksum for FWCU is generated by adding read data bytewise. The checksums for FWDU is generated by reading DWORDs (32 bit) and adding read data bytewise to checksum.

Figure 102:
Relevant Registers for NVRAM Check

Register	Parameter	Description
CR_IEH (Interrupt & Error Handling)	**EF_EN_XX_XX_XX**	Bits 12 to 14 enable error flags corresponding to register **SRR_ERR_FLAG** 12: FWCU check failed 13: FWDU check failed 13: Applied FW check failed
SHR_GPO (General Purpose Out)	**FW_XX_CS_ERR**	For more details on applied FW 12: FWCU checksum error 13: FWDU checksum error 14 to 18 Any of applied FW checksum errors
CR_MRG_TS (Measure Rate Generator & Task Sequencer)	**TS_CST_RATE**	Firmware Check(sum) Timer Rate 000: disabled 001: 1h 010: 2h 011: 6h 100: 24h 101: 48h 110: 96h 111: 168h

Under worst case conditions, the checksum generation can consume up to 8.6 ms.

This should be considered when invoking this task timer based during measure cycle: The FW check(sum) task has to be integrated with all other measure tasks within 1 measure cycle.

11.6.4 CPU Error

As part of the error handling the CPU Error Flag indicates following error cases of CPU:

- Invalid program counter
- Stack overflow of program counter

Figure 103:
Relevant Registers for CPU Error

Register	Parameter	Description
CR_IEH (Interrupt & Error Handling)	**EF_EN_CPU_ERR**	Bits 15 enables error flags corresponding to register **SRR_ERR_FLAG** 15: CPU Error

11.7 Assembler

The AS6031 assembler is a multi-pass assembler that translates assembly language files into HEX files as they will be downloaded into the device. For convenience, the assembler can include header

files. The user can write his own header files but also integrate the library files as they are provided by ScioSense. The assembly program is made of many statements which contain instructions and directives. The instructions have been explained in the former section 3 of this datasheet. In the following sections we describe the directives and some sample code.

Each line of the assembly program can contain only one directive or instruction statement. Statements must be contained in exactly one line.

Symbols

A symbol is a name that represents a value. Symbols are composed of up to 31 characters from the following list:

A - Z, a - z, 0 - 9, _

Symbols are not allowed to start with numbers. The assembler is case sensitive, so care has to be taken for this.

Numbers

Numbers can be specified in hexadecimal or decimal. Decimals have no additional specifier. Hexadecimals are specified by leading “0x”.

Expressions and Operators

An expression is a combination of symbols, numbers and operators. Expressions are evaluated at assembly time and can be used to calculate values that otherwise would be difficult to be determined.

The following operators are available with the given precedence:

Figure 104:
Assembler Operators

Level	Operator	Description
1	()	Brackets, specify order of execution
2	* /	Multiplication, Division
3	+ —	Addition, Subtraction

Example:

```
const       value  1

equal       ((value + 2)/3)
```

Directives

The assembler directives define the way the assembly language instructions are processed. They also provide the possibility to define constants, to reserve memory space and to control the placement of the code. Directives do not produce executable code.

The following table provides an overview of the assembler directives.

Figure 105:
Useful Caption

Directive	Description	Example
CONST	Constant definition, CONST [name] [value] value might be a number, a constant, a sum of both	CONST REV_ADDRESS 3964 CONST FW_VER + 2
LABEL:	Label for target address of jump instructions. Labels end with a colon. All rules that apply to symbol names also apply to labels.	jsub BLD_CFG; BLD_CFG: move y,16;
;	Comment, lines of text that might be implemented to explain the code. It begins with a semicolon character. The semicolon and all subsequent characters in this line will be ignored by the assembler. A comment can appear on a line itself or follow an instruction.	; Call Address: XXX
org	Sets a new origin in program memory for subsequent statements.	org 0
equal	Insert three bytes of user defined data in program memory, starting at the address as defined by org.	equal 0xcfcf01
#include	Include the header or library file named in the quotation marks "". The code will be added at the line of the include command. In quotation marks there might be just the file name in case it is in the same folder as the program, but also the complete path.	#include "common.h"

11.7.1 Basic Structure

The following flow chart shows the basic structure of a AS6031 firmware.

Figure 106:
Basic Program Flow Chart

t.b.d.

12 Memory and Register Description

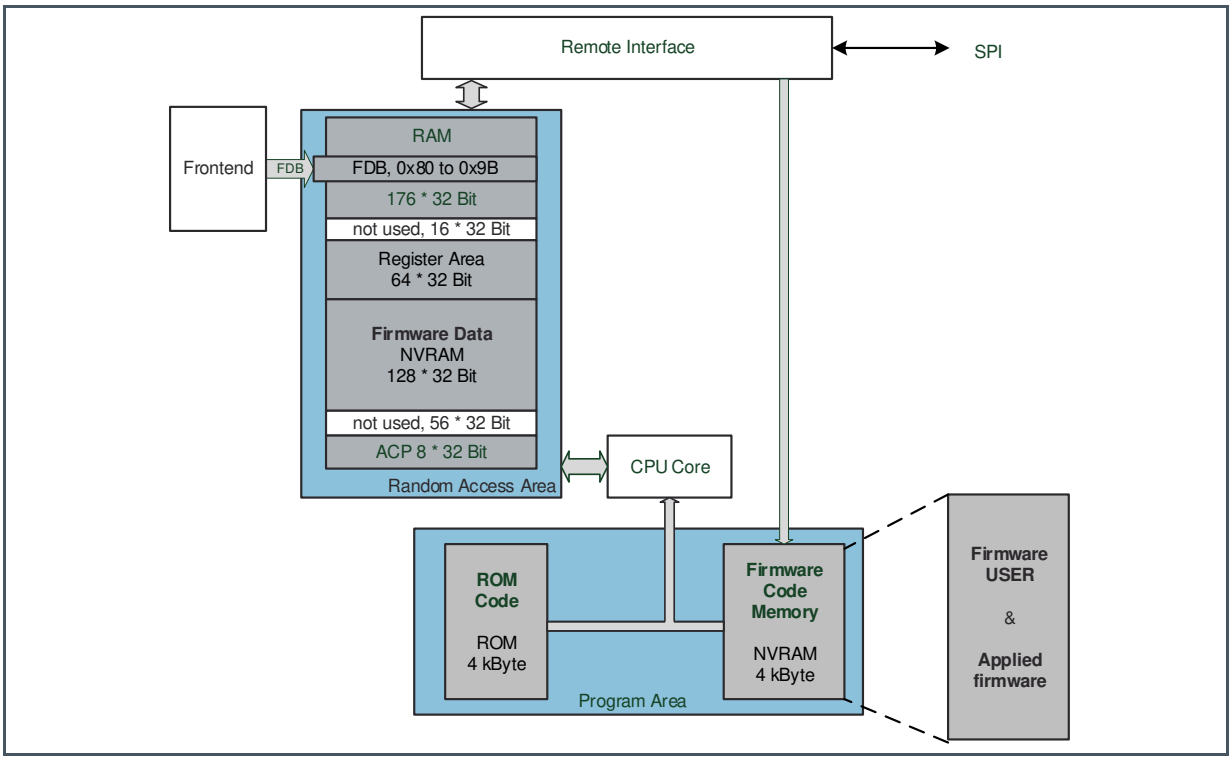
The AS6031 has two operation modes with different usage of the memory.

- Time conversion mode: In this mode, the CPU is not active. The device runs as a pure front-end providing the raw time measurement results for flow, temperature and amplitude. The only relevant section of the memory is the random access area (RAA) with the configuration registers, system handling registers, result register and status registers.
- Flow meter mode: In this mode, The CPU is active and does post-processing on the raw data, typically calculating temperature, flow and volume, accompanied by error handling. The program code is stored in NVRAM, in addition to ROM for ready supporting functions. Customers may write their own code or use the applied firmware of AS6031F1. In any case there is a small mandatory applied firmware section. There is a separate section in the NVRAM to store firmware-specific data as well as configuration data. The CPU uses the full 176 × 32 bit RAM to read measurement results, to do its calculations and to write the final results. ROM and firmware code memory share a different address bus system and are not readable from outside the chip.

The firmware code memory and the firmware data memory are zero static power NVRAMs. Since they don’t draw current when not in use, they are not switched down and remain permanently usable. However, the address and data bus of the RAA can only be allocated to one system at a time, so access to RAA memory cells from outside is usually not possible when the frontend or the CPU operate on it.

The following diagram shows the memory organization and the interaction of the frontend, the CPU and the remote interface.

Figure 107:
Memory Organization



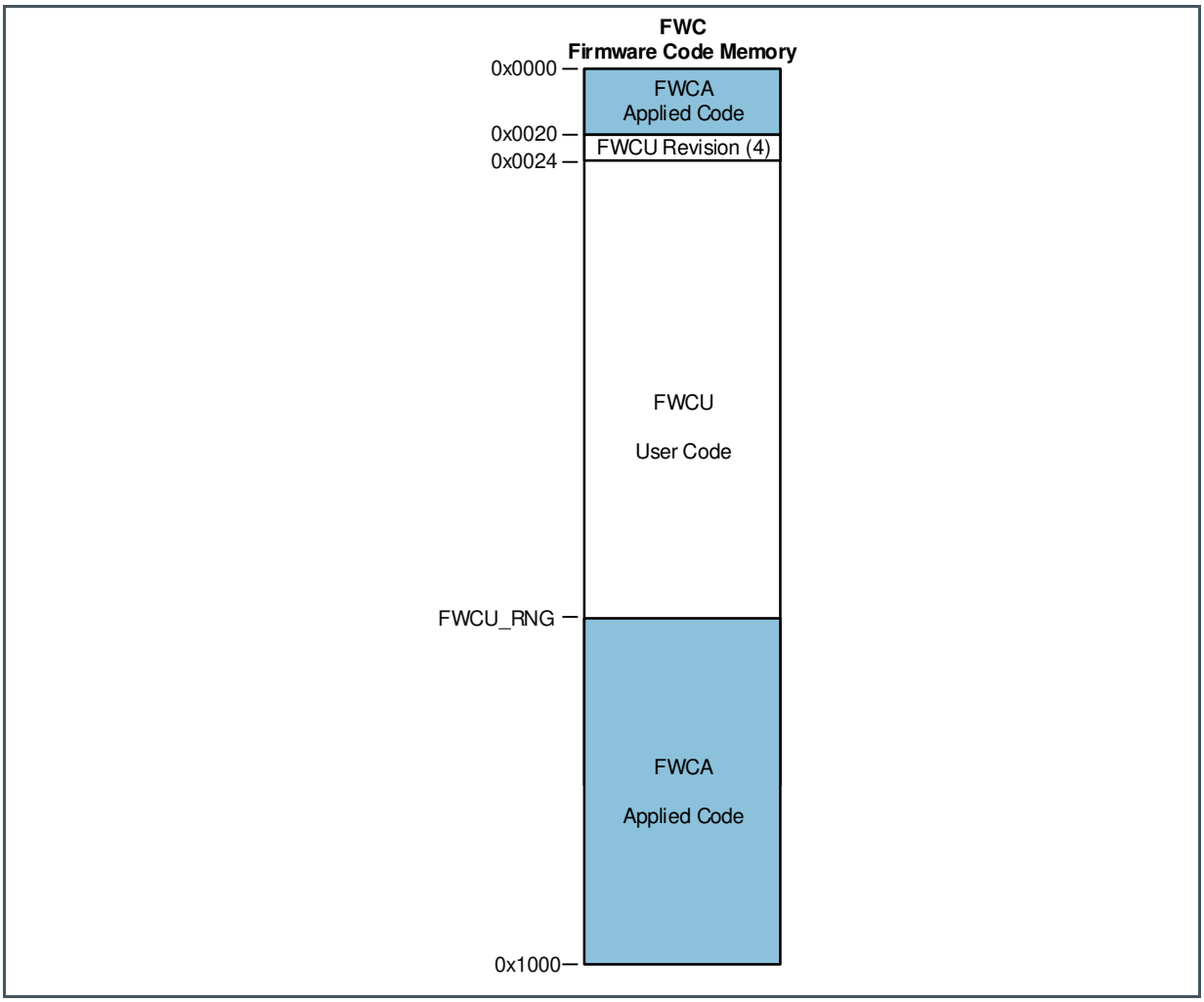
12.1 Program Area

The program area consists of two memory parts: A 4 kB NVRAM for re-programmable program code, and a 4 kB ROM with read-only program code.

The firmware code in re-programmable NVRAM memory consists of:

- A USER part which can be programmed by customer
- A divided **ScioSense** part, pre-programmed by **ScioSense** including general subroutines addressable by customer.

Figure 108:
Program Area



The available size of USER Firmware (FWU) is defined in register **SRR_FWU_RNG.** The user can read this. In addition, the USER firmware has a reserved area of 4 byte at the beginning of the code memory, which can be used to implement a revision number. The revision can be read via register **SRR_FWU_REV**. The revision of applied firmware (FWA) can be read via **SRR_FWA_REV**. Note that these two registers get updated by the bootloader, which is run after each POR, system reset or system init. The bootloader updates the SRR_FWU_RNG as well as the revision registers.

The firmware code in read-only ROM memory includes system subroutines (bootloader, checksum generation) and general subroutines which are also addressable by users. It further handles an initial check of CPU requests set in **SHR_CPU_REQ** register

12.2 Random Access Area (RAA)

The random access area can be separated into 4 sections:

- Random access memory (RAM) storing volatile firmware data and including frontend data buffer
- Register area
- Non-volatile RAM (NVRAM) storing non-volatile firmware data
- Asynchronous Communication Port

The RAA has the following structure:

Figure 109:
Random Access Area (RAA)

IP	Address	DWORD	Section	Description	Type[1]
RAM 176×32	0x000 to 0x07F	128	FWV	Firmware variables	RW
	0x080 to 0x087	8	FDB	Frontend Data Buffer	RW
	0x088 to 0x09B	20	FDB / (FWV)	Frontend Data Buffer / Firmware variables	RW
	0x09C to 0x09F	4	FWV	Firmware variables	RW
	0x0A0 to 0x0AF	16	FWV or (TEMP)	Firmware variables or temporary variables	RW
	0x0B0 to 0x0BF	16	NU	Not used	
Direct Mapped Registers	0x0C0 to 0x0CF	16	CR	Configuration Registers	RW
	0x0D0 to 0x0DF	16	SHR	Special Handling Registers	RW
	0x0E0 to 0x0EF	16	SRR	Status & Result Registers	RO
	0x0F0 to 0x0F7	8	NU	Not used	
	0x0F8 to 0x0FB	4	DR	For internal use only	RO
	0x0FC to 0x0FF	4	NU	Not used	
NVRAM 120×32	0x100	1	FWCU_CS	Firmware Code User, Checksum	RW
	0x101	1	FWDU_CS	Firmware Data User, Checksum	RW
	0x102 to 0x16A	105	FWDU	Firmware Data	RW
	0x16B	1		ACR Autoconfig Release Code	RW
	0x16C to 0x177	12		CDU Configuration Data User	RW
NVRAM 8×32	0x178 to 0x17A	3	FWDA_CD	CDA Configuration Data **ScioSense**	RW[1]
	0x17B to 0x17C	2	PUID	Production Unique ID	RW[1]
	0x17D	1	FWDA_CS	Firmware Data SCIOSENSE, Checksum	RW[1]
	0x17E	1	FWCA_CS	Firmware Code SCIOSENSE, Checksum	RW[1]
	0x17F	1	FWD_CPT	For internal use only	RW[1]
	0x180 to 0x1BF	64	NU	Not used	
Direct Mapped Registers	0x1C0 to 0x1C7	8	ACP	Asynchronous communication port GET: CPU → SPI	RO/WO[2]
	0x1C8 to 0x1FF	56	NU	Not used	

(1) Write Access for Page 2 of NVRAM is disabled for users, and only foreseen for **ScioSense** production flow.
(2) Read only by remote interface, write only by CPU

12.2.1 Frontend Data Buffer (FDB)

The frontend data buffer is used by the ultrasonic measurement (including time of flight-, amplitude- and pulse-width-measurement). Time-of-flight measurement and the temperature measurement cover the same FBD section alternately. The RAM content of the FDB depends on which measurement has been executed recently.

FDB in case of Time-of-Flight Measurement

In case of an ultrasonic ToF measurement the FDB contains the results for up to 10 individual zero crossing data in up and down as well as the average over all as being configured. In addition, the pulse width ratio, the amplitude measurement result and the amplitude calibration result are stored.

Figure 110:
FDB in case of ToF

Addr	Name[1]	Description
0x080	**FDB_US_TOF_SUM_OF_ALL_U**	Ultrasonic TOF Sum of All Value Up
0x081	**FDB_US_PW_U**	Ultrasonic Pulse Width Ratio Up
0x082	**FDB_US_AM_U**	Ultrasonic Amplitude Value Up
0x083	**FDB_US_AMC_VH**	Ultrasonic Amplitude Calibrate Value High
0x084	**FDB_US_TOF_SUM_OF_ALL_D**	Ultrasonic TOF Sum of All Value Down
0x085	**FDB_US_PW_D**	Ultrasonic Pulse Width Ratio Down
0x086	**FDB_US_AM_D**	Ultrasonic Amplitude Value Down
0x087	**FDB_US_AMC_VL**	Ultrasonic Amplitude Calibrate Value Low
0x088	**FDB_US_TOF_0_U**	Ultrasonic TOF Up: Value 0 (1-0)
0x089	**FDB_US_TOF_1_U**	Ultrasonic TOF Up: Value 1 (1-1)
0x08A	**FDB_US_TOF_2_U**	Ultrasonic TOF Up: Value 2 (1-2)
0x08B	**FDB_US_TOF_3_U**	Ultrasonic TOF Up: Value 3 (2-0)
0x08C	**FDB_US_TOF_4_U**	Ultrasonic TOF Up: Value 4 (2-1)
0x08D	**FDB_US_TOF_5_U**	Ultrasonic TOF Up: Value 5 (2-2)
0x08E	**FDB_US_TOF_6_U**	Ultrasonic TOF Up: Value 6 (3-0)
0x08F	**FDB_US_TOF_7_U**	Ultrasonic TOF Up: Value 7 (3-1)
0x090	**FDB_US_TOF_8_U**	Ultrasonic TOF Up: Value 8 (3-2)
0x091	**FDB_US_TOF_9_U**	Ultrasonic TOF Up: Value 0 (3-3)
0x092	**FDB_US_TOF_0_D**	Ultrasonic TOF Down: Value 0 (1-0)
0x093	**FDB_US_TOF_1_D**	Ultrasonic TOF Down: Value 1 (1-1)
0x094	**FDB_US_TOF_2_D**	Ultrasonic TOF Down: Value 2 (1-2)
0x095	**FDB_US_TOF_3_D**	Ultrasonic TOF Down: Value 3 (2-0)
0x096	**FDB_US_TOF_4_D**	Ultrasonic TOF Down: Value 4 (2-1)
0x097	**FDB_US_TOF_5_D**	Ultrasonic TOF Down: Value 5 (2-2)
0x098	**FDB_US_TOF_6_D**	Ultrasonic TOF Down: Value 6 (3-0)
0x099	**FDB_US_TOF_7_D**	Ultrasonic TOF Down: Value 7 (3-1)

Addr	Name[1]	Description	
0x09A	**FDB_US_TOF_8_D**	Ultrasonic TOF Down: Value 8	(3-2)
0x09B	**FDB_US_TOF_9_D**	Ultrasonic TOF Down: Value 0	(3-3)

(1) Important registers are marked with blue

FDB in case of Temperature Measurement

The temperature measurement is both based on RDC's discharge time measurement. Various combinations are possible, and the FDB content varies accordingly.

Figure 111:
Combinations for temperature measurement

Mode	
Off	
Internal	
Internal + 2-wire	**1 port**
	2 ports
2-wire	**1 port**
	2 ports
4-wire	**1 port**
	2 ports

Figure 112:
FDB in case of Temperature Measurement

Addr	Name[1]	Unit[1]	Seq.[2]	Description
0x080	**FDB_TPM1_M1AB_RAB_G12**	C	1	Gain compensation
	FDB_TPM1_M1A_RAB_G12			
	FDB_TPM1_M1B_RAB_G12			
0x081	**FDB_TPM1_RAB_G12**			Reference port REF-AB
0x082	**FDB_TPM1_M1A_G12**	T		Temperature port M1-A
0x083	**FDB_TPM1_M2A_G12**			Temperature port M2-A
0x084	**FDB_TPM1_RA_G12**	C		RDSON compensation
0x085	**FDB_TPM1_MI_R_G12**	I		Internal temperature reference
0x086	**FDB_TPM1_MI_RM_G12**			Internal temperature compensation
0x087	**FDB_TPM1_MI_M_G12**			Internal temperature measurement
0x088 to 0x8D	**NOT USED**			
0x08E	**FDB_TPM2_M1AB_RAB_G12**	C	2	Gain compensation
	FDB_TPM2_M1A_RAB_G12			

Addr	Name[1]	Unit[1]	Seq.[2]	Description
	FDB_TPM2_M1B_RAB_G12			
0x08F	**FDB_TPM2_RAB_G12**			Reference port REF-AB
0x090	**FDB_TPM2_M1A_G12**	T		Temperature port M1-A
0x091	**FDB_TPM2_M2A_G12**			Temperature port M2-A
0x092	**FDB_TPM2_RA_G12**	C		RDSON compensation
0x093	**FDB_TPM2_MI_R_G12**	I		Internal temperature reference
0x094	**FDB_TPM2_MI_RM_G12**			Internal temperature compensation
0x095	**FDB_TPM2_MI_M_G12**			Internal temperature measurement

(1) C [grey] = Compensation T [red] = Temperature I [orange] = Internal temperature
(2) Seq. = measurement sequence

Figure 113:
FDB in case of 4-wire Temperature

Addr	Name[1]	Seq.[1]	Description
0x080	**FDB_T4W1_M1AB_RAB_G12**	1	Gain compensation
0x081	**FDB_T4W1_RAB_G12**		Reference port REF-AB
0x082	**FDB_T4W1_M1AB_G12**		Temperature port M1-AB
0x083	**FDB_T4W1_M2AB_G12**		Temperature port M2-AB
0x084	**FDB_T4W1_RA_G12**		Reference port REF A
0x085	**FDB_T4W1_RB_G12**		Reference port REF B
0x086	**FDB_T4W1_M1A_G12**		Temperature port M1-A
0x087	**FDB_T4W1_M1B_G12**		Temperature port M1-B
0x088	**FDB_T4W1_M1AB_G1**		Temperature port M1-AB
0x089	**FDB_T4W1_M1AB_G2**		Temperature port M1-AB
0x08A	**FDB_T4W1_M2A_G12**		Temperature port M2-A
0x08B	**FDB_T4W1_M2B_G12**		Temperature port M2-B
0x08C	**FDB_TPM1_M2B_G1**		Temperature port M2-AB
0x08D	**FDB_TPM1_M2B_G2**		Temperature port M2-AB
0x08E	**FDB_T4W2_M1AB_RAB_G12**	2	Gain compensation
0x08F	**FDB_T4W2_RAB_G12**		Reference port REF-AB
0x090	**FDB_T4W2_M1AB_G12**		Temperature port M1-AB
0x091	**FDB_T4W2_M2AB_G12**		Temperature port M2-AB
0x092	**FDB_T4W2_RA_G12**		Reference port REF A
0x093	**FDB_T4W2_RB_G12**		Reference port REF B
0x094	**FDB_T4W2_M1A_G12**		Temperature port M1-A
0x095	**FDB_T4W2_M1B_G12**		Temperature port M1-B
0x096	**FDB_T4W2_M1AB_G1**		Temperature port M1-AB
0x097	**FDB_T4W2_M1AB_G2**		Temperature port M1-AB

Addr	Name[1]	Seq.[1]	Description
0x098	**FDB_T4W2_M2A_G12**		Temperature port M2-A
0x099	**FDB_T4W2_M2B_G12**		Temperature port M2-B
0x09A	**FDB_T4W2_M2AB_G1**		Temperature port M2-AB
0x09B	**FDB_T4W2_M2AB_G2**		Temperature port M2-AB

(1) Seq. = Measurement sequence

12.2.2 Configuration Registers

The AS6031 has 15 configuration registers of up to 32 bit word length. Configuration registers mainly contain fixed parameters which define the operation of all functional blocks of the UFC. They can be automatically initialized by the bootloader from firmware data in the NVRAM.

Figure 114:
Configuration Registers Overview

Addr	Name	Description
0x0C0	**CR_WD_DIS**	Watchdog Disable
0x0C1	**CR_IFC_CTRL**	Interfaces Control
0x0C2	**CR_GP_CTRL**	General Purpose Control
0x0C3	**CR_USM_OPT**	USM: Options
0x0C4	**CR_IEH**	Interrupt & Error Handling
0x0C5	**CR_CPM**	Clock & Power Management
0x0C6	**CR_MRG_TS**	Measure Rate Generator & Task Sequencer
0x0C7	**CR_TPM**	Temperature Measurement
0x0C8	**CR_USM_PRC**	USM: Processing
0x0C9	**CR_USM_FRC**	USM: Fire & Receive Control
0x0CA	**CR_USM_TOF**	USM: Time of Flight
0x0CB	**CR_USM_AM**	USM: Amplitude Measurement
0x0CC	**CR_TRIM1**	Trim Parameter
0x0CD	**CR_TRIM2**	Trim Parameter
0x0CE	**CR_TRIM3**	Trim Parameter
0x0CF	**NOT USED**	Not used

12.2.3 Special Handling Registers (SHR)

The AS6031 has 15 special handling registers of up to 32 bit word length. Special handling registers define the operation of the various units of UFC, like the configuration registers. Unlike the configuration registers, they contain data that is supposed to change during operation. Most of these

registers are not automatically initialized. The bootloader initializes SHR_TOF_RATE with FWD(118)[29:24] and SHR_ZCD_FHL_U/D with FWD(119)[31:24].

Figure 115:
Special Handling Registers Overview

Addr	Name	Description
0x0D0	**SHR_TOF_RATE**	Time-of-Flight rate
0x0D1	**SHR_USM_RLS_DLY_U**	Multi-hit Start Delay Up
0x0D2	**SHR_USM_RLS_DLY_D**	Multi-hit Start Delay Down
0x0D3	**SHR_GPO**	General Purpose Out
0x0D4	**SHR_PI_NPULSE**	Pulse Interface Number of Pulses
0x0D5	**SHR_PI_TPA**	Pulse Interface Time Pulse Distance
0x0D6	**SHR_PI_IU_TIME**	Pulse Interface, Internal Update Time Distance
0x0D7	**SHR_PI_IU_NO**	Pulse Interface Number of internal Update
0x0D8	**NOT USED**	not used
0x0D9	**SHR_ZCD_LVL**	Zero cross detection, level
0x0DA	**SHR_ZCD_FHL_U**	Zero Cross Detection First Hit Level Up
0x0DB	**SHR_ZCD_FHL_D**	Zero Cross Detection First Hit Level Down
0x0DC	**SHR_CPU_REQ**	CPU Requests
0x0DD	**SHR_EXC**	Executables
0x0DE	**SHR_RC**	Remote Control
0x0DF	**SHR_RC_RLS**	Release Code for actions of SHR_RC

12.2.4 Status & Result Registers

The AS6031 has 14 status & result registers of up to 32 bit word length. The status & result registers contain information generated by the chip hardware, e.g. status information like error flags or timing information, or measurement values from various hard-coded calibrations. It is not possible to write them directly.

Figure 116:
Status & Result Registers Overview

Addr	Name	Description
0x0E0	**SRR_IRQ_FLAG**	Interrupt Flags
0x0E1	**SRR_ERR_FLAG**	Error Flags
0x0E2	**SRR_FEP_STF**	Frontend Processing Status Flags
0x0E3	**SRR_GPI**	General Purpose In
0x0E4	**SRR_HCC_VAL**	High-Speed Clock Calibration Value
0x0E5	**SRR_VCC_VAL**	Measurement Value for VCC Voltage
0x0E6	**SRR_TSV_HOUR**	Time Stamp Value: Hours

Addr	Name	Description
0x0E7	**SRR_TSV_MIN_SEC**	Time Stamp Value: Minutes & Seconds
0x0E8	**NOT USED**	not used
0x0E9	**SRR_TS_TIME**	Task Sequencer Time
0x0EA	**SRR_MSC_STF**	Miscellaneous Status Flags
0x0EB	**SRR_I2C_RD**	2-wire Master Interface Read Data
0x0EC	**SRR_FWU_RNG**	Range Firmware Code User
0x0ED	**SRR_FWU_REV**	Revision Firmware Code User
0x0EE	**SRR_FWA_REV**	Revision Firmware Code **ScioSense**
0x0EF	**NOT USED**	Not used

12.3 Detailed Register Description

12.3.1 Frontend Data Buffer

Data Format of TDC Time Values in Frontend Data Buffer

This data format is given for all provided result values in Frontend Data Buffer, except value for pulse width ratio.

Figure 117:
FDB DATA FORMAT OF TDC DATA

Bit	Description
31:0	**TDC Time Value** Unsigned integer, 1 LSB: $1/2^{16} * t_{period}(HSO)$ $t_{period}(HSO) = 250$ ns TDC running with 4 MHz $t_{period}(HSO) = 125$ ns TDC running with 8 MHz

Notes:
1. The sum of TOF values may not exceed 16 ms @ 4 MHz, 8 ms @ 8 Mhz. Individual TOF values shall not exceed 4 ms @ 4 MHz, 2 ms @ 8 MHz
2. The average of the TOF hits, (FDB_US_TOF_SUM_OF_ALL_x / TOF_HIT_SUM_NO), is shifted by (TOF_HIT_SUM_NO – 1)/2*T_{ref} versus the first hit FDB_US_TOF_0_x

Data Format of Pulse Width Ratio

This data format is only given for value of pulse width ratio.

Figure 118:
FDB DATA FORMAT OF PULSE WIDTH DATA

Bit	Description
31:0	**Pulse Width Ratio** Ratio of pulse width between first hit and start hit Unsigned integer [7:0], 1 LSB: 1/ 2^7, Range: 0 to 1.992

12.3.2 Configuration Registers

CR_WD_DIS Register (Address 0x0C0)

Figure 119:
CR_WD_DIS Register

Addr: 0x0C0		CR_WD_DIS (Watchdog Disable)
Bit	**Bit Name**	**Bit Description**
31:0	**WS_DIS**	Code to disable Watchdog: 0x48DB_A399, Write only register, with default 0xAF0A7435. Status of watchdog can be checked in **WD_DIS** in register **SRR_MSC_STF**

CR_IFC_CTRL Register (Address 0x0C1)

Figure 120:
CR_IFC_CTRL Register

Addr: 0x0C1		CR_IFC_CTRL (Interfaces Control)
Bit	**Bit Name**	**Bit Description**
7:0	**PI_TPW**	Pulse Interface, Pulse Width = **PI_TPW** * 976.5625 µs (**LP_MODE** = 1), = **PI_TPW** * 1 ms (**LP_MODE** = 0)
8	**PI_EN**	Pulse Interface Enable, if operating in flow meter mode 0: Pulse Interface disabled 1: Pulse Interface enabled
9	**PI_OUT_MODE**	0: Output of pulses on 1 line with additional direction signal 1: Output of pulses on different lines for each direction
10	**PI_UPD_MODE**	0: Automatic Update disabled, only by **PI_UPD** in **SHR_EXC** 1: Automatic Update wit next TOF Trigger
11	**NOT_USED**	Mandatory setting: b0

Addr: 0x0C1		CR_IFC_CTRL (Interfaces Control)
Bit	**Bit Name**	**Bit Description**
13:12	**I2C_MODE**	2-wire master interface mode, I2C like 00: I2C disabled 01: I2C enabled on GPIO 0/1 10: I2C enabled on GPIO 2/3 11: Not allowed
20:14	**I2C_ADR**	2-wire master interface slave address
21	**NOT_USED**	Mandatory setting: b0
23:22	**SPI_INPORT_CFG**	Configuration of SPI input ports: SSN, MOSI & SCK 00: Inputs High Z (recommended if SPI is connected) 01: Inputs Pull Up (recommended if SPI is disconnected) 10: Inputs Pull Down 11: Inputs High Z
31:24	**NOT USED**	

CR_GP_CTRL Register (Address 0x0C2)

Figure 121:
CR_GP_CTRL Register

Addr: 0x0C2		CR_GP_CTRL (General Purpose Control)
Bit	**Bit Name**	**Bit Description**
1:0	**GP0_DIR**	Direction of General Purpose Port 0 00: Output 01: Input Pull Up 10: Input Pull Down 11: Input High Z
3:2	**GP0_SEL**	Selection for General Purpose Port 0 Output (**GP0_DIR** = 00) 00: General Purpose Out[0] 01: Pulse Interface -> Pulse 10: Low Speed Clock 11: Ultrasonic Fire Burst Input (**GP2_DIR** = 01 / 10 / 11) provided to **SRR_GPI**[0] 00: Mandatory setting 01 / 1x : Not allowed
5:4	**GP1_DIR**	Direction of General Purpose Port 1 see definition for **GP0_DIR**

Addr: 0x0C2		CR_GP_CTRL (General Purpose Control)
Bit	**Bit Name**	**Bit Description**
7:6	**GP1_SEL**	Selection for General Purpose Port 1 Output (**GP1_DIR** = 00) 00: General Purpose Out[1] 01: Pulse Interface -> Direction 10: Error Flag (low active) 11: Ultrasonic Direction Input (**GP1_DIR** = 01 / 10 / 11) provided to **SRR_GPI**[1] 00: Mandatory setting 01 / 1x : Not allowed
9:8	**GP2_DIR**	Direction of General Purpose Port 2 see definition for **GP0_DIR**
11:10	**GP2_SEL**	Select of General Purpose Port 2 Output (**GP2_DIR** = 00) 00: General Purpose Out[2] 01: Pulse Interface -> Pulse 10: TI_FP_PCH 11: not used Input (**GP2_DIR** = 01 / 10 / 11) provided to **SRR_GPI**[2] 00: Mandatory setting 01 / 1x : Not allowed
13:12	**GP3_DIR**	Direction of General Purpose Port 3 00: Output 01: Input Pull Up 10: Input Pull Down 11: Input High Z
15:14	**GP3_SEL**	Selection for General Purpose Port 3 Output (**GP3_DIR** = 00) 00: General Purpose Out[3] 01: Pulse Interface -> Direction 10: not used 11: Ultrasonic Fire Busy (**TI_FIRE_BUSY**) Input (**GP3_DIR** = 01 / 10 / 11) provided to **SRR_GPI**[3] 00: Mandatory setting 01 / 1x : Not allowed
17:16	**GP4_DIR**	Direction of General Purpose Port 4 00: Output 01: Input Pull Up 10: Input Pull Down 11: Input High Z
19:18	**GP4_SEL**	Selection for General Purpose Port 4 Output (**GP4_DIR** = 00): 00: General Purpose Out[4] 01: Ultrasonic Measurement Busy 10: PGA Enable (**TI_PGA_EN**) 11: not used Input (**GP4_DIR** = 01 / 10 / 11) provided to **SRR_GPI**[4] 00: Mandatory setting 01 / 1x : Not allowed

Addr: 0x0C2		CR_GP_CTRL (General Purpose Control)
Bit	**Bit Name**	**Bit Description**
21:20	**GP5_DIR**	Direction of General Purpose Port 5. 00: Output 01: Input Pull Up 10: Input Pull Down 11: Input High Z
23:22	**GP5_SEL**	Selection for General Purpose Port 5 Output (GP5_DIR = 00) 00: General Purpose Out[5] 01: not used 10: PGA VREF Enable (**TI_PGA_VREF**) 11: not used Input (GP5_DIR = 01 / 10 / 11) provided to **SRR_GPI**[5] 00: Mandatory setting 01 / 1x : Not allowed
31:24	**NOT_USED**	Not used

CR_USM_OPT Register (Address 0x0C3)

Figure 122:
CR_USM_OPT Register

Addr: 0x0C3		CR_USM_OPT (Ultrasonic Measurement Options)
Bit	**Bit Name**	**Bit Description**
4:0	**USM_OPT**	Mandatory setting b00001
31:5	**NOT USED**	

CR_IEH (Interrupt & Error Handling)

Figure 123:
CR_IEH Register

Addr: 0x0C4		CR_IEH (Interfaces Control)
Bit	**Bit Name**	**Bit Description**
0	**EF_EN_TDC_TMO**	Error Flag Enable, TDC Timeout
1	**EF_EN_TOF_TMO**	Error Flag Enable, TOF Timeout
2	**EF_EN_AM_TMO**	Error Flag Enable, Amplitude Measurement Timeout
3	**EF_EN_TM_OC**	Error Flag Enable, Temperature Measurement Open Circuit

Addr: 0x0C4		CR_IEH (Interfaces Control)
Bit	**Bit Name**	**Bit Description**
4	**EF_EN_TM_SC**	Error Flag Enable, Temperature Measurement Short Circuit
5	**EF_EN_ZCC_ERR**	Error Flag Enable, Zero Cross Calibration Error
6	**EF_EN_LBD_ERR**	Error Flag Enable, Low Battery Detect Error
7	**EF_EN_USM_SQC_TMO**	Error Flag Enable, Ultrasonic Sequence Timeout
8	**EF_EN_TM_SQC_TMO**	Error Flag Enable, Temperature Sequence Timeout
9	**EF_EN_TSQ_TMO**	Error Flag Enable, Task Sequencer Timeout
10	**EF_EN_I2C_ACK_ERR**	Error Flag Enable, EEPROM Acknowledge Error
11	**NOT USED**	Mandatory setting: b0
12	**EF_EN_NVM_FWCU_ERR**	Error Flag Enable, NVM FWCU Error
13	**EF_EN_NVM_FWDU_ERR**	Error Flag Enable, NVM FWDU Error
14	**EF_EN_NVM_FWA_ERR**	Error Flag Enable, NVM Applied Firmware Error
15	**EF_EN_CPU_ERR**	Error Flag Enable, CPU Error
16	**IRQ_EN_TSQ_FNS**	Interrupt Request Enable, Task Sequencer finished
17	**IRQ_EN_TRANS_FNS**	Interrupt Request Enable, FW Transaction finished
18	**IRQ_EN_BLD_FNS**	Interrupt Request Enable, Bootload finished
19	**IRQ_EN_CHKSUM_FNS**	Interrupt Request Enable, Checksum generation finished
20	**IRQ_EN_FW_S**	Interrupt Request Enable , Firmware, synchronized with task sequencer
21	**IRQ_EN_TSQ_TO**	Interrupt Request Enable, Task Sequencer Timeout
22	**NOT_USED**	Mandatory to set: b0
23	**IRQ_EN_ERR_FLAG**	Interrupt Request Enable, Error Flag
26:24	**NOT_USED**	Mandatory setting: b000
27	**CPU_REQ_EN_GPH**	CPU Request Enable, General Purpose Handling 0: disabled 1: enabled, to be triggered by GP Timer via **TS_GPT_RATE**, via **SHR_EXC** or **RC_REQ_GPH**

Addr: 0x0C4		CR_IEH (Interfaces Control)
Bit	**Bit Name**	**Bit Description**
31:28	**TS_GPT_RATE**	General Purpose Timer Rate 0000: GPT Timer disabled 0001: 1 sec 0010: 2 sec 0011: 5 sec 0100: 10 sec 0101: 30 sec 0110: 1 min 0111: 2 min 1000: 5 min 1001: 10 min 1010: 30 min 1011: 1 h 1100: 2 h 1101: 6 h 1110: 24 h 1111: 48 h If enabled, also CPU_REQ_EN_GPH must be set !

CR_CPM (Clock- & Power-Management)

Figure 124:
CR_CPM Register

Addr: 0x0C5		CR_CPM (Clock- & Power-Management)
Bit	**Bit Name**	**Bit Description**
0	**HSC_DIV_MODE**	High Speed Clock Divider Mode 0: Recommended for HS_CLK = 4 MHz 1: Recommended for HS_CLK = 8 MHz
1	**NOT_USED**	Mandatory to set: b0
4:2	**HSC_CLK_ST**	High-Speed Clock Settling Time 000: On Request, Settling Time 74 µs 001: On Request, Settling Time 104 µs 010: On Request, Settling Time 135 µs 011: On Request, Settling Time 196 µs 100: On Request, Settling Time 257 µs 101: On Request, Settling Time 379 µs 110: On Request, Settling Time 502 µs 111: On Request, Settling Time ~5000 µs
7:5	**NOT_USED**	Recommended setting: b001
8	**HSC_DIV**	High-Speed Clock Divider 0: Recommended for HS_CLK = 4 MHz 1: HS_CLK divided by 2 for all individual high speed clock dividers. Recommended for HS_CLK = 8 MHz

Addr: 0x0C5		CR_CPM (Clock- & Power-Management)
Bit	**Bit Name**	**Bit Description**
11:9	**HSC_RATE**	High-Speed Clock Calibration Rate, every Nth measure cycle trigger 000: disabled 001: every 010: every 2nd 011: every 5th 100: every 10th 101: every 20th 110: every 50th 111: every 100th
12	**HSC_MODE_CPU**	High-Speed Clock Mode CPU 0: High Speed Clock for CPU running with 4 MHz 1: High Speed Clock for CPU running with 1 MHz Note: Clock source can be changed in FW code with opcode clkmode, default is internal CPU clock.
15:13	**VM_RATE**	VCC Voltage measurement rate, every Nth measure cycle trigger 000: disabled 001: every 010: every 2nd 011: every 5th 100: every 10th 101: every 20th 110: every 50th 111: every 100th
21:16	**LBD_TH**	Low battery detection threshold, can be used with V_{CC} measurement 1 LSB: 25 mV LBD_TH = 0: 2.15 V LBD_TH = 63: 3.725 V
22	**TSV_UPD_MODE**	Time stamp update mode 0: updated by **TSV_UPD** in SHR_EXC 1: automatically updated every measure cycle trigger
23	**BF_SEL**	Base Frequency Select 0: 50 Hz T_{BF} = 20 ms 1: 60 Hz T_{BF} = 16.66 ms
27:24	**NOT_USED**	Mandatory to set: b0000
28	**NOT_USED**	Set to b0
30:29	**TI_PATH_SEL**	Transducer Fire Buffer Impedance: 00:= buffer disabled 01:= enables 550 Ohm Buffer 10:= enables 350 Ohm buffer 11:= enables 214 Ohm buffer
31	**NOT_USED**	Mandatory to set: b0

CR_MRG_TS (Measure Rate Generator & Task Sequencer)

Figure 125:
CR_MRG_TS Register

Addr: 0x0C6		CR_MRG_TS (Measure Rate Generator & Task Sequencer)
Bit	**Bit Name**	**Bit Description**
12:0	**MR_CT**	Measure rate cycle time 0: disabled 1 to 8191: Cycle time = **MR_CT** x 976.5625 µs (**LP_MODE** = 1), = **MR_CT** x 1 ms (**LP_MODE** = 0)
13	**TS_MCM**	Task Sequencer Measure Cycle Mode 0: Cycle Trigger in same phase for USM and TM 1: Cycle Trigger in different phases for USM and TM
14	**TS_PP_T_EN**	Enables final post processing T (after last measurement task) 0: Post Processing T disabled 1: Post Processing T enabled
15	**TS_PP_F_EN**	Enables post processing F (after flow and amplitude measurement task) 0: Post Processing F disabled 1: Post Processing F enabled
16	**TS_PP_MODE**	Post processing mode (only if post processing is enabled) 0: Post processing requested with every task sequencer trigger 1: Post processing only requested if a measurement task is requested
19:17	**TS_CST_RATE**	Firmware Check(sum) Timer Rate 000: disabled 001: 1h 010: 2h 011: 6h 100: 24h 101: 48h 110: 96h 111: 168h
23:20	**TS_NVR_RATE**	Recall Timer Rate 0000: NVR Timer disabled … 1011: 1 h 1100: 2 h 1101: 6 h 1110: 24 h 1111: 48 h
25:24	**NOT_USED**	Mandatory to set: b01

Addr: 0x0C6		CR_MRG_TS (Measure Rate Generator & Task Sequencer)
Bit	**Bit Name**	**Bit Description**
30:26	**NOT_USED**	Mandatory to set: b00000
31	**TS_CST_MODE**	Checksum Handling Mode 0: performed as soon as timer request occurs (recommended if TS_MCM = 0) 1: only performed if no TPM or USM measurement task is requested in this task sequencer cycle (recommended if TS_MCM = 1)

CR_TPM (Temperature Measurement)

Figure 126:
CR_TPM Register

Addr: 0x0C7		CR_TPM (Temperature Measurement)
Bit	**Bit Name**	**Bit Description**
9:0	**TM_RATE**	Temperature Measurement Rate 0: disabled 1 to 1023: Rate related to sequencer cycle trigger
12:10	**TPM_PAUSE**	Pause time between 2 temperature measurements 00x: not used 010: Pause = 0.25 * T(BF_SEL) ms 011: Pause = 0.5 * T(BF_SEL) ms 100: Pause = 1.0 * T(BF_SEL) ms 101: Pause = 1.5 * T(BF_SEL) ms 110: Pause = 2.0 * T(BF_SEL) ms 111: Pause = 2.5 * T(BF_SEL) ms
15:13	**TM_MODE**	Temperature Measurement Mode 000: Off 001: Internal only 010: Internal & 2-wire/1 port 011: Internal & 2-wire/2 ports 100: 2-wire/1 port 101: 2-wire/2 ports 110: 4-wire/1 port 111: 4-wire/2 ports
16	**NOT USED**	Mandatory to set: b0
17	**TPM_PORT_MODE**	Temperature Measurement Port Mode 0: Inactive ports pulled to GND while measurement 1: Inactive ports set to HighZ while measurement (only for extern measurement)

Addr: 0x0C7		CR_TPM (Temperature Measurement)
Bit	**Bit Name**	**Bit Description**
19:18	**TM_PORT_ORDER**	Temperature Measurement Port Order 10: 1st measurement: default order / 2nd measurement: reversed order (recommended)
21:20	**TPM_CLOAD_TRIM**	Temperature Measurement Load Trim Defines a delay between enabling of measure port(s) and starting measurement (switching point of CLOAD between charging & discharging) 10: 3.95 µs ± 1ns (recommended)
22	**TPM_CYCLE_SEL**	Temperature Measurement Cycle Select 0: 512 µs (recommended) 1: 1024 µs
23	**TPM_FAKE_NO**	Number of Fake measurements 0: 2 fake measurements (recommended) 1: 8 fake measurements
31:24	**NOT USED**	Not used Mandatory setting: h00

CR_USM_PRC (Ultrasonic Measurement Processing)

Figure 127:
CR_USM_PRC Register

Addr: 0x0C8		CR_USM_PRC (Ultrasonic Measurement Processing)
Bit	**Bit Name**	**Bit Description**
2:0	**USM_PAUSE**	Pause time between two ultrasonic measurements 000: no pause, only 1 measurement performed * 001: not allowed 010: 0.25 * T(BF_SEL) ms 011: 0.5 * T(BF_SEL) ms 100: 1.0 * T(BF_SEL) ms 101: 1.5 * T(BF_SEL) ms 110: 2.0 * T(BF_SEL) ms 111: 2.5 * T(BF_SEL) ms If no pause is configured (USM_PAUSE = 0), CR_TRIM2[7:6] has to be configured to b00.
3	**NOT_USED**	Mandatory setting: b0
5:4	**USM_DIR_MODE**	Ultrasonic Measurement Direction Mode 00: Always starting firing via UP-buffer 01: Always starting firing via DOWN-buffer 1x: Toggling direction with every ultrasonic measurement

Addr: 0x0C8		CR_USM_PRC (Ultrasonic Measurement Processing)
Bit	**Bit Name**	**Bit Description**
15:6	**USM_NOISE_MASK_WIN**	Defines the window as long any signal (e.g. noise) is masked on receive path. Starting time refers to rising edge of 1st fire pulse. End time defines switching point between firing and receiving state of transducer interface. Offset: -0.4 µs 1 LSB: 1 µs
17:16	**USM_TO**	Timeout 00: 128 µs 01: 256 µs 10: 1024 µs 11: 4096 µs If HSC_DIV_TDC = 0 for HS_CLK = 4 MHz or it HSC_DIV_TDC = 1 for HS_CLK = 8 MHz. It starts with the sequence of the fire burst. It has to be selected to a value which covers fire burst sequence, time of flight and receive burst sequence.
18	**NOT_USED**	Mandatory to set: b0
19	**USM_RLS_MODE**	Select mode for multihit start release 0: Start release condition derived by detection of First Hit Level only 1: Start release condition derived by Ultrasonic Release Delay only or in combination by First Hit Level detection
22:20	**ZCC_RATE**	Zero Cross Calibration Rate Triggered by the measurement cycle trigger B 000: disabled 001: every cycle 010: every 2nd cycle 011: every 5th cycle 100: every 10th cycle 101: every 20th cycle 110: every 50th cycle 111: 100th cycle
31:23	**NOT_USED**	Mandatory to set: b000000000

CR_USM_FRC (Ultrasonic Measurement Fire & Receive Control)

Figure 128:
CR_USM_FRC Register

Addr: 0x0C9		CR_USM_FRC (Ultrasonic Measurement Fire & Receive Control)
Bit	**Bit Name**	**Bit Description**
6:0	**FBG_CLK_DIV**	Clock divider for fire burst generator Frequency = High speed clock divided by **FPG_CLK_DIV** 0, 1: not allowed 2 to 127: divided by 2 to 127
7	**FBG_MODE**	Fire Burst Generator Mode 0: Insertion of low phase 1: Insertion of high phase
15:8	**FBG_PHASE_INS**	Fire Burst Generator, Phase Insertion 1 LSB: 1/(2 * f(FBG_HS_CLK)) 0: 2 LSBs 1: 2 LSBs 2 to 255: 2 to 255 LSBs If $f_{FBG\_HS\_CLK} = f_{HS\_CLK}$ then it's recommended to configure an even value. Otherwise pulse width distortion of HS_CLK will affect inserted phase.
21:16	**FBG_BURST_PRE**	Fire Burst Generator, number of pulses in the initial sequence (pre-burst) 0: Not allowed 1 to 63: 1 to 63 pre pulses
27:22	**FBG_BURST_POST**	Fire Burst Generator, number of pulses in the ending sequence (post-burst) 0: Not allowed 1 to 63: 1 to 63 pre pulses
28	**NOT_USED**	Mandatory to set: b0
29	**NOT_USED**	Mandatory to set: b1
30	**TOF_HIT_MODE**	TOF data in FDB according to: 0: Multi-hit mode AS6031, 10 TOF data in 3 bundles 1: Multi-hit mode GP30, 10 TOF data in 1 bundle
31	**NOT_USED**	Mandatory to set: b0

CR_USM_TOF (Ultrasonic Measurement Time of Flight)

Figure 129:
CR_USM_TOF Register

Addr: 0x0CA		CR_USM_TOF (Ultrasonic Measurement Time of Flight)
Bit	**Bit Name**	**Bit Description**
0	**NOT USED**	Mandatory setting: b0
5:1	**TOF_HIT_START**	Defines number of detected hits (including first hit) before hit which is taken as TOF start hit for TDC measurement 0: 0 hits not allowed because start hit cannot be first hit 1: 1 hits (not recommended) 2: 2 hits …. 31: 31 hits
7:6	**TOF_HIT_IGN**	Number of multi hits ignored between two hits taken for TOF measurement 00: 0 hits 01: 1 hit 10: 2 hits 11: 3 hits
12:8	**TOF_HIT_SUM_NO**	Number of hits taken for sum value of TOF measurement 0: not allowed 1: 1 hit 2: 2 hits ... 31: 31 hits **Note**: The sum of TOF values may not exceed 16 ms @ 4 MHz, 8 ms @ 8 Mhz. Individual TOF values shall not exceed 4 ms @ 4 MHz, 2 ms @ 8 MHz
19:13	**TOF_HIT_END**	TOF_HIT_MODE =1: not applicable, set to **TOF_HIT_END** =127 TOF_HIT_MODE =0: Defines hit after start hit which triggers multi-hit end sequence 0: not allowed 1: 1 hit 2: 2 hits ... 127: 127 hits Necessary condition: **TOF_HIT_END** ≥ **TOF_HIT_SUM_NO** + 4
21:20	**NOT_USED**	Mandatory to set: b00

Addr: 0x0CA		CR_USM_TOF (Ultrasonic Measurement Time of Flight)
Bit	**Bit Name**	**Bit Description**
23:22	**TOF_EDGE_MODE**	Time of Flight, edge mode 00: Time measurement on positive edge of TOF Hit 01: Time measurement on negative edge of TOF Hit 10: Edge for TOF hit toggling after every measurement cycle 11: Edge for TOF hit toggling after every 2. measurement cycle
29:24	**TOF_RATE_INIT**	FWD copy of initial value for TOF rate Only important if autoconfig release code is set. Then, during the boot process, the according FWD cell content is copied into this register (with no further effect) but also into the register SHR_TOF_RATE from which TOF rate is supplied for dynamic operation.
31:30	**NOT USED**	Not used

CR_USM_AM (Ultrasonic Amplitude Measurement)

Figure 130:
CR_USM_AM Register

Addr: 0x0CB		CR_USM_AM (Ultrasonic Amplitude Measurement)
Bit	**Bit Name**	**Bit Description**
2:0	**AM_RATE**	Amplitude measurement rate 000: disabled 001: every TOF trigger 010: every 2nd TOF trigger 011: every 5th TOF trigger 100: every 10ht TOF trigger 101: every 20th TOF trigger 110: every 50th TOF trigger 111: every 100th TOF trigger
3	**NOT USED**	Set to default 0

Addr: 0x0CB		CR_USM_AM (Ultrasonic Amplitude Measurement)
Bit	**Bit Name**	**Bit Description**
8:4	**AM_PD_END**	Amplitude measurement, end of peak detection, defined by number of detected hits after hit count has been released 0: not allowed 1: after 1st detected hit 2: after 2nd detected hit … 30: after 30th detected hit 31: not allowed Recommended conditions: Amplitude Measurement disabled (AM_RATE = b000) **AM_PD_END** = 1 Amplitude Measurement enabled (AM_RATE > b000) **AM_PD_END** ≤ End of TOF measurement
11:9	**NOT USED**	Mandatory setting: b111
14:12	**AMC_RATE**	Amplitude measurement calibration rate 000: disabled 001: with every amplitude measurement 010: every 2nd amplitude measurement 011: every 5th amplitude measurement 100: every 10ht amplitude measurement 101: every 20th amplitude measurement 110: every 50th amplitude measurement 111: every 100th amplitude measurement
15	**PWD_EN**	Enables pulse width detection 0: pulse width detection disabled 1: pulse width detection enabled
18:16	**PGA_TRIM**	PGA_TRIM sets the DC gain of the PGA via trim bits in steps of 0 := 2 V/V, 5 := 10 V/V, 10 := 50 V/V, 1 := 3 V/V, 6 := 14 V/V, 11 := 69 V/V 2 := 4 V/V, 7 := 19 V/V
19	**NOT_USED**	Set to default 0
20	**PGA_EN_MODE**	PGA enable mode 0: PGA enabled as given by frontend control 1: PGA permanently enabled
21	**PGA_MODE**	Ultrasonic measurement PGA Mode 0: PGA disabled 1: PGA enabled
22	**NOT_USED**	Mandatory setting: b0

Addr: 0x0CB		CR_USM_AM (Ultrasonic Amplitude Measurement)
Bit	**Bit Name**	**Bit Description**
23	**AM_PD_START_MODE**	0: AM peak detection starts after noise mask window expires 1: AM peak detection starts after ultrasonic release delay expires Suitable only for combined start hit mode, when ultrasonic release delay is configured between end of noise mask window and ultrasonic receive burst
31:24	**ZCD_FHL_INIT**	FWD copy of initial value for first hit levels Only important if autoconfig release code is set. Then, during the boot process, the according FWD cell content is copied into this register (with no further effect) but also into the register SHR_FHL_U & SHR_FHL_D from which first hit levels are supplied for dynamic operation

CR_TRIM1 (Trim Parameter 1)

Figure 131:
CR_TRIM1 Register

Addr: 0x0CC		CR_TRIM1 (Trim Parameter 1)
Bit	**Bit Name**	**Bit Description**
31:0	**TRIM1**	Default 0x95A0C06C . Has to be **0x94A0C46C**

CR_TRIM2 (Trim Parameter 2)

Figure 132:
CR_TRIM2 Register

Addr: 0x0CD		CR_TRIM2 (Trim Parameter 2)
Bit	**Bit Name**	**Bit Description**
31:0	**TRIM2**	Default 0x40110000 . Has to be **0x401100C4**

CR_TRIM3 (Trim Parameter 3)

Figure 133:
CR_TRIM3 Register

Addr: 0x0CE		CR_TRIM3 (Trim Parameter 3)
Bit	**Bit Name**	**Bit Description**
31:0	**TRIM3**	Default 0x4027000F. Has to be **0x00A7400F**

12.3.3 Special Handling Registers

SHR_TOF_RATE (Time Of Flight Rate)

Figure 134:
SHR_TOF_RATE Register

Addr: 0x0D0		SHR_TOF_RATE (Time Of Flight Rate)
Bit	**Bit Name**	**Bit Description**
5:0	**TOF_RATE**	TOF Rate 0: TOF Measurement disabled 1 to 63: Rate of TOF Measurement relative to measure rate cycle trigger
31:6	**NOT USED**	Not used

SHR_USM_RLS_DLY_U (Ultrasonic Release Delay Up)

Figure 135:
SHR_USM_RLS_DLY_U Register

Addr: 0x0D1		SHR_USM_RLS_DLY_U (Ultrasonic Release Delay Up)
Bit	**Bit Name**	**Bit Description**
18:0	**USM_RLS_DLY_U**	Delay window in up direction, releasing ultrasonic measurement The start time of the delay window refers to rising edge of the 1st fire pulse 1 LSB: 7.8125 ns
31:19	**NOT USED**	Not used

SHR_USM_RLS_DLY_D (Ultrasonic Release Delay Down)

Figure 136:
SHR_USM_RLS_DLY_D Register

Addr: 0x0D2		SHR_USM_RLS_DLY_D (Ultrasonic Release Delay Down)
Bit	**Bit Name**	**Bit Description**
18:0	**USM_RLS_DLY_D**	Delay window in down direction, releasing ultrasonic measurement. The start time of the delay window refers to rising edge of the 1st fire pulse 1 LSB: 7.8125 ns
31:19	**NOT USED**	Not used

SHR_GPO (General Purpose Out)

Figure 137:
SHR_GPO Register

Addr: 0x0D3		SHR_GPO (General Purpose Out)
Bit	**Bit Name**	**Bit Description**
5:0	**GPO**	General Purpose Out
7:6	**NOT_USED**	Not used
8	**PI_OUT_FRC0**	Forces LOW on pulse output (unless **PI_OUT_FRC1** is set) Typically set by firmware for zero flow
9	**PI_OUT_FRC1**	Forces HIGH on pulse output (priority over **PI_OUT_FRC0**) Typically set by firmware for error indication
10	**PI_DIR_FRC0**	Forces Low on pulse direction (unless **PI_DIR_FRC1** is set) Typically set by firmware
11	**PI_DIR_FRC1**	Forces HIGH on pulse direction (priority over **PI_DIR_FRC0**) Typically set by firmware
12	**FWCU_CS_ERR**	FWCU checksum error Set by NVRAM check subroutine in ROM code Triggers **EF_NVM_FWCU_ERR**
13	**FWDU_CS_ERR**	FWDU checksum error Set by NVRAM check subroutine in ROM code Triggers **EF_NVM_FWDU_ERR**
18:14	**FWA_CS_ERR**	Different FWA checksum errors Set by NVRAM check subroutine in ROM code Triggers **EF_NVM_FWA_ERR**

Addr: 0x0D3		SHR_GPO (General Purpose Out)
Bit	**Bit Name**	**Bit Description**
19	**FW_ERR**	FW error Typically set by FW code Triggers **EF_FWA_ERR**
31:20	**NOT USED**	Not used

SHR_PI_NPULSE (Pulse Interface Number of Pulses)

Figure 138:
SHR_PI_NPULSE Register

Addr: 0x0D4		SHR_PI_NPULSE (Pulse Interface Number of Pulses)
Bit	**Bit Name**	**Bit Description**
31:0	**PI_NPULSE**	Number of pulses, signed integer 1 LSB: $1/2^{24}$

SHR_PI_TPA (Pulse Interface Time Pulse Distance)

Figure 139:
SHR_PI_TPA Register

Addr: 0x0D5		SHR_PI_TPA (Pulse Interface Time Pulse Distance)
Bit	**Bit Name**	**Bit Description**
15:0	**PI_TPA**	Minimal distance between two pulses 1 LSB: 0.97656 ms (**LP_MODE** = 1) 1 LSB: 1 ms (**LP_MODE** = 0) Mandatory condition: **PI_TPA** > **PI_TPW**
31:16	**NOT USED**	Not used

SHR_PI_IU_TIME (Pulse Interface Internal Update Time)

Figure 140:
SHR_ PI_IU_TIME Register

Addr: 0x0D6		SHR_PI_IU_TIME (Pulse Interface Internal Update Time)
Bit	**Bit Name**	**Bit Description**
15:0	**PI_IU_TIME**	Time between two internal updates 1 LSB: 0.97656 ms (**LP_MODE** = 1) 1 LSB: 1 ms (**LP_MODE** = 0) Mandatory condition: **PI_IU_TIME** > 2 and **PI_IU_TIME** > **PI_TPW**
31:16	**NOT USED**	Not used

SHR_PI_IU_NO (Pulse Interface Number of Auto Updates)

Figure 141:
SHR_ PI_IU_NO Register

Addr: 0x0D7		SHR_PI_IU_NO (Pulse Interface Number of Auto Updates)
Bit	**Bit Name**	**Bit Description**
7:0	**PI_IU_NO**	Number of internal updates between two general updates Recommended condition for uniformed pulse generation: (**PI_IU_NO** + 1) * **PI_IU_TIME** = **TOF_RATE** * **MR_CT**
31:8	**NOT USED**	Not used

SHR_ZCD_LVL (Zero Cross Detection Level)

Figure 142:
SHR_ZCD_LVL Register

Addr: 0x0D9		SHR_ZCD_LVL (Zero Cross Detection Level)
Bit	**Bit Name**	**Bit Description**
9:0	**ZCD_LVL**	Zero Cross Detection Level 1 LSB: ~ 0.88 mV
31:10	**NOT USED**	Not used

SHR_FHL_U (Zero Cross Detection Level)

Figure 143:
SHR_FHL_U Register

Addr: 0x0DA		SHR_FHL_U (Zero Cross Detection Level)
Bit	**Bit Name**	**Bit Description**
7:0	**ZCD_FHL_U**	First Hit Level Up 1 LSB ~ 0.88 mV, maximum 200 mV
31:8	**NOT USED**	Not used

SHR_FHL_D (First Hit Level Down)

Figure 144:
SHR_FHL_D Register

Addr: 0x0DB		SHR_ FHL_D (First Hit Level Down)
Bit	**Bit Name**	**Bit Description**
7:0	**ZCD_FHL_D**	First Hit Level Down 1 LSB ~ 0.88 mV, maximum 200 mV
31:8	**NOT USED**	Not used

SHR_CPU_REQ (CPU Requests)

It is strongly recommended to write to this register via SPI interface only for debug purpose. In this case, Bit 17 of CR_TRIM3 has to be configured to 0.

Figure 145:
SHR_CPU_REQ Register

Addr: 0x0DC		SHR_CPU_REQ (CPU Requests)
Bit	**Bit Name**	**Bit Description**
0	**CPU_REQ_BLD_EXC**(1)	CPU Request Bootloader Execute 0: Bootloader subroutine in CPU not requested 1: Bootloader subroutine in CPU requested Triggered by task sequencer, automatically cleared when CPU stops

Addr: 0x0DC		SHR_CPU_REQ (CPU Requests)
Bit	**Bit Name**	**Bit Description**
1	**CPU_REQ_CHKSUM**(1)	CPU Request Build Checksum 0: Build checksum in CPU not requested 1: Configuration compare in CPU requested Triggered by task sequencer, automatically cleared when CPU stops
2	**CPU_REQ_PP_T**(1)	CPU Request Post Processing PB 0: Post processing PB in CPU not requested 1: Post processing PB in CPU requested Triggered by task sequencer, automatically cleared when CPU stops
3	**CPU_REQ_PP_F**(1)	CPU Request Post Processing PA 0: Post processing PA in CPU not requested 1: Post processing PA in CPU requested Triggered by task sequencer, automatically cleared when CPU stops
4	**CPU_REQ_GPH**(1)	CPU Request General Purpose Handling 0: General purpose handling in CPU not requested 1: General purpose handling in CPU requested Triggered by task sequencer, automatically cleared when CPU stops
5	**CPU_REQ_FW_INIT**(1)	CPU Request Firmware Initialization 0: Firmware initialization not requested 1: Firmware initialization requested Triggered by bootloader sequence in ROM code, automatically cleared when CPU stops
6	**NOT USED**	Not used
7	**NOT USED**	Not used
8	**CPU_SFLAG_HSO_ST_TO**	0: High speed oscillator not settled yet (not in timeout condition) 1: High speed oscillator settled and stable (in timeout condition) Cleared by HSO_CLR in SHR_EXC Updated and valid only while CPU is running (synchronized by CPU clock)
9	**CPU_COM_REQ**	0: No communication request (RC_COM_REQ) set by SPI interface 1: Communication request (RC_COM_REQ) set by SPI interface Updated and valid only while CPU is running (synchronized by CPU clock)
10	**CPU_LS_CORE_CLK**	0: Low phase of LS_CORE_CLK 1: High phase of LS_CORE_CLK Updated and valid only while CPU is running (synchronized by CPU clock)
31:11	**NOT USED**	Not used

(1) BIT-T= Bits have to be cleared by the system program code or the user program code.

SHR_EXC (Executables)

Figure 146:
SHR_EXC Register

Addr: 0x0DD		SHR_EXC (Executables)
Bit	**Bit Name**	**Bit Description**
0	**IF_CLR**(1)	Interrupt Flag Clear 0: No action 1: Clears flag SRR_IRQ_FLAG
1	**EF_CLR**(1)	Error Flag Clear 0: No action 1: Clears flag SRR_ERR_FLAG
2	**FES_CLR**(1)	Frontend Status Clear 0: No action 1: Clears flag SRR_FEP_STF
3	**TSC_CLR**(1)	Time Stamp Clear 0: No action 1: Clears time stamp counter
4	**TSV_UPD**(1)	Time Stamp Value Update 0: No action 1: Update time stamp value
5	**PI_UPD**(1)	Pulse Interface Update 0: No action 1: Updates pulse interface
6	**BG_REFRESH**(1)	Bandgap Refresh 0: No action 1: Bandgap refresh
7	**MCT_CLR**(1)	Measure Cycle Timer Clear 0: No action 1: Clears measure cycle timer
8	**RATE_CTR_CLR**(1)	Rate Counter Clear 0: No action 1: Clears all rate counters
9	**ZCC_RNG_CLR**(1)	Zero Cross Calibration Range Clear 0: No action 1: Clears zero cross calibration range
10	**FW_IRQ_S**(1)	FW Interrupt Request, synchronized with task sequencer 0: No action 1: Interrupt request triggered by FW and synchronized with task sequencer
11	**NOT_USED**	Not used

Addr: 0x0DD		SHR_EXC (Executables)
Bit	**Bit Name**	**Bit Description**
12	**COM_REQ_CLR**(1)	Communication Request Clear 0: No action 1: Clears communication request via remote interface
13	**GPR_REQ_CLR**(1)	General Purpose Request Clear 0: No action 1: Clears general purpose request via remote interface
14	**GPH_TRIG**(1)	General Purpose Handling Trigger 0: No action 1: Triggers general purpose handling for CPU via task sequencer
15	**I2C_CLR**(1)	I2C Clear 0: No action 1: Clears I2C interface controller
16	**ACP_PAGE_TGL**(1)	Toggles ACP page (asynchronous communication)
17	**HSO_REQ**(1)	Requests high speed oscillator
18	**HSO_CLR**(1)	Clears high speed oscillator
31:19	**NOT USED**	Not used

(1) SCB= Self-clearing bit

SHR_RC (Remote Control)

The remote control register is implemented with radio buttons and self-clearing bits. It is used when operating in time conversion mode accessed by remote control. Radio buttons have the advantage in that single states of the register settings can be changed without knowing the complete state of the register. This saves a pre-reading of the register when operating in remote mode.

To change a dedicated bit, write a 1 to this one and a 0 to all others.

Figure 147:
SHR_RC Register

Addr: 0x0DE		SHR_RC (Remote Control)
Bit	**Bit Name**	**Bit Description**
1:0	**CFG_OK**(1)	UFC Configuration OK. Set by bootloader 00: No change of CFG_OK state (WO(3)) 01: Not properly configured 10: Properly configured 11: No change of CFG_OK state (WO)

Addr: 0x0DE		SHR_RC (Remote Control)
Bit	**Bit Name**	**Bit Description**
3:2	**HSC_DIV_STATE**	State of HSC_DIV in CR_CPM[8] (Read Only) 01: HSC_DIV = 0 10: HSC_DIV = 1 00, 11: not possible
5:4	**RC_FLAG2**(1)	User Definable Flag, can also be set by firmware. 00: No Change of RC_FLAG2 state (WO) 01: RC_FLAG2 not set 10: RC_FLAG2 set 11: No Change of RC_FLAG2 state (WO) This flag is used if flow firmware is applied. Please refer for appropriate firmware manual for detailed description.
7:6	**NOT_USED**	Mandatory to set: b01
9:8	**HSO_MODE**(1)	High Speed Oscillator Mode 00: No change of HSO_MODE state (WO) 01: HSO controlled as configured 10: HSO always on 11: No change of HSO_MODE state (WO)
11:10	**BG_MODE**(1)	Bandgap Mode 00: No change of BG_MODE state (WO) 01: Bandgap controlled as configured 10: Bandgap always on 11: No Change of BG_MODE state (WO)
13:12	**RC_FLAG3**(1)	User Definable Flag, can also be set by firmware. 00: No Change of RC_FLAG3 state (WO) 01: RC_FLAG2 not set 10: RC_FLAG2 set 11: No Change of RC_FLAG3 state (WO)
14	**SYS_RST**(2)	System Reset 0: No action 1: Performs a system reset Execution needs to be enabled by RC_RLS_2
15	**SYS_INIT**(2)	System Initialization 0: No action 1: Performs a system init Execution needs to be enabled by RC_RLS_2
16	**FW_STORE_ALL**(2)	Stores Firmware Code & Firmware Data 0: No action 1: Requests storing of complete firmware code & data Execution needs to be enabled by RC_RLS_1
17	**FW_STORE_LOCK**(2)	Stores & Lock Firmware Program Code & Firmware Data 0: No action 1: Requests storing & locking of user firmware program code & data Execution needs to be enabled by RC_RLS_1

Addr: 0x0DE		SHR_RC (Remote Control)
Bit	**Bit Name**	**Bit Description**
18	**FW_ERASE**(2)	Erases User Firmware Program Code & Firmware Data 0: No action : Requests erasing user firmware program code & data Execution needs to be enabled by RC_RLS_1
19	**FWC_RECALL**(2)	Recalls Firmware Program Code 0: No action 1: Requests recalling of firmware program code from Flash to SRAM Execution needs to be enabled by RC_RLS_1
20	**FWD_RECALL**(2)	Recalls Firmware Data 0: No action 1: Requests recalling of firmware data from Flash to SRAM Execution needs to be enabled by RC_RLS_1
21	**FWC_STORE**(2)	Stores Firmware Program Code 0: No action 1: Requests storing of firmware program code from SRAM to Flash Execution needs to be enabled by RC_RLS_1
22	**FWD_STORE**(2)	Stores Firmware Data (FWDU, user part only) 0: No action 1: Requests storing of user firmware data from SRAM to Flash Execution needs to be enabled by RC_RLS_1
31:23	**NOT USED**	Not used

(1) RB = Radio button
(2) SCB = Self-clearing bit
(3) WO = Write only, RO = Read only

SHR_RC_RLS (Remote Control Release)

Figure 148:
SHR_RC_RLS Register

Addr: 0x0DF		SHR_RC_RLS (Remote Control Release)
Bit	**Bit Name**	**Bit Description**
31:0	**RC_RLS**	Release codes for dedicated self-clearing bits in SHR_RC: RC_RLS_1 = h50F5_B8CA: Releases bits [22:16] in SHR_RC RC_RLS_2 = hAF0A_4735: Releases bits [15:14] in SHR_RC Status of RC_RLS_1/2 can be checked in RC_RLS_1/2 in SRR_MSC_STF Automatically cleared after one transaction via bits [22:14]. Release code must be re-written for each transaction again.

12.3.4 Status & Result Registers

The status registers contain all the flags indicating interrupt sources, error sources, updated data, GPIOs. In addition, they contain the measurement results for the high-speed clock calibration and for voltage measurement. The time stamp in hours, minutes and seconds is found there as well as I2C read data and firmware revision numbers.

SRR_IRQ_FLAG (Interrupt Flags)

Figure 149:
SRR_IRQ_FLAG Register

Addr: 0x0E0		SRR_IRQ_FLAG (Interrupt Flags)
Bit	**Bit Name**	**Bit Description**
0	**TSQ_FNS**	Task sequencer finished
1	**FW_TRANS_FNS**	Firmware transaction finished
2	**BLD_FNS**	Bootloader finished
3	**CHKSUM_FNS**	Checksum subroutine finished
4	**FW_IRQ_S**	Firmware interrupt request, synchronized with task sequencer
5	**TSQ_TMO**	Task sequencer timeout

Addr: 0x0E0		SRR_IRQ_FLAG (Interrupt Flags)
Bit	**Bit Name**	**Bit Description**
6	**NOT_USED**	Not used
7	**ERR_FLAG**	At least 1 error flag is set
31:8	**NOT USED**	Not used

SRR_ERR_FLAG (Error Flags)

Figure 150:
SRR_ERR_FLAG Register

Addr: 0x0E1		SRR_ERR_FLAG (Error Flags)
Bit	**Bit Name**	**Bit Description**
0	**EF_TDC_TMO**	Error flag TDC timeout
1	**EF_TOF_TMO**	Error flag TOF timeout
2	**EF_AM_TMO**	Error flag amplitude measurement timeout
3	**EF_TM_OC_ERR**	Error flag temperature measurement open circuit
4	**EF_TM_SC_ERR**	Error flag temperature measurement short circuit
5	**EF_ZCC_ERR**	Error flag zero cross calibration
6	**EF_LBD_ERR**	Error flag low battery detect
7	**EF_USM_SQC_TMO**	Error flag ultrasonic sequence timeout
8	**EF_TM_SQC_TMO**	Error flag temperature sequence timeout
9	**EF_TSQ_TMO**	Error flag task sequencer timeout
10	**EF_I2C_ACK_ERR**	Error flag EEPROM acknowledge
11	**NOT USED**	
12	**EF_NVM_FWCU_ERR**	Error flag NVM error in FWCU area
13	**EF_NVM_FWDU_ERR**	Error flag NVM error in FWDU area
14	**EF_NVM_FWA_ERR**	Error flag NVM error in any FWA area or set by FW (any bit set of SHR_GPO[19:14])
15	**EF_CPU_ERR**	CPU error (invalid program counter or PC stack overflow)
16	**NOT USED**	Not used

SRR_FEP_STF (Frontend Processing Status Flags)

Figure 151:
SRR_FEP_STF Register

Addr: 0x0E2		SRR_FEP_STF (Frontend Processing Status Flags)
Bit	**Bit Name**	**Bit Description**
0	**HCC_UPD**	High-Speed Clock Calibration Update 0: No update in SRR_HCC_VAL 1: Updated value in SRR_HCC_VAL
1	**TM_UPD**	Temperature Measurement Update 0: No update in frontend buffer 1: Updated value in temperature measurement related frontend buffer
2	**NOT USED**	Not used
3	**TPM_ST**	Temperature Subtask 0: Temperature measurement with 1 subtask 1: Temperature measurement with 2 subtasks
4	**US_U_UPD**	Ultrasonic Update in Up direction 0: No update in frontend buffer 1: Updated value in ultrasonic up area of frontend buffer
5	**US_D_UPD**	Ultrasonic Update in Down direction 0: No update in frontend buffer 1: Updated value in ultrasonic down area of frontend buffer
6	**US_TOF_UPD**	Ultrasonic Update for TOF measurement 0: No update in frontend buffer 1: Updated value in TOF area of frontend buffer
7	**US_TOF_EDGE**	TOF Measurement Edge 0: Positive edge 1: Negative edge
8	**US_AM_UPD**	Update for Amplitude measurement 0: No update in frontend buffer 1: Updated value in AM area of frontend buffer
9	**US_AMC_UPD**	Update for Amplitude Calibration Measurement 0: No update in frontend buffer 1: Updated value in AMC area of frontend buffer
31:10	**NOT USED**	Not used

SRR_GPI (General Purpose In)

Figure 152:
SRR_GPI Register

Addr: 0x0E3		SRR_GPI (General Purpose In)
Bit	**Bit Name**	**Bit Description**
5:0	**GPI**	General Purpose Input, default 0x3F
7:6	**NOT_USED**	Not used
8	**LP_MODE**	Low Power Mode, default 1
31:8	**NOT USED**	Not used

SRR_HCC_VAL (High-Speed Clock Calibration Value)

Figure 153:
SRR_ Register

Addr: 0x0E4		SRR_HCC_VAL (High-Speed Clock Calibration Value)
Bit	**Bit Name**	**Bit Description**
25:0	**HCC_VAL**	Clock calibration value, used for the correction factor. Eight times the real reference frequency at the TDC: 8 x f_{HSO} /Hz
31:26	**NOT USED**	Not used

SRR_VCC_VAL (VCC Value)

Figure 154:
SRR_VCC_VAL Register

Addr: 0x0E5		SRR_VCC_VAL (VCC Value)
Bit	**Bit Name**	**Bit Description**
5:0	**VCC_VAL**	Measured value of VCC voltage 1 LSB: 25 mV, default 0x2F VCC_VAL = 0: 2.15 V VCC_VAL = 63: 3.725 V
31:6	**NOT USED**	Not used

SRR_TS_HOUR (Time Stamp Hours)

Figure 155:
SRR_TS_HOUR Register

Addr: 0x0E6		SRR_TS_HOUR (Time Stamp Hours)
Bit	**Bit Name**	**Bit Description**
17:0	**TS_HOUR**	Timestamp Hours 1 LSB: 1h
31:18	**NOT USED**	Not used

SRR_TS_MIN_SEC (Time Stamp Minutes & Seconds)

Figure 156:
SRR_TS_MIN_SEC Register

Addr: 0x0E7		SRR_TS_MIN_SEC (Time Stamp Minutes & Seconds)
Bit	**Bit Name**	**Bit Description**
7:0	**TS_SEC**	Timestamp Minutes 1 LSB: 1min Range (0 to 59)
15:8	**TS_MIN**	TS_SEC: Timestamp Seconds 1 LSB: 1sec Range (0 to 59)
31:16	**NOT USED**	Not used

SRR_TS_TIME (Task Sequencer time)

Figure 157:
SRR_TS_TIME Register

Addr: 0x0E9		SRR_TS_TIME (Task Sequencer time)
Bit	**Bit Name**	**Bit Description**
11:0	**TS_TIME**	Consumed time within a task sequencer cycle Current time = TS_TIME * 1953,125 µs (LP_MODE = 1), = TS_TIME* 2 ms (LP_MODE = 0)
31:12	**NOT USED**	Not used

SRR_MSC_STF (Miscellaneous Status Flags)

Figure 158:
SRR_MSC_STF Register

Addr: 0x0EA		SRR_MSC_STF (Miscellaneous Status Flags)
Bit	**Bit Name**	**Bit Description**
0	**RC_RLS_1**	Release 1 of remote communication self-clearing bits
1	**STUP_TO**	Start-up timeout
2	**FW_UNLOCKED**	FW unlocked
3	**SI_BUSY**	Serial remote interface busy
4	**COM_REQ**	Communication request by remote interface
5	**GPR_REQ**	General purpose request by remote interface
6	**GPT_REQ**	General purpose request by GP timer
7	**GPH_REQ**	General purpose request by GPH_TRIG in SHR_EXC
8	**MCT_RLS**	Measure cycle timer release
9	**NVM_RDY**	NVRAM ready
10	**NVR_REQ**	Request by NVRAM recall timer
11	**NOT USED**	Not used
12	**HSO_ST_TO**	High speed oscillator settling timeout
13	**I2C_ACK**	2-wire interface acknowledge
14	**I2C_BSY**	2-wire interface busy
15	**WD_DIS**	Watchdog disabled
16	**RC_RLS_2**	Release 2 of remote communication self-clearing bits
31:17	**NOT USED**	Not used

SRR_I2C_RD (I2C Read Data)

Figure 159:
SRR_RD Register

Addr: 0x0EB		SRR_I2C_RD (I2C Read Data)
Bit	**Bit Name**	**Bit Description**
7:0	**I2C_DATA**	2-wire interface read data Read data from external device connected via 2-wire interface
31:8	**NOT USED**	Not used

SRR_FWU_RNG (FW User Range)

Figure 160:
SRR_FWU_RNG Register

Addr: 0x0EC		SRR_FWU_RNG (FW User Range)
Bit	**Bit Name**	**Bit Description**
11:0	**FWU_RNG**	FW User Range Defines end address FW user code. End address = FWU_RNG - 1
31:12	**NOT USED**	Not used

SRR_FWU_REV (FW User Revision)

Figure 161:
SRR_FWU_REV Register

Addr: 0x0ED		SRR_FWU_REV (FW User Revision)
Bit	**Bit Name**	**Bit Description**
31:0	**FWU_REV**	FW User Revision First 4 bytes in FW user code range, reserved for revision.

SRR_FWA_REV (FW SCIOSENSE Revision)

Figure 162:
SRR_FWA_REV Register

Addr: 0x0EE		SRR_FWA_REV (FW SCIOSENSE Revision)
Bit	**Bit Name**	**Bit Description**
31:0	**FWA_REV**	FW **ScioSense** Revision

12.3.5 Asynchronous Communication Port

The asynchronous communication port consists of two register banks to write 8 customer specific words. A register bank is only accessible by the FW in CPU. With ACP_PAGE_TGL in SHR_EXC it is possible to toggle between the two register banks. They are located in the RAA from 0x1C0 to 0x1C7. Over the remote interface the register content can be read. It is mandatory to read 0x1C0 first, it will

trigger that the register content after toggling in CPU FW will be latched and readable by remote interface.

Note: Any communication during a measurement can have a negative impact on the measurement quality. Therefore, it is recommended to use a filter in the postprocessing of the ToF measurements, i.e. outlier filter, median filter etc.

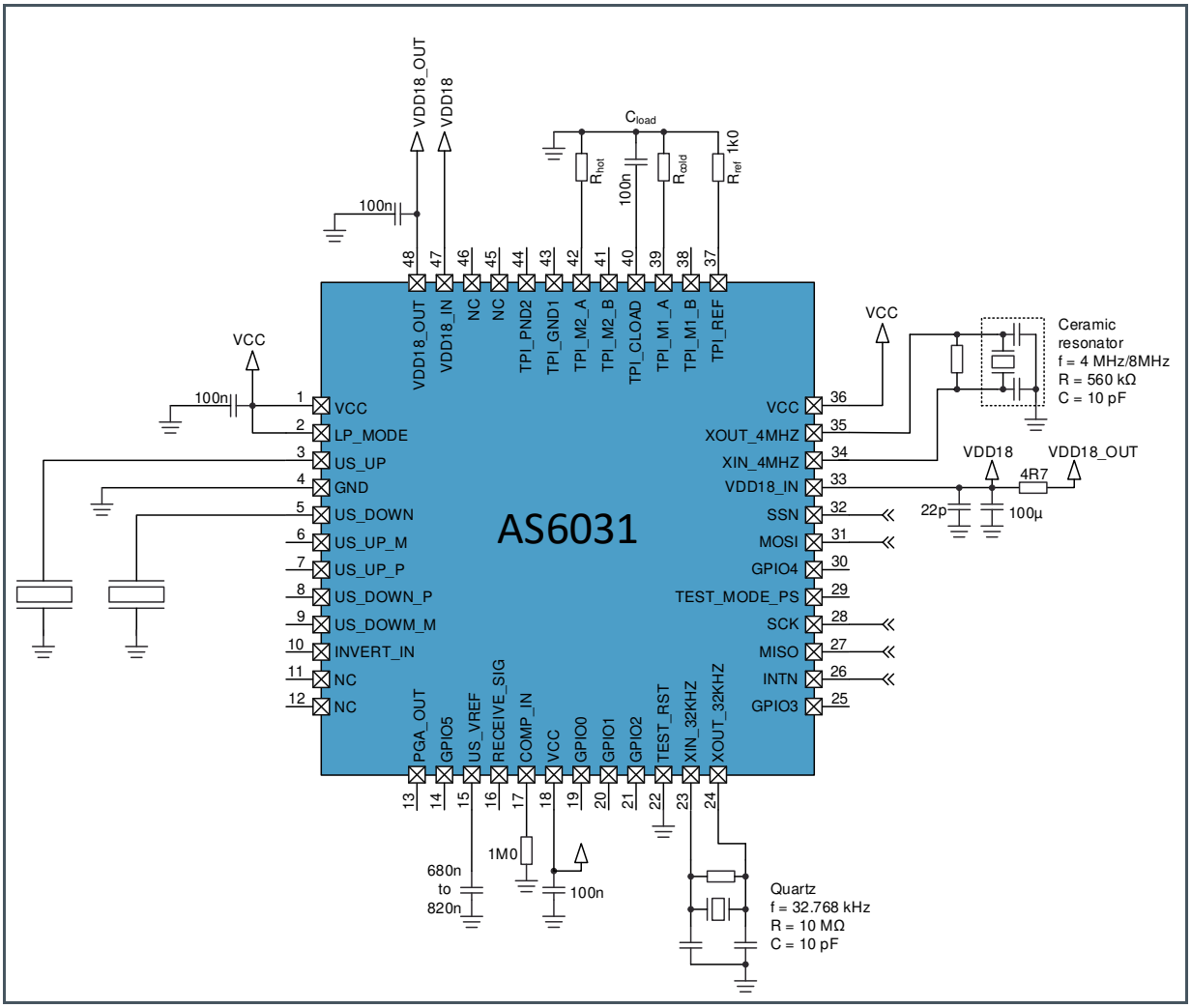
Figure 163:
Asynchronous Port Communication with CPU FW

Step	Description
1	Write up to 8 words with measurement results to address 0x1C0…0x1C7 in CPU FW
2	Setting ACP_PAGE_TGL, bit16 in SHR_EXC to 0x1 toggles between the two banks
3	The data is then available at the remote interface by accessing 0x1C0 first

13 Application Information

13.1 Schematic

Figure 164:
AS6031 Schematic



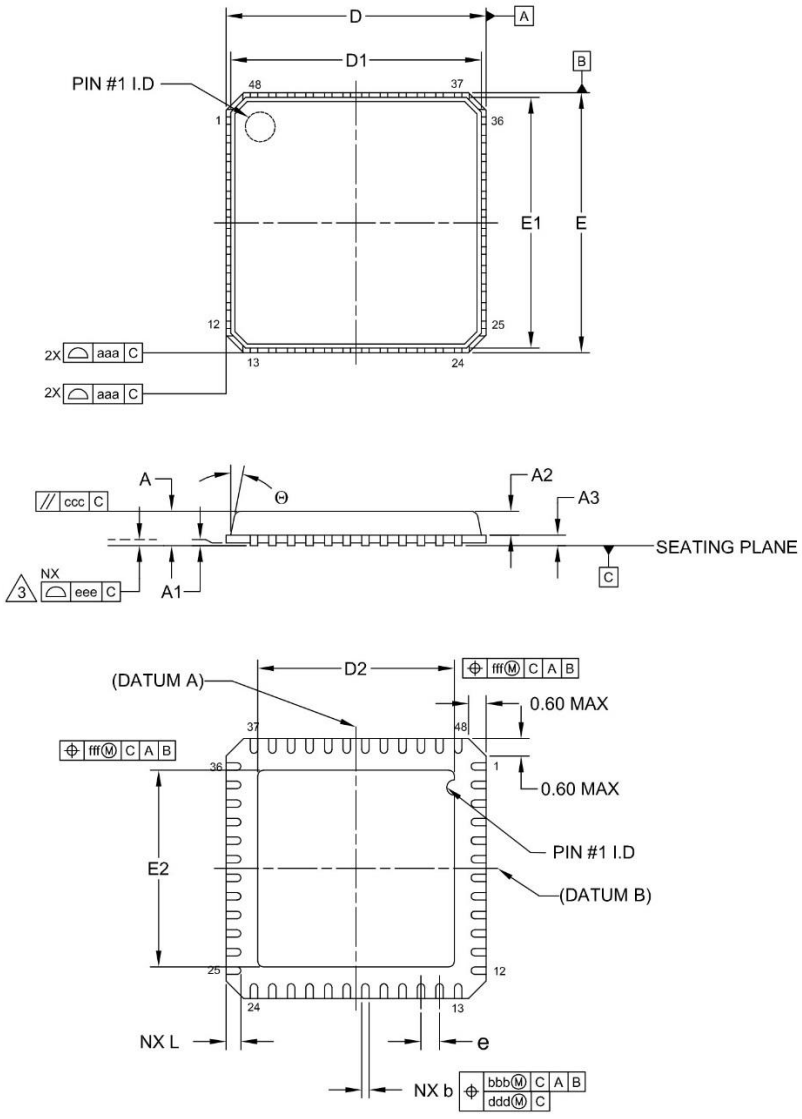
13.2 External Components

Figure 165:
AS6031 Demo Board BOM

Item	Quantity	Designator	Value	Part description
2	3	C1, C10, C11	10p	CHIP-CAPACITOR 0603
3	1	C18	22p	CHIP-CAPACITOR 0603
4	4	C3, C4, C5, C28	100n	CHIP-CAPACITOR 0603
5	1	C6	680n	CHIP-CAPACITOR 0603
6	2	C50, C51	100n	CHIP-CAPACITOR 0805
7	2	C12, C32	100u	CHIP-CAPACITOR 0805
8	1	C29	100n C0G	CHIP-CAPACITOR 1206
9	3	R1, R2, R3	4R7	CHIP-RESISTOR 0603
10	1	R16	1k	CHIP-RESISTOR 0603
11	1	R10	560k	CHIP-RESISTOR 0603
12		1R19	1M	CHIP-RESISTOR 0603
13	1	R8	10M	CHIP-RESISTOR 0603
14	1	U1		AS6031
15	1	U2	3,0V	XC6206 Torex
16	1	X1	4MHz	Ceramic resonator CSTCR4M00G53-R0 Murata
17	1	X2	32,768kHz	Quartz crystal KX-327XS Geyer
18	1	J1		Male Connector 7x1x180° 2,54
19	1	J2		Male Connector 2x1x180° 2,54
20	1	J3		Male Connector 2x1x90° 2,54

14 Package Drawings & Markings

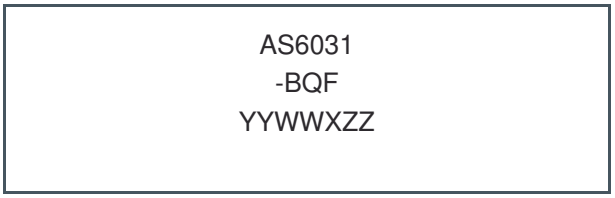
Figure 166:
QFN48 Package Outline Drawing



Ref.	Min	Nom	Max
A	0.80	0.90	1.00
A1	0	0.02	0.05
A2	-	0.65	1.00
A3		0.20 REF	
L	0.35	0.40	0.45
Θ	0°		14°
b	0.18	0.25	0.30
D		7.00 BSC	
E		7.00 BSC	
e		0.50 BSC	
D2	4.20	4.30	4.40
E2	4.20	4.30	4.40
D1		6.75 BSC	
E1		6.75 BSC	
aaa		0.15	
bbb		0.10	
ccc		0.10	
ddd		0.05	
eee		0.08	
fff		0.10	
N		48	

(1) All dimensions are in millimeters. Angles in degrees.
(2) Dimensioning and tolerancing conform to ASME Y14.5M-1994.
(3) N is the total number of terminals.
(4) This package contains no lead (Pb).
(5) This drawing is subject to change without notice.

Figure 167:
QFN48 Package Marking/Code



YY Manufacturing Year
WW Manufacturing Week
X Assembly Plant Identifier
ZZ Assembly Traceability Code
@ Sublot Identifier
xxxxxxxx Tracecode

15 Appendix

15.1 Notational Conventions

Throughout the AS6031 documentation, the following stile formats are used to support efficient reading and understanding of the documents:

- Hexadecimal numbers are denoted by a leading 0x, e.g. 0xAF = 175 as decimal number. Decimal numbers are given as usual.
- Binary numbers are denoted by a leading 0b, e.g. 0b1101 = 13. The length of a binary number can be given in bit (b) or Byte (B), and the four bytes of a 32b word are denoted B0, B1, B2 and B3 where B0 is the lowest and B3 the highest byte.
- Abbreviations and expressions which have a special or uncommon meaning within the context of AS6031 application is listed and shortly explained in the list of abbreviations, see following page. They are written in plain text. Whenever the meaning of an abbreviation or expression is unclear, please refer to the glossary at the end of this document.
- **Variable names** for hard coded registers and flags are in bold. Meaning and location of these variables is explained in the datasheet (see registers CR, SRR and SHR).
- Variable names which represent memory or code addresses are in grey. Many of these addresses have a fixed value inside the ROM code, others may be freely defined by software. Their meaning is explained in the firmware and ROM code description, and their physical addresses can be found in the header files. These variable names are defined by the header files and thus known to the assembler as soon as the header files are included in the assembler source code. Note that different variable names may have the same address, especially temporary variables.
- *Physical variables* are in italics (real times, lengths, flows or temperatures).

15.2 Abbreviations

Figure 168:
Abbreviations

Short	Description
AM	Amplitude measurement
CD	Configuration Data
CPU	Central Processing Unit
CR	Configuration Register
DIFTOF, DIFTOF_ALL	Difference of up and down → TOF
FEP	Frontend Processing

Short	Description
FDB	Frontend data buffer
FHL	First hit level (physical value V_{FHL})
FW	Firmware, embedded software stored on the chip
FWC	Firmware Code
FWD	Firmware Data
FWD-RAM	Firmware Data, volatile memory part
GPIO	General purpose input/output
Hit	Stands for a detected wave period
HSO	High speed oscillator
INIT	Initialization process of → CPU or → FEP
IO	Input/output
I2C	Inter-Integrated Circuit bus
LSO	Low speed oscillator
MRG	Measurement Rate Generator
NVRAM, NVM	Programmable Non-Volatile Memory
PI	Pulse interface
PP	Post Processing
PWR	Pulse width ratio
R	RAM address pointer of the CPU, can also stand for the addressed register
RAA	Random Access Area
RAM	Random Access Memory
RI	Remote Interface
ROM	Read Only Memory
ROM code	Hard coded routines in ROM
SHR	Special Handling Register
SPI	Serial Peripheral Interface
SRAM	Static RAM
SRR	Status & Result Register
SUMTOF	Sum of up and down TOF
Task	Process, job
TDC	Time-to-digital-converter
TOF, TOF_ALL	Time of Flight
TS	Task Sequencer
TM	Temperature measurement

Short	Description
USM	Ultrasonic measurement
V_{ref}	Reference voltage
X,Y,Z	Internal registers of the CPU
ZCD	Zero cross detection, physical level V_{ZCD}

15.3 Glossary

Figure 169: Glossary

Term	Meaning	AS6031 Interpretation
ACP	Asynchronous communication port	Register to hold date for asynchronous communication with an external microcontroller. Content needs to be filled by firmware.
AM	Amplitude measurement	This is a peak measurement of the received signal amplitude, which allows to pick the overall signal maximum (to control the signal level).
Backup	Permanent storage of a data copy	AS6031 is prepared for an external data backup, foreseen over the built-in I2C-bus, which permits write and read with an external EEPROM. In principle, a user may also utilize the → GPIOs for his own interface implementation for external backup.
Bootloader	System routine that initializes CPU operation	Typically, after a system reset, first time when the →TS calls the → CPU, the bootloader routine is called. If the → firmware is released, the bootloader loads the chip configuration from FWD into CR and does other hardware initializations like reading firmware revision numbers and calculation of checksums.
Burst	Analog signal containing a number of → wave periods	For a flow measurement, a → fire burst, that means a fixed number of → wave periods of the measurement frequency, is send over a →transducer into the flow medium. After some travel time (see →TOF), a receive burst appears at the opposed transducer, which is detected as a number of →hits. Note that the peak amplitude of the receive burst must not exceed → Vref to avoid negative voltages.
Calibration	Parameter adjustment to compensate variations	In AS6031, different calibration processes are implemented and needed for high quality measurements: → Firmware calibrations: Flow and temperature calibration, but also the → FHL adjustment are under full control of the firmware. Half-automated calibrations: → AM calibration and → HSO calibration are based on dedicated measurements, initiated by the → TS on demand. The actual calibrations need further evaluation by the firmware. Fully hard-coded calibrations: these calibrations need no interaction from firmware. One example is → ZCD level calibration, which only needs to be initiated by the → TS frequently. Another example is → TDC calibration which happens automatically before each measurement.
CD	Configuration Data	16 x (up to) 32b words of → flash memory for configuration of the chip, address range 0x16C - 0x17A (→ NVRAM). Is copied to → CR for actual usage.
Comparator	Device that compares two input signals	See → ZCD-comparator

Term	Meaning	AS6031 Interpretation
CPU	Central Processing Unit	32b processor (Harvard architecture type) for general data processing. The CPU has a fixed instruction set and acts directly on its three input- and result-registers → X,Y and Z as well as on addressed RAM. The fourth register of the CPU is the → RAM address pointer R. Instructions for the CPU are read as → FWC or → ROM code at an address given by the → program counter.
CR	Configuration Register	The chip actually uses for its hardware configuration a copy of the → CD into the CR address range 0x0C0 - 0x0CF (see → direct mapped registers).
C0G		Material of a ceramic capacitor with a very low temperature drift of capacity
DIFTOF, DIFTOF_ALL	Difference of up and down → TOF	The difference between up and down → TOF is the actual measure for flow speed. (see also → SUMTOF). DIFTOF_ALL is the DIFTOF using → TOF_ALL results, averaged over all TOF → hits
Direct mapped registers	Registers with direct hardware access	These register cells are not part of some fixed memory block, they rather have individual data access. This makes them suitable for hardware control. See → SHR, → SRR, → CR and → DR. Labels have the according prefix.
FEP	Frontend Processing	Task of the → TS where frontend measurements are performed
FDB	Frontend data buffer	Part of the → RAM where the → frontend temporarily stores its latest measurement results (→ RAA address range from 0x80 up to maximally 0x9B)
FHL, V_{FHL}	First hit level	Voltage level similar to the → ZCD level, but shifted away from Zero level, for save detection of a first → hit. The FHL determines, which of the → wave periods of the receive → burst is detected as first hit. It thus has a strong influence on → TOF and must be well controlled, in order to achieve comparable TOF measurements.
Fire, fire burst, fire buffer	Send signal → burst	The measurement signal on sending side is called fire burst, its output amplifier correspondingly fire buffer.
Firmware	Firmware	Firmware is the combination of -> Firmware Code and -> Firmware Data. A part of it is provided by ScioSense with the possibility of extended firmware programming by customer.”
Flow meter mode	Operation mode of AS6031 as full flow meter system	In flow meter mode, the AS6031 also performs further evaluation of → TOF results, to calculate physical results like flow and temperature. To do this, it uses a → firmware running on its internal CPU. See for comparison → time conversion mode
Frontend	Main measurement circuit block	This part of the AS6031 chip is the main measurement device, containing the analog measurement interface (including the → TDC). The frontend provides measurement results which are stored in the → FDB.
FWC	Firmware Code	Firmware code denotes the complete content of the → NVRAM’s 4kB section (address range 0x0000 to 0x 0FFF). The difference to the term → firmware is on the one hand that firmware code means the program in the file. On the other hand, a particular firmware code may provide just a part of the complete FWC. FWC is addressed by the CPU’s program counter, it is not available for direct read processes like RAM. Firmware code by user is limited to the address range 32 to FWU_RNG.
FWD	Firmware Data	The firmware configuration and calibration data, to be written to the → FWD-RAM
FWD-RAM	Firmware Data memory	128 x 32b words of → NVRAM (built as volatile → SRAM and non-volatile flash memory). Main purpose is calibration and configuration
GPIO	General purpose input/output	AS6031 has up to 6 GPIO pins which can be configured by the user. Some of them can be configured as → PI or → I2C-interface.

Term	Meaning	AS6031 Interpretation
Hit	Stands for a detected wave period	The receive → burst is typically a signal which starts with → wave periods of the measurement frequency at increasing signal levels. While the first of these wave periods are too close to noise for a reliable detection, later signal wave periods with high level can be detected safely by the → ZCD-comparator. The comparator converts the analog input signal into a digital signal, which is a sequence of hits. To detect the first hit at an increased signal level, away from noise, the input signal is compared to the → FHL. After the first hit, the level for comparison is immediately reduced to the → ZCD level, such that all later hits are detected at zero crossing (note that the ZCD level is defined to zero with respect to the receive signal, it is actually close to → Vref or another user-defined level). Different hits are denoted according to their usage: • Hit (in general) stands for any detected → wave period. • First hit is actually the first hit in a → TOF measurement (not the first wave period!) • TOF hits means all hits which are evaluated for → TOF measurements. Note that typically the first hit is not a TOF hit. • Start hit is the initial TOF hit. This is typically not the first hit, but (according to configuration) some well-defined later hit. Minimum the 3rd hit has to set as Start hit. • Last TOF hit. It is also defined by configuration and should not be too close to the end of the receive → burst. • Ignored hits are all hits which are not evaluated for the TOF measurement: All hits between first hit and start hit, as well any hit between TOF hits or after the stop hit.
HSO	High speed oscillator	The 4 or 8 MHz oscillator of the AS6031. In usual operation only switched on when needed, to reduce energy consumption. This is the time base for → TDC measurements. The HSO is typically less accurate that the → LSO. It should be frequently → calibrated against the LSO to obtain the desired absolute accuracy of the → TDC.
INIT	Initialization process of → CPU or → FEP	In AS6031 terminology, INIT processes don't reset registers or digital IOs, while → reset does at least one of it. Several different INIT processes are implemented, see chapter "Reset hierarchy" for details.
IO	Input/output	Connections to the outside world for input or output
I2C	Inter-integrated circuit bus	Standard serial bus for communication with external chips.
LSO	Low speed oscillator	The 32768 Hz crystal oscillator of the AS6031. This oscillator controls the main timing functions (→ MRG and → TS, real time clock).
MRG	Measurement rate generator	The measurement rate generator controls the cyclic → tasks of AS6031 by setting task requests in a rate defined by configuration (→ CR). When the MRG is activated, it periodically triggers the → TS for initiating the actual → tasks.
NVRAM, NVM	Programmable Non-Volatile Memory	AS6031 contains two sections of programmable non-volatile memory: One section of 4kB → FWC memory, and another of → FWD-RAM (FWD1:→ RAM addresses 0x100 - 0x11F and FWD2: RAM addresses 0x120 – 0x17F), in total 128 x 32b words. It is organized as a volatile SRAM part which is directly accessed from outside, and a non-volatile flash memory part.
PI	Pulse interface	Standard 2-wire interface for flow output of a water meter. Typically outputs one pulse per some fixed water volume (e.g. one pulse per 0.1 l), while the other wire signals the flow direction. Permits stand-alone operation and is fully compatible to mechanical water meters.
PP	Post Processing	Processing activities of the → CPU, typically after frontend processing (e.g. a measurement) , initiated by –>TS. Can be split for post processing related to a flow measurement and to a temperature measurement
Program code	Program	Program Code is the combination of -> Firmware Code and -> ROM Code. Program Code is addressed by CPU's -> program counter."

Term	Meaning	AS6031 Interpretation
Program counter	Pointer to the current code address of the → CPU	The program counter addresses the currently evaluated → FWC or → ROM-code cell during → CPU operation The program counter always starts at 0xF000, when any CPU action is requested.
PWR	Pulse width ratio	Width of the pulse the first → hit, related to the pulse width at the start hit. This width indicates the position of the → FHL relative to the level of the detected → wave period and thus gives some information on detection safety (small value means FHL is close to the peak amplitude and the desired wave period may be missed due to noise; large value indicates the danger that an earlier wave period may reach FHL level and trigger the first hit before the desired wave period).
R	RAM address pointer of the CPU	The → CPU acts on the data of the → X-,Y- and Z-register and on one single RAM cell. The pointer R defines the address of the current RAM cell.
RAA	Random Access Area	Address range from 0x000 to 0x1FF covering the → RAM addresses. Memory cells within this address range can all be read, most of them can also be written (except → SRR and → DR). The RAA covers memory cells of different technology: → RAM (including → FDB), → FWD-RAM (including → CD), → direct mapped registers (→ SHR, → SRR, → CR). Holds also the asynchronous communication port registers (→ACP).
RAM	Random Access Memory	176 x 32b words of volatile memory, used by → FDB and → Firmware. Address range 0x000 to 0x0AF
RAM address	Address of a cell in the RAA range	A RAM address is used by the firmware or over → RI to point to a memory cell for data storage or retrieval. Note that RAM addresses cover not only actual RAM, but all cells in the RAA range. Address range from 0x000 to 0x1FF
Register	Memory cell for dedicated data storage	Memory cells are typically called register when they contain flags or configuration bits, or when they have a single dedicated purpose (see → CPU, → CR, → SHR and → SRR).
Reset	Reset of the chip	AS6031 has different processes and commands that can call resets and initializations at different levels. Some of them refresh → CR or GPIO state, others just (re-) initialize CPU or frontend. The latter are rather denoted → INIT. See chapter “Reset hierarchy” for details.
RI	Remote Interface	Interface for communication with a remote controller (see → SPI)
ROM	Read Only Memory	4kB of fixed memory, contains hard coded routines for general purpose and parts of **ScioSense**’ → program (ROM code). Address range 0xF000 – 0xFFFF. The ROM code is addressed by the CPU’s program counter, it is not available for direct read processes like RAM.
ROM code	Hard coded routines in ROM	See → ROM.
SCL	Serial Clock	Serial clock of I2C interface
SDA	Serial Data	Serial data of I2C interface
SHR	Special Handling Register	Registers that directly control chip operation. The data & flags of special handling registers have a dynamic character. They are typically updated by post processing, but some of them have to be initially configured before measurement starts.
SPI	Serial Peripheral Interface	Standard interface for communication of the AS6031 with an external master controller
SRAM	Static RAM	AS6031 does not use any dynamic RAM, in fact all RAM in AS6031 is static RAM. However, the term “SRAM” is in particular used for the RAM-part of the → NVRAM.
SRR	Status & Result Register	The SRR-registers describe the current state of the chip. They are set by the chip hardware and contain error and other condition flags, timing information and so on.

Term	Meaning	AS6031 Interpretation
SUMTOF, SUMTOF_ALL	Sum of up and down TOF	The sum of up and down → TOF is a measure for the speed of sound in the medium, which can be used for temperature calculation. SUMTOF_ALL is the SUMTOF using → TOF_ALL results, averaged over all TOF → hits.
Supervisor	Functional block of AS6031 that controls voltage and timing	The supervisor of AS6031 controls chip operation and timing through the measurement rate generator (→ MRG) and the task sequencer (–>TS). It also covers voltage control and adjustment functions as well as the main oscillators → LSO and >HSO
Task	Process, job	The term task is used for a process which aims at fulfilling some fixed purpose, separate from other tasks with different goals. Typical tasks in AS6031 are → TOF measurement, temperature measurement (→ TM), post processing (→ PP), remote communication and voltage measurement.
Time conversion mode	Remotely controlled operation of AS6031	In time conversion mode, the AS6031 mainly acts as a → TOF measurement system. It may operate self-controlled or remotely controlled, but it does no further result evaluation. This operation mode is similar to the typical usage of the **ScioSense** chips GP21 and GP22. For comparison see → Flow meter mode
TDC	Time-to-digital-converter	The core measurement device of AS6031. Measures times between a start- and a stop-signal at high accuracy and high resolution. The internal fast time base of the TDC is automatically → calibrated against the → HSO before each measurement.
TOF, TOF_ALL	Time of Flight	Basic measurement result for an ultrasonic flow meter: The time between send and receive → burst (with some offset, depending on → hit detection). Measurements of TOF are done in flow direction (down TOF) and in the opposite direction (up TOF). AS6031 also provides the sum of all TOF → hits in the values TOF_ALL.
TS	Task Sequencer	The task sequencer arranges and initiates the → tasks which are requested by the → MRG in one measurement cycle or which are initiated remotely.
TM	Temperature measurement	This task means a temperature measurement using sensors, in contrast to temperatures which are calculated results from a TOF measurement (see → SUMTOF)
Transducer	Electromechanical conversion device	Transducers for flow measurements are piezoelectric devices that convert an electrical signal into ultrasound and reverse. They are usually matched to the flow medium (e.g. water). AS6031 can connect directly to the send and receive transducer.
USM	Ultrasonic measurement	The principle of an ultrasonic flow meter is to measure → TOFs of ultrasound in flow direction and against it, and to calculate the flow from the result. See also → transducer. In this manual this includes also the amplitude measurement.
V_{ref}	Reference voltage	The analog interface of AS6031 refers to V_{ref}, a nominal voltage for → VZCD of typically 0.7V. This makes it possible to receive a DC-free AC-signal with a single supply voltage. Up to the level of V_{ref}, negative swings of the receive signal are avoided.
V_{ZCD}	Zero cross detection level	This voltage level represents the virtual zero line for the receive → burst. It is normally close to → Vref, just differing by the offset of the → ZCD-comparator. Needs frequent → calibration to compensate the slowly changing offset. Optionally, this voltage can be configured differently in SHR_ZCD… through the firmware.
Watchdog, watchdog clear	Reset timer for chip re-initialization	The watchdog of AS6031 → resets the chip (including → CR refresh) if no watchdog clear (→ firmware command clrwdt) within 15.2 s (typically) is executed. This is a safety function to interrupt hang-up situations. It can be disabled for remote control, when no firmware clears the watchdog automatically.

Term	Meaning	AS6031 Interpretation
Wave period	One period of the signal wave	A period of typically 1us length for a 1 MHz measurement frequency. This may be a digital pulse, for example when sending, or a more sinusoidal wave when receiving. Fire or receive → bursts are sequences of wave periods.
X-, Y- and Z-register	Input- and result registers of the CPU	The → CPU acts on these → registers for data input and result output.
ZCD	Zero cross detection	All → hits following the first hit are detected when the received signal crosses a voltage level VZCD, defined as zero with respect to the receive → burst. In contrast, the first hit is detected when the received signal crosses the different voltage level VFHL(→ FHL).
ZCD-Comparator	→ comparator for → hit detection	The ZCD-comparator in AS6031 detects → hits in the received → burst signal by comparing the received signal level to a given reference voltage (see also → FHL, → ZCD and → hit).

15.4 CPU Commands in Detail

The following description lists every instruction which is recognized by the assembler. Most of them directly correspond to an op-code, which is a sequence of bytes in an executable code for AS6031, as it is produced by the assembler. The tabular lines have the following meaning and usage:

Figure 170:
Structure

Command	Short Description
Syntax:	Command name, followed by parameters p1, p2, p3…
Parameters:	Description of parameters. They may be registers REG [x, y, z, r] or numbers in a given range.
Calculus:	Mathematical operation in Verilog notation (uncommon syntax is explained in case). This line also defines the result output, which is most of the time simply p1. Note that this means that the content of p1 is changed by the operation.
Flags affected:	Some or all of the flags C (carry), Z (Zero), S (sign) and O (Overflow) are affected by the described operation, according to the result
Bytes:	Length of the complete op-code, including parameter designation
Cycles:	Number of calculation cycles needed by the CPU
Description:	Literal description and remarks on the operation
Category:	One of the categories in the overview

There are some more expressions used in the list:

- PC: The program counter; this is actually the code address where the next CPU op-code is read.
- JUMPLABEL: Label for a jump destination, which becomes an actual code address after code assembly. In assembler code, this is usually a placeholder for a position within the code, it may also be a fixed number (not recommended). To define a jump destination by a jump label in assembler code, write the label followed by a colon.
- LSB: Least significant bit, the rightmost bit of a binary number
- MSB: Most significant bit, the leftmost bit of a binary number. In the common two's complement representation, the MSB is used to indicate the sign of a number; MSB = 1 defines a negative number.
- ">>" or "<<": right shift and left shift, e.g. "1<<p2": a 1 shifted left by p2 bit positions

In the following, there is a list of all CPU instructions in alphabetic order.

abs	Absolute value of register
Syntax:	abs p1
Parameters:	p1 = REG [x, y, z, r]
Calculus:	p1 = | p1 |
Flags affected:	C O Z S
Bytes:	2
Cycles:	2
Description:	Absolute value of register
Category:	Simple arithmetic

add	Addition
Syntax:	add p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-Bit number
Calculus:	p1 = p1 + p2
Flags affected:	C O Z S
Bytes:	1 (p2 = REG) 5 (p2 = number)
Cycles:	1 (p2 = REG) 5 (p2 = number)
Description:	Addition of two registers or addition of a constant to a register
Category:	Simple arithmetic

and	Logic AND
Syntax:	and p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-Bit number
Calculus:	p1 = p1 AND p2 *in the resulting bit sequence in p1, a bit is 1 when the corresponding bits of P1 and P2 are both equal to 1*
Flags affected:	Z S

and	Logic AND
Bytes:	2 (p2 = REG) 6 (p2 = number)
Cycles:	3 (p2 = REG) 7 (p2 = number)
Description:	Bitwise logic AND of 2 registers or Logic AND of register and constant
Category:	Logic

bitclr	Clear single bit
Syntax:	bitclr p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = number 0 to 31
Calculus:	p1 = p1 and not (1<<p2) *"1<<p2": a "1" shifted left by p2 bit positions*
Flags affected:	Z S
Bytes:	2
Cycles:	2
Description:	Clear the single bit on position p2 in the destination register p1, other bits remain unchanged Note: Don't use on register R in combination with bytesel ≠ 0
Category:	Bitwise

bitinv	Invert single bit
Syntax:	bitinv p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = number 0 to 31
Calculus:	p1 = p1 XOR (1<<p2) *"1<<p2": a "1" shifted left by p2 bit positions*
Flags affected:	Z S
Bytes:	2
Cycles:	2
Description:	Invert the single bit on position 1<<p2 in the destination register p1, other bits remain unchanged Note: Don't use on register R in combination with bytesel ≠ 0
Category:	Bitwise

bitset	Set single bit
Syntax:	bitset p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = number 0 to 31
Calculus:	p1 = p1 OR (1<<p2) *"1<<p2": a "1" shifted left by p2 bit positions*
Flags affected:	Z S
Bytes:	2
Cycles:	2
Description:	Set the single bit on position p2 in the destination register p1, other bits remain unchanged Note: Don't use on register R in combination with bytesel ≠ 0

bitset	Set single bit
Category:	Bitwise

bytedir	Define configuration for bytesel
Syntax:	bytedir p1
Parameters:	p1 = number 0 or 1
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Basic definition for the configuration of bytesel (see description of bytesel) p1 = 0 : align read data at LSB p1 = 1 : shift read byte(s) to various positions Important remarks: Bytedir permanently sets the read configuration until it is changed. Bytedir is not affected by any conditional or unconditional skip command. Usage within the range of any skip command is not permitted.
Category:	RAM access

bytesel	Define RAM reading mode
Syntax:	bytesel p1
Parameters:	p1 = number 0 to 7
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1

bytesel	Define RAM reading mode
Description:	Read from addressed register R using a byte-oriented shift operation in various configurations. The bytesel command is implemented to simplify bytewise operations, for example read and write from an external EEPROM, or internal selection of coefficients which are shorter than 32 Bit. It actually provides a means for fast bytewise shifting. Denoting the four bytes in the 32-Bit word in R by B3/B2/B1/B0, the following content of R is actually read, depending on the last bytedir setting: After “bytedir 0” (or without using bytedir): p1 = 0 : R is read as B3/B2/B1/B0 (default setting, no shifts) p1 = 1 : R is read as 00/00/B2/B1 p1 = 2 : R is read as 00/00/B1/B0 p1 = 3 : R is read as 00/00/B3/B2 p1 = 4 : R is read as 00/00/00/B0 p1 = 5 : R is read as 00/00/00/B1 p1 = 6 : R is read as 00/00/00/B2 p1 = 7 : R is read as 00/00/00/B3 After “bytedir 1”: p1 = 0 : R is read as B3/B2/B1/B0 (default setting, no shifts) p1 = 1 : R is read as 00/B1/B0/00 p1 = 2 : R is read as 00/00/B1/B0 p1 = 3 : R is read as B1/B0/00/00 p1 = 4 : R is read as 00/00/00/B0 p1 = 5 : R is read as 00/00/B0/00 p1 = 6 : R is read as 00/B0/00/00 p1 = 7 : R is read as B0/00/00/00 Important remarks: Bytesel affects the read direction for any register addressed as R. Any read access to R is affected, so the content of R for any operation is configured according to the list above. Bytesel has no effect in write direction. Bytesel permanently sets the read configuration until it is changed. Bytesel is not affected by any conditional or unconditional skip command. Usage within the range of any skip command is not recommended. Note that the commands bitset, bitclr or bitinv and shiftL, shiftR, rotL and rotR include read access. Set bytesel = 0 before applying one of these commands to R, to avoid undefined results.
Category:	RAM access

clear	Clear register
Syntax:	clear p1
Parameters:	p1 = REG [x, y, z, r]
Calculus:	p1 = 0
Flags affected:	Z S
Bytes:	1
Cycles:	1
Description:	Clear addressed register to 0
Category:	Register wise

clkmode	Clock mode
Syntax:	clkmode p1
Parameters:	p1 = number 0 or 1
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	2
Description:	p1 = 0 : CPU clock is the internal oscillator p1 = 1 : CPU clock is 2 MHz, derived from the high speed clock (HSC) Remark: clkmode sets the clock mode permanently until the next change or until stop. After stop or after power up, clkmode is 0.
Category:	Miscellaneous

clrC	Clear flags
Syntax:	clrC
Parameters:	-
Calculus:	-
Flags affected:	C O
Bytes:	2
Cycles:	2
Description:	Clear Carry and Overflow flags
Category:	Flags

clrwdt	Clear watchdog
Syntax:	clrwdt
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	
Description:	Clear watchdog. This instruction is used to restart the watchdog timer at the end of a program run. Apply clrwdt right before ‚stop' to avoid a reset by the watchdog, if enabled.
Category:	Miscellaneous

compare	Compare two values
Syntax:	compare p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-Bit number
Calculus:	no register change; only the flags are set to the result of the operation p2 - p1
Flags affected:	C O Z S

compare	Compare two values
Bytes:	1 (p1 = REG, p2 = REG) 5 (p1 = REG, p2 = number)
Cycles:	1 (p1 = REG, p2 = REG) 5 (p1 = REG, p2 = number)
Description:	Comparison of the two inputs by subtraction. The flags are changed according to the subtraction result, but not the register contents themselves.
Category:	Simple arithmetic

compl	Complement
Syntax:	compl p1
Parameters:	p1 = REG [x, y, z, r]
Calculus:	p1 = - p1 = (NOT p1) + 1
Flags affected:	Z S
Bytes:	2
Cycles:	2
Description:	two‘s complement of register
Category:	Simple arithmetic

decr	Decrement
Syntax:	decr p1
Parameters:	p1 = REG [x, y, z, r]
Calculus:	p1 = p1 – 1
Flags affected:	C O Z S
Bytes:	1
Cycles:	1
Description:	Decrement register by 1
Category:	Simple arithmetic

decramadr	Decrement RAM address pointer
Syntax:	decramadr
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Decrement RAM address pointer by one
Category:	RAM access

div	Signed division 32 Bit
Syntax:	div p1, p2

div	Signed division 32 Bit
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r]
Calculus:	p1 = (p1 << 32) / p2 *"p1<<32": p1 shifted left by 32 bit positions* *or* p1 = p1*232/ p2 *in standard notation* condition for correct calculation: |p1| < |2*p2| In consequence, the result integers in p1 are between $-0.5*2^{32}$ and $0.5*2^{32}$
Flags affected:	Z and S according to the result in p1
Bytes:	2
Cycles:	38
Description:	Signed division of 2 registers: 32 fractional bits of the division of 2 registers are assigned to p1; p2 remains unchanged
Category:	Complex arithmetic

divmod	Signed modulo division
Syntax:	divmod p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r]
Calculus:	p1 = integer (p1 / p2) p2 = p1 % p2 *"%" is the modulo operation*
Flags affected:	Z and S according to the result in p1
Bytes:	2
Cycles:	Similar to div
Description:	Signed modulo division of 2 registers, 32 higher bits of the integer division of 2 registers, result is assigned to p1; the remainder is assigned to p2
Category:	Complex arithmetic

eor	Exclusive OR
Syntax:	eor p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-Bit number
Calculus:	p1 = p1 XOR p2 *in the resulting bit sequence in p1, a bit is 0 when the corresponding bits of P1 and P2 are equal, or 1 otherwise*
Flags affected:	Z S
Bytes:	2 (p1 = REG, p2 = REG) 6 (p1 = REG, p2 = number)
Cycles:	3 (p1 = REG, p2 = REG) 7 (p1 = REG, p2 = number)
Description:	Bitwise Logic exclusive OR (antivalence) of the two given parameters
Category:	Logic

eorn	Exclusive NOR
Syntax:	eorn p1, p2

eorn	Exclusive NOR
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-Bit number
Calculus:	p1 = p1 XNOR p2 *in the resulting bit sequence in p1, a bit is 1 when the corresponding bits of P1 and P2 are equal, or 0 otherwise*
Flags affected:	Z S
Bytes:	2 (p1 = REG, p2 = REG) 6 (p1 = REG, p2 = number)
Cycles:	3 (p1 = REG, p2 = REG) 7 (p1 = REG, p2 = number)
Description:	Bitwise Logic, exclusive not OR (equivalence) of the two given parameters
Category:	Logic

equal	Write 3 given Bytes to the executable code
Syntax:	equal p1
Parameters:	p1 = 3-Byte string
Calculus:	-
Flags affected:	-
Bytes:	3
Cycles:	3 (or more if an executable command was written)
Description:	This instruction is recognized by the assembler. It writes exactly the three bytes given in p1 to the executable code. This can be used to add customized information like version numbers. Handle with care, since the bytes will be interpreted as code when the PC points to them.
Category:	Miscellaneous

equal1	Write 1 given Bytes to the executable code
Syntax:	equal1 p1
Parameters:	p1 = Byte string
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1 (or more if an executable command was written)
Description:	This instruction is recognized only by the assembler. It writes exactly the one byte given in p1 to the executable code. This can be used to add customized information like version numbers. Handle with care, since the byte will be interpreted as code when the PC points to it.
Category:	Miscellaneous

getflag	Set S and Z flags
Syntax:	getflag p1
Parameters:	p1 = REG [x, y, z, r]
Calculus:	Signum flag S is set if p1 < 0 Zero flag Z indicates Zero if p1 = 0
Flags affected:	Z S

getflag	Set S and Z flags
Bytes:	1
Cycles:	1
Description:	Set the signum and zero flag according to the addressed register, content of the register is not affected
Category:	Simple arithmetic

getramadr	Set RAM address pointer to the value in Z
Syntax:	getramadr
Parameters:	- *The input address is always taken from Z*
Calculus:	RAM address pointer = Z
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Set the RAM address pointer to the value given in Z
Category:	RAM access

goto	jump without condition
Syntax:	goto p1
Parameters:	p1 = JUMPLABEL
Calculus:	PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump without condition. Program counter (PC) is set to target address. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoBitC	Jump on bit clear
Syntax:	gotoBitC p1, p2, p3
Parameters:	p1 = REG [x, y, z, r] p2 = number [0...31] p3 = JUMPLABEL or number
Calculus:	if (bit p2 of register p1 == 0) *if ((1<<p2 and p1) == 0)* PC = p3
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)

gotoBitC	Jump on bit clear
Description:	Jump on bit clear. Program counter (PC) is set to target address if selected bit p2 in register p1 is clear. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoBitS	Jump on bit set
Syntax:	gotoBitS p1, p2, p3
Parameters:	p1 = REG [x, y, z, r] p2 = number [0..31] p3 = JUMPLABEL or number
Calculus:	if (bit p2 of register p1 == 1) *if ((2^{p2} AND p1) == 1) …* PC = p3
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump on bit set. Program counter (PC) is set to target address if selected bit p2 in register p1 is set. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoCarC	Jump on carry clear
Syntax:	gotoCarC p1
Parameters:	p1 = JUMPLABEL or number
Calculus:	if (carry is clear) PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump on carry clear. Program counter (PC) is set to target address if the last operation that affected the carry (C) flag left it clear. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoCarS	Jump on carry set
Syntax:	gotoCarS p1
Parameters:	p1 = JUMPLABEL or number
Calculus:	if (carry is set) PC = p1
Flags affected:	-

gotoCarS	Jump on carry set
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump on carry set. Program counter (PC) is set to target address if the last operation that affected the carry (C) flag left it set. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoEQ	Jump on equal zero
Syntax:	gotoEQ p1
Parameters:	p1 = JUMPLABEL or number
Calculus:	if (Z indicates zero) PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions*" 4 (absolute jump)
Description:	Jump on equal zero. Program counter (PC) is set to target address if the last operation that affected the zero (Z) flag indicated a zero result. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoNE	Jump on not equal zero
Syntax:	gotoNE p1
Parameters:	p1 = JUMPLABEL or number
Calculus:	if (Z indicates not-equal zero) PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump on not-equal zero. Program counter (PC) is set to target address if the last operation that affected the zero (Z) flag indicated a not-equal zero result. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoNeg	Jump on negative
Syntax:	gotoNeg p1
Parameters:	p1 = JUMPLABEL or number

gotoNeg	Jump on negative
Calculus:	if (S indicates negative) PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump on negative. Program counter (PC) is set to target address if the last operation that affected the sign (S) flag indicated a result below 0. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoOvrC	Jump on overflow clear
Syntax:	gotoOvrC p1
Parameters:	p1 = JUMPLABEL or number
Calculus:	if (O is clear) PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump on overflow clear. Program counter (PC) is set to target address if the last operation that affected the overflow (O) flag indicated no overflow. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoOvrS	Jump on overflow set
Syntax:	gotoOvrS p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (O is set) PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump on overflow set. Program counter (PC) is set to target address if the last operation that affected the overflow (O) flag indicated an overflow. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

gotoPos	Jump on positive
Syntax:	gotoPos p1
Parameters:	p1 = JUMPLABEL or number
Calculus:	if (S indicates positive) PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump on positive. Program counter (PC) is set to target address if the last operation that affected the sign (S) flag indicated a result equal or above 0. The target address is given by using a jump label or by an absolute number. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jump

The following three I2C instructions are very basic and listed for the sake of completeness only. The user is asked to access the ready-made ROM routines as described in section 15.4.

i2crw	I2C read / write
Syntax:	i2crw p1
Parameters:	p1 = number 0 or 1
Description:	p1 = 0 : I2C write p1 = 1 : I2C read

incr	Increment
Syntax:	incr p1
Parameters:	p1 = REG [x, y, z, r]
Calculus:	p1 = p1 + 1
Flags affected:	C O Z S
Bytes:	1
Cycles:	1
Description:	Increment register by one
Category:	Simple arithmetic

incramadr	Increment RAM address
Syntax:	incramadr
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Increment RAM address pointer by 1

incramadr	Increment RAM address
Category:	RAM access

invert	Bitwise inversion
Syntax:	invert p1
Parameters:	p1 = REG [x, y, z, r]
Calculus:	p1 = NOT p1
Flags affected:	Z S
Bytes:	2
Cycles:	2
Description:	Bitwise inversion of register
Category:	Logic

jsub	Unconditional jump to a subroutine
Syntax:	jsub p1
Parameters:	p1 = JUMPLABEL or number
Calculus:	PC = p1
Flags affected:	-
Bytes:	2 (relative jump) *see section "branch instructions"* 3 (absolute jump)
Cycles:	3 (relative jump) *see section "branch instructions"* 4 (absolute jump)
Description:	Jump to subroutine without condition. The program counter is loaded by the address given through the parameter. The subroutine is processed until the keyword ‚jsubret' occurs. Then a jump back is performed and the next command after the jsub instruction is executed. Jsub needs temporarily a place in the program counter (PC) stack to remember the return address. The PC stack has a depth of 8, so jsub works for up to 8 nested calls. Jump range: 0 to 4095 (Firmware code) and 61440 to 65535 (ROM code)
Category:	Jsub

jsubret	Return from subroutine
Syntax:	jsubret
Parameters:	-
Calculus:	PC = PC after last jsub operation
Flags affected:	-
Bytes:	1
Cycles:	3
Description:	Return from subroutine. A subroutine called via ‚jsub' has to be exited by using jsubret. The program is continued at the next command following the calling jsub instruction. The address for continuing is stored in the program counter (PC) stack, which has a depth of 8. This means, the combination jsub-jsubret can be used for up to 8 nested calls.
Category:	Jsub

mcten	Enable / disable measure cycle timer
Syntax:	mcten p1
Parameters:	p1 = number 0 or 1
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	2
Description:	p1 = 0 : Measure cycle timer disabled p1 = 1 : Measure cycle timer enabled
Category:	Miscellaneous

Move	Move
Syntax:	move p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-bit number
Calculus:	p1 = p2
Flags affected:	Z S
Bytes:	1 (p1 = REG, p2 = REG) 5 (p1 = REG, p2 = number)
Cycles:	1 (p1 = REG, p2 = REG) 5 (p1 = REG, p2 = number)
Description:	Move content of p2 to p1 (p1 = REG, p2 = REG) Move constant to p1 (p1 = REG, p2 = number)
Category:	Register wise

mult	Signed 32-Bit multiplication
Syntax:	mult p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r]
Calculus:	p1, p2 = p1 * p2 *the 32-bit numbers p1 and p2 are multiplied to a 64-bit result which is stored in p1 (upper 32 bits and sign) and p2 (lower 32 bits)*
Flags affected:	Z and S according to p1
Bytes:	2
Cycles:	38
Description:	Signed multiplication of two registers. Higher 32 bits of the multiplication result are placed to p1; lower 32 bits of the multiplication result are placed to p2. Note that the sign of the whole number is defined through the MSB of p1, while the MSB of p2 is just bit 31 of the result (p2 is unsigned). This can lead to misinterpretation by subsequent operations which assume signed numbers.
Category:	Complex arithmetic

nand	Logic NAND
Syntax:	nand p1, p2

nand	Logic NAND
Parameters:	p1 = REG [x, y, z, r] p1 = REG [x, y, z, r] or 32-Bit number
Calculus:	p1 = p1 NAND p2 *not (p1 AND p2); in the resulting bit sequence in p1, a bit is 0 when the corresponding bits of P1 and P2 are both equal to 1, otherwise the bit is 1*
Flags affected:	Z S
Bytes:	2 (p1 = REG, p2 = REG) 6 (p1 = REG, p2 = number)
Cycles:	3 (p1 = REG, p2 = REG) 7 (p1 = REG, p2 = number)
Description:	Bitwise logic NAND (negated AND) of the two input parameters
Category:	Logic

nop	No operation
Syntax:	nop
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Placeholder code or timing adjust, no operation. May be needed sometimes to separate two code bytes to prevent an assembler error message.
Category:	Miscellaneous

nor	Logic NOR
Syntax:	nor p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-Bit number
Calculus:	p1 = p1 NOR p2 *p1 = not (p1 OR p2); in the resulting bit sequence in p1, a bit is 1 when the corresponding bits of P1 and P2 are both equal to 0, otherwise the bit is 0*
Flags affected:	Z S
Bytes:	2 (p1 = REG, p2 = REG) 6 (p1 = REG, p2 = number)
Cycles:	3 (p1 = REG, p2 = REG) 7 (p1 = REG, p2 = number)
Description:	Bitwise logic NOR (negated OR) of the two input parameters
Category:	Logic

or	Logic OR
Syntax:	or p1, p2

or	Logic OR
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-Bit number
Calculus:	p1 = p1 OR p2 *in the resulting bit sequence in p1, a bit is 0 when the corresponding bits of P1 and P2 are both equal to 0, otherwise 1*
Flags affected:	Z S
Bytes:	2 (p1 = REG, p2 = REG) 6 (p1 = REG, p2 = number)
Cycles:	3 (p1 = REG, p2 = REG) 7 (p1 = REG, p2 = number)
Description:	Bitwise logic OR of the two input parameters
Category:	Logic

ramadr	Set RAM address pointer
Syntax:	ramadr p1
Parameters:	p1 = RAM cell name or 8-Bit number
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	2
Description:	Set pointer to RAM address (range: 0...255)
Category:	RAM access

rotL	Rotate left
Syntax:	rotL p1(, p2)
Parameters:	p1 = REG [x, y, z, r] p2 = no entry or number 2...15
Calculus:	case rotL p1, without p2: p1 = (p1 << 1) + carry ; carry = MSB(p1) p1 = 2*p1 + carry ; carry = MSB(p1) case rotL p1, p2: p1 = repeat (p2 times) rotL p1 *Adding carry finally lets the bits of p1 circulate left over 1 or p2 positions.*
Flag affected:	C O (resulting from the last rot step), Z S (according to the final result in p1)
Bytes:	1 (p1 = REG, p2 = none) 2 (p1 = REG, p2 = number)
Cycles:	1 (p1 = REG, p2 = none) 1 + p2 (p1 = REG, p2 = number)
Description:	Without p2 or p2 = 1 : Rotate p1 left by one bit position over carry. This means in detail, shift p1 register to the left, fill LSB with present carry, then move the former MSB to carry. With 2 ≤ p2 ≤ 15 : Rotate p1 left by p2 bit positions over carry. This means in detail, shift p1 register p2 times to the left, in each step fill LSB with the present carry and then move the former MSB to carry. Note: Don’t use on register R in combination with bytesel ≠ 0
Category:	Shift and rotate

rotR	Rotate right
Syntax:	rotR p1(, p2)
Parameters:	p1 = REG [x, y, z, r] p2 = no entry or number 2...15
Calculus:	case rotR p1, without p2: p1 = (p1 >> 1) + (carry << 31) ; carry = LSB(p1) → Carry is shifted left to position 31, or p1 = integer (p1 / 2) + (carry*2^{31}) ; carry = LSB(p1) case rotR p1, p2: p1 = repeat (p2 times) rotL p1 *Placing carry at MSB lets the bits of p1 circulate right over 1 or p2 positions.*
Flags affected:	C O (resulting from the last rot step), Z S (according to the final result in p1)
Bytes:	1 (p1 = REG, p2 = none) 2 (p1 = REG, p2 = number)
Cycles:	1 (p1 = REG, p2 = none) 1 + p2 (p1 = REG, p2 = number)
Description:	Without p2 or p2 = 1 : Rotate p1 right by one bit position over carry. This means in detail, shift p1 register to the right, fill MSB with present carry, then move the former LSB to carry. With 2 ≤ p2 ≤ 15 : Rotate p1 right by p2 bit positions over carry. This means in detail, shift p1 register p2 times to the right, in each step fill MSB with the present carry and then move the former LSB to carry. Note: Don't use on register R in combination with bytesel ≠ 0
Category:	Shift and rotate

setC	Set carry flag
Syntax:	setC
Parameters:	-
Calculus:	-
Flags affected:	C O
Bytes:	2
Cycles:	2
Description:	Set carry flag and clear overflow flag
Category:	Flags

shiftL	Shift Left
Syntax:	shiftL p1(, p2)
Parameters:	p1 = REG [x, y, z, r] p2 = no entry or number 2...15
Calculus:	case shiftL p1, without p2: p1 = (p1 << 1); carry = MSB(p1) *"p1 << 1": p1 shifted left by 1 bit, actually means p1 multiplied by 2* *in standard notation: p1 = 2 *p1 as long as MSB remains unchanged* case shiftL p1, p2: p1 = repeat (p2 times) shiftL p1 *in standard notation: p1 = p1*2p2 as long as MSB remains unchanged*
Flags affected:	C O (resulting from the last shift step), Z S (according to the final result in p1)
Bytes:	1 (p1 = REG, p2 = none) 2 (p1 = REG, p2 = number)

shiftL	Shift Left
Cycles:	1 (p1 = REG, p2 = none) 1 + p2 (p1 = REG, p2 = number)
Description:	Without p2 or p2 = 1 : Unsigned Shift p1 left by one bit position, LSB set to zero, MSB shifted out to carry. Note that this can cause fake sign changes. With 2 ≤ p2 ≤ 15 : Unsigned Shift p1 left by p2 bit positions, b2 lower bits set to zero, MSB of last step shifted out to carry. Note that this operation can cause fake sign changes. Check by OVL flag Note: Don’t use on register R in combination with bytesel ≠ 0
Category:	Shift and rotate

shiftR	Shift right
Syntax:	shiftR p1(, p2)
Parameters:	p1 = REG [x, y, z, r] p2 = no entry or number 2...15
Calculus:	case shiftR p1, without p2: p1 = (p1 >> 1); carry = LSB(p1) *“p1 >> 1”: p1 shifted right by 1 bit, actually means p1 divided by 2* *in standard notation:* p1 = p1 /2 with a truncation error if LSB(p1)=1 case shiftR p1, p2: p1 = repeat (p2 times) shiftR p1 *in standard notation:* p1 = p1/2p2 *with some truncation error due to lost lower bits*
Flags affected:	C O (resulting from the last shift step), Z S (according to the final result in p1)
Bytes:	1 (p1 = REG, p2 = none) 2 (p1 = REG, p2 = number)
Cycles:	1 (p1 = REG, p2 = none) 1 + p2 (p1 = REG, p2 = number)
Description:	Without p2 or p2 = 1 : Signed Shift p1 right by one bit position, MSB duplicated to keep sign unchanged, LSB shifted out to carry. The latter can be used to correct a possible truncation error. With 2 ≤ p2 ≤ 15 : Signed Shift p1 right by p2 bit positions, p2 leading bits set to initial MSB to keep sign unchanged. Carry is set to the last LSB shifted out, which can be used to reduce a possible truncation error. Note: Don’t use on register R in combination with bytesel ≠ 0
Category:	Shift and rotate

sign	Sign
Syntax:	sign p1
Parameters:	p1 = REG [x, y, z, r]
Calculus:	p1 = 1 = 0x00000001 if p1 >= 0 p1 = -1 = 0xFFFFFFFF if p1 < 0
Flags affected:	Z S
Bytes:	2
Cycles:	2
Description:	Sign of addressed register in complement of two notations. A positive value returns 1, a negative value returns -1 Zero is assumed to be positive
Category:	Simple arithmetic

skip	Skip
Syntax:	skip p1
Parameters:	p1 = number [1, 2, 3]
Calculus:	PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands
Description:	Skip p1 instructions without conditions. The one, two or three active instructions following the skip command produce no result, except some instructions that may not be skipped (see below). Note that the skipped instructions are processed, but they produce no result or further activity. Use the skip commands (conditional or unconditional) for structured programming or to ignore very short code sequences – for long sequences goto is more effective. Note: The following instructions may not be skipped: bytedir, bytesel, clkmode, clrwdt, equal, equal1, i2crw, mcten
Category:	Skip

skipBitC	Skip on bit clear
Syntax:	skipBitC p1, p2,p3
Parameters:	p1 = REG [x, y, z, r] p2 = number [0...23] p3 = number [1, 2, 3]
Calculus:	if (bit p2 of register p1 == 0) PC = PC + code bytes of next p3 instructions
Flags affected:	-
Bytes:	2
Cycles:	2 + cycles of the skipped commands
Description:	Skip p3 commands if bit p2 of register p1 is clear. See “skip” for more details.
Category:	Skip

skipBitS	Skip on bit set
Syntax:	skipBitS p1, p2,p3
Parameters:	p1 = REG [x, y, z, r] p2 = number[0...23] p3 = number[1, 2, 3]
Calculus:	if (bit p2 of register p1 == 1) PC = PC + code bytes of next p3 instructions
Flags affected:	-
Bytes:	2
Cycles:	2 + cycles of the skipped commands
Description:	Skip p3 commands if bit p2 of register p1 is set. See “skip” for more details.
Category:	Skip

skipCarC	Skip carry clear
Syntax:	skipCarC p1
Parameters:	p1 = number [1, 2, 3]
Calculus:	if (carry == 0) PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands
Description:	Skip p1 commands if carry clear. See “skip” for more details.
Category:	Skip

skipCarS	Skip carry set
Syntax:	skipCarS p1
Parameters:	p1 = number [1, 2, 3]
Calculus:	if (carry == 1) PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands
Description:	Skip p1 commands if carry set. See “skip” for more details.
Category:	Skip

skipEQ	Skip on zero
Syntax:	skipEQ p1
Parameters:	p1 = number [1, 2, 3]
Calculus:	if (Z indicates zero) PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands
Description:	Skip p1 commands if result of previous operation is equal to zero. See “skip” for more details.
Category:	Skip

skipNE	Skip on non-zero
Syntax:	skipNE p1
Parameters:	p1 = number [1, 2, 3]
Calculus:	if (Z indicates not-equal zero) PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands

skipNE	Skip on non-zero
Description:	Skip p1 commands if result of previous operation is not equal to zero. See “skip” for more details.
Category:	Skip

skipNeg	Skip on negative
Syntax:	skipNeg p1
Parameters:	p1 = number [1, 2, 3]
Calculus:	if (S indicates negative) PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands
Description:	Skip p1 commands if result of previous operation was smaller than 0. See “skip” for more details.
Category:	Skip

skipOvrC	Skip on overflow clear
Syntax:	skipOvrC p1
Parameters:	p1 = number [1, 2, 3]
Calculus:	if (O is clear) PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands
Description:	Skip p1 commands if overflow is clear. See “skip” for more details.
Category:	Skip

skipOvrS	Skip on overflow set
Syntax:	skipOvrS p1
Parameters:	p1 = number [1, 2, 3]
Calculus:	if (O is set) PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands
Description:	Skip p1 commands if overflow is set. See “skip” for more details.
Category:	Skip

skipPos	Skip on positive
Syntax:	skipPos p1
Parameters:	p1 = number [1, 2, 3]

skipPos	Skip on positive
Calculus:	if (S indicates positive) PC = PC + code bytes of next p1 instructions
Flags affected:	-
Bytes:	1
Cycles:	1 + cycles of the skipped commands
Description:	Skip p1 commands if result of previous operation was greater or equal to 0. See “skip” for more details.
Category:	Skip

stop	Stop
Syntax:	stop
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	The CPU and the CPU clock are stopped. Usually this instruction is the last command in the assembler listing, it ends any CPU activity. New activity starts by request of the task sequencer or over external communication. Note that the request flag that started the CPU activity must be cleared by the CPU before stop, to indicate that this request was processed.
Category:	Miscellaneous

sub	Subtraction
Syntax:	sub p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r] or 32-Bit number
Calculus:	p1 = p2 – p1
Flags affected:	C O Z S
Bytes:	1 (p1 = REG, p2 = REG) 5 (p1 = REG, p2 = number)
Cycles:	1 (p1 = REG, p2 = REG) 5 (p1 = REG, p2 = number)
Description:	Subtraction of the two parameters
Category:	Simple arithmetic

swap	Swap
Syntax:	swap p1, p2
Parameters:	p1 = REG [x, y, z, r] p2 = REG [x, y, z, r]
Calculus:	p1 = p2 and p2 = p1
Flags affected:	-

swap	Swap
Bytes:	1
Cycles:	3
Description:	Swap of 2 registers. The value of two registers is exchanged between each other.
Category:	Register wise

15.5 ROM Routines in Detail

15.5.1 Data Filtering

ROM routine name	*ROM_INIT_FILTER / ROM_INIT_FILTER1*
Description	Routine to initialize the RAM cells for any block of RAM cells of size N and starting at a given address with a given value (use to initialize rolling average filter). The routine has an alternative start address *ROM_INIT_FILTER1*, where *RAM_R_V32_FILTER* remains unchanged, but Y returns undefined.
Prerequisite	-
Input parameters / register values	X: contains value to be initialized in the RAM cells (any format) Y: Number of RAM cells to be initialized N (integer) Z: Starting RAM Address of the filter
Output/Return value	NVRAM cells starting at the address in Z are initialized with the value in X
Temporary RAM	*RAM_R_V32_FILTER* (remains unchanged when using *ROM_INIT_FILTER1*)
Permanent RAM	-
Routines used	-
Unchanged registers	X, Z; (Y unchanged when using *ROM_INIT_FILTER*)

ROM routine name	*ROM_ROLL_AVG*
Description	This routine averages a list of the last N *FILTER_IN* values using a rolling average filter. With every call it removes the oldest value from the list end and adds the new one at the beginning. Then it determines the new average by calculating the arithmetic mean from the N list values. The final output value must not exceed the maximal representable number/N in the chosen format (else overflow occurs).
Prerequisite	For correct results, all N RAM cells of the filter must contain valid values. Use ROM_INIT_FILTER once before first routine call for proper initialization.
Input parameters / register values	X: new value to be added to the filter list (*FILTER_IN*) (any format) Y: filter length N (integer) Z: Starting address of the rolling average filter RAM cell block of length N
Output/Return value	X: new average (same format as X input)

ROM routine name	*ROM_ROLL_AVG*
Temporary RAM	RAM_R_V32_FILTER
Permanent RAM	-
Routines used	-
Unchanged registers	(all registers X, Y, Z and R are in use)

ROM routine name	*ROM_ROLLAVG_2OUTLIER*
Description	This routine averages a list of the last N FILTER_IN values using a rolling average filter. With every call it removes the oldest value from the list end and adds the new one at the beginning. Then it determines the new average by calculating the arithmetic mean from the N list values – except the one value that deviates the most from the last average (the OUTLIER). This one value is replaced by the previous one (ONLY for the calculation, the original value remains in the list), in order to filter out single error points. The final output value must not exceed the maximal representable number/N in the chosen format (else overflow occurs).
Prerequisite	For correct results, all N RAM cells of the filter must contain valid values. Use *ROM_INIT_FILTER* once before first routine call for proper initialization.
Input parameters / register values	X: new value to be added to the filter list (*FILTER_IN*) (any format) Y: filter length N (integer) Z: Starting address of the rolling average filter RAM cell block of length N *RAM_R_V30_PREV_AVG* with the previous averaged result
Output/Return value	X: new average (same format as X input) Z: last valid value that replaced the OUTLIER (same format as X input) Bit BNR_NEW_VAL_IS_OUTLIER in *RAM_R_FW_STATUS* is set if the current new value is then OUTLIER – to be recognized for later error handling, for example to replace the new value by Z in other calculations.
Temporary RAM	RAM_R_V32_FILTER, RAM_R_V31_FILTER_2, RAM_R_V33_FILTER_SUM
Permanent RAM	RAM_R_FW_STATUS, RAM_R_V30_PREV_AVG
Routines used	-
Unchanged registers	(all registers X, Y, Z and R are in use)

ROM routine name	*ROM_FILTER_FLOW*
Description	Routine to filter the flow value with a rolling average filter of length 16. The routine initializes the filter at its first call with its input value and calculates with each new call a new averaged flow value with the oldest value replaced by the new input from RAM_R_VA2_FLOW_LPH_TO_FLT. The final output value must not exceed the maximal representable number/16 in the chosen format (else overflow occurs).
Prerequisite	All 16 filter cells in RAM, starting at *RAM_R_ROLAVG_1*, must still contain the former values.
Input parameters / register values	*RAM_R_VA2_FLOW_LPH_TO_FLT* (any format)
Output/Return value	X: averaged flow value (same format as input) Values in 16 filter cells in RAM, starting at *RAM_R_ROLAVG_1*, are shifted by one cell, dropping the oldest value at the end and storing the new input value in *RAM_R_ROLAVG_1* First call: all 16 filter cells get initialized to *RAM_R_VA2_FLOW_LPH_TO_FLT*, and Bit BNR_FLOW_FILT_INIT_DONE in *RAM_R_FW_STATUS* is set.

ROM routine name	*ROM_FILTER_FLOW*
Temporary RAM	-
Permanent RAM	*RAM_R_FW_STATUS, RAM_R_VA2_FLOW_LPH_TO_FLT, RAM_R_ROLAVG_1, RAM_ ROLAVG_2 . RAM_ROLAVG_16*
Routines used	*ROM_INIT_FILTER, ROM_ROLL_AVG*
Unchanged registers	(all registers X, Y, Z and R are in use)

15.5.2 Error Detection and Handling

ROM routine name	*ROM_EH*
Description	Error Handling: This routine checks all error flags and suppresses processing of wrong results.
Prerequisite	-
Input parameters / register values	error flags in *SRR_ERR_FLG*
Output/Return value	error counters and flags are updated.
Temporary RAM	-
Permanent RAM	*RAM_R_TM_ERR_CTR, RAM_R_AM_ERR_CTR, RAM_R_USM_ERR_CTR, RAM_R_FW_STATUS, RAM_R_PT_INT_TEMPERATURE, RAM_R_PTC_TEMPERATURE, RAM_R_PTC_TEMPERATURE, RAM_R_V2D_PT_INT_TEMPERATURE_OLDVAL, RAM_R_V2B_PTC_TEMPERATURE_OLDVAL, RAM_R_V2C_PTH_TEMPERATURE_OLDVAL, RAM_R_FHL_ERR_CTR, RAM_R_FLOW_LPH, RAM_R_THETA, RAM_R_V29_FLOW_LPH_OLDVAL, RAM_R_V2A_FLOW_THETA_OLDVAL*
Routines used	*ROM_REPLACE_WITH_OLD_TOFS*
Unchanged registers	Z

ROM routine names	*ROM_PP_AM_MON / ROM_PP1_AM_MON*
Description	AM monitoring: This routine reads the raw amplitude values from the front end data buffer and checks if they are above the user given limit in *FWD_R_AM_MIN*. This is done by direct comparison of *FDB_US_AM_U* and *FDB_US_AM_D* with *RAM_R_AM_MIN_RAW* (provided by *ROM_PP_AM_CALIB* or *ROM_PP1_AM_CALIB*). The routine sets the flag BNR_AMP_VAL_TOO_LOW bit in RAM_R_FW_ERR_FLAGS register, if any of the amplitudes is too low, or clears it in the opposite case (sufficient signal amplitudes). The alternative call *ROM_PP1_AM_MON* does not need the RAM cell *RAM_R_AM_MIN_RAW*, it gets the same value from Z.
Prerequisite	An AM measurement must be done before, with valid results in FDB_US_AM_U and FDB_US_AM_D.
Input parameters / register values	RAM_R_AM_MIN_RAW: Min. amplitude raw value in HSC periods (fd 16) equivalent to the user's minimum amplitude limit FWD_R_AM_MIN in mV. or alternatively, with *ROM_PP1_AM_MON*, use Z instead as input: Z: Minimal. amplitude raw value in HSC periods (as above) (fd 16)
Output/Return value	BNR_AMP_VAL_TOO_LOW in *RAM_R_FW_ERR_FLAGS* register
Temporary RAM	-
Permanent RAM	*RAM_R_FW_ERR_FLAGS* only *ROM_PP_AM_MON*: *RAM_R_AM_MIN_RAW*

ROM routine names	*ROM_PP_AM_MON / ROM_PP1_AM_MON*
Routines used	-
Unchanged registers	X, Y

ROM routine name	*ROM_PP_AM_CALIB / ROM_PP1_AM_CALIB*
Description	These routines are used after an amplitude calibration measurement. Using the new amplitude calibration values (FDB_US_AMC_VH and FDB_US_AMC_VL), they calculate gradient *RAM_R_AMC_GRADIENT* and offset *RAM_R_AMC_OFFSET* that are needed for calculating actual amplitudes (this is not done here, see manual for equations). In addition, they scale the amplitude limit FWD_R_AM_MIN into an equivalent raw value *RAM_R_AM_MIN_RAW*, which can be directly compared to measured time values. This avoids frequent multiplications or divisions. Note that all calculated values change slowly over time, so they need to be updated rarely, but regularly. The routine *ROM_PP_AM_CALIB* uses a hard-coded typical AM amplitude value of 350mV as reference. The alternative call *ROM_PP1_AM_CALIB* has an input cell from firmware data (*FWD_R_VCAL_TYP*) instead, such that the voltage reference can be adapted if necessary. Applied formulae: *RAM_R_AMC_GRADIENT* = *VCAL*/ (FDB_US_AMC_VH - FDB_US_AMC_VL) *RAM_R_AMC_OFFSET* = (2 * FDB_US_AMC_VL - FDB_US_AMC_VH) * *RAM_R_AMC_GRADIENT* (note: for subsequent amplitude calculations, *RAM_R_AMC_OFFSET* must be further corrected outside the routine: offset = *RAM_R_AMC_OFFSET* +(*SHR_ZCD_LVL* - 796)*0.9V/1024) *RAM_R_AM_MIN_RAW* = (FWD_R_AM_MIN + AMC_OFFSET + AM_CORR_FACTOR) / AMC_GRADIENT AM_CORR_FACTOR= (SHR_ZCD_LVL - 796) * 0.9V /1024 SHR_ZCD_LVL is the current zero cross detection level (LSB ≈0.88mV).
Prerequisite	AM calibration measurements must be done before, with valid results in FDB_US_AMC_VH and FDB_US_AMC_VL. The zero cross detection level SHR_ZCD_LVL must be adjusted. All these parameters are assumed to change slowly enough to be considered constant between two AM calibrations.
Input parameters / register values	FWD_R_AM_MIN: User given lower amplitude limit in mV (fd 16) only *ROM_PP1_AM_CALIB*: FWD_R_VCAL_TYP: reference amplitude in mV (integer)
Output/Return value	X = *RAM_R_AM_MIN_RAW* in HSC periods: (fd 16) raw lower amplitude limit for direct measurement comparison Y = *RAM_R_AMC_GRADIENT* in mV/HSC period, (fd 16) *RAM_R_AMC_OFFSET* in mV: (fd 16) parameters for calculation of amplitudes in mV from FDB_US_AM_U and FDB_US_AM_D (use raw values in HSC periods)
Temporary RAM	*RAM_R_VA9_AMC_DIFF*
Permanent RAM	FWD_R_AM_MIN, *RAM_R_AM_MIN_RAW, RAM_R_AMC_GRADIENT, RAM_R_AMC_OFFSET*; only *ROM_PP1_AM_CALIB: FWD_R_VCAL_TYP*
Routines used	*ROM_FORMAT1_64_TO_32BIT*
Unchanged registers	(all registers X, Y, Z and R are in use)

15.5.3 Pulse Interface and Flow Volume

ROM routine name	*ROM_CFG_PULSE_IF*
Description	This routine configures the pulse interface with the parameters calculated from the given configuration. The routine thus prepares the use of *ROM_PI_UPD* for flow output over the pulse interface. The implemented configuration aims at generating not more than one pulse per auto update, to prevent multiple pulses.
Prerequisite	-
Input parameters / register values	X : Number of pulses per liter (*PULSE_PER_LITER*) (integer) Y : Maximum flow in liter per hour (MAX_FLOW) (integer) Z : *MEAS_RATE_INV* = ((TS_CM + 1) * TS_CT * TOF_RATE) / 1024 (fd 16) - TS_CM : Cycle mode (Task sequencer) - TS_CT: Cycle time (Task sequencer) - TOF_RATE: Time of Flight Rate - HS_CLK = 4 MHz
Output/Return value	Pulse interface registers SHR_PI_AU_NMB, SHR_PI_AU_TIME, SHR_PI_TPA and the PI_TPW bits in CR_PI are configured. X: *FLOW_SCALE_FACT* = *PULSE_PER_LITER*MEAS_RATE_INV* (fd 16) The *FLOW_SCALE_FACT* is used to be multiplied with the actual flow (l/h) for updating the pulse interface. It is typically applied by moving it to *RAM_R_FLOW_SCALE_FACT* before calling *ROM_PI_UPD*.
Temporary RAM	*RAM_R_VA4_FLOWVAR_2, RAM_R_VA5_FLOWVAR_1*
Permanent RAM	-
Routines used	*ROM_DIV_BY_SHIFT*
Unchanged registers	(all registers X, Y, Z and R are in use)

ROM routine name	*ROM_PI_UPD*
Description	Pulse Interface Update routine This routine calculates the number of pulses equivalent to a given flow (in l/h), based on the configuration settings, and initializes it in the SHR_PI_NPULSE register.
Prerequisite	*ROM_CFG_PULSE_IF* must be called ONCE before using this routine, and the resulting *FLOW_SCALE_FACTOR* must be moved to *RAM_R_FLOW_SCALE_FACT*.
Input parameters / register values	X : Flow in liter per Hour (fd 16) *RAM_R_FLOW_SCALE_FACT* must contain *FLOW_SCALE_FACT* (fd 16) (using *ROM_CFG_PULSE_IF* routine, see above)
Output/Return value	SHR_PI_NPULSE register is updated with the Number of Pulses equivalent to the current flow in l/h.
Temporary RAM	*RAM_R_VA5_FLOWVAR_1*
Permanent RAM	*RAM_R_FLOW_SCALE_FACT*
Routines used	-
Unchanged registers	(all registers X, Y, Z and R are in use)

ROM routine name	*ROM_PP_PI_UPD*
Description	This routine organizes the update of the pulse interface by calling *ROM_PI_UPD* with *RAM_R_FLOW_LPH* as input argument.
Prerequisite	-
Input parameters / register values	*RAM_R_FLOW_LPH*: current flow in l/h (fd 16)
Output/Return value	SHR_PI_NPULSE register is updated with the Number of Pulses equivalent to the current flow in l/h.
Temporary RAM	-
Permanent RAM	*RAM_R_FLOW_LPH*
Routines used	*ROM_PI_UPD*
Unchanged registers	(all registers X, Y, Z and R are in use)

ROM routine names	***ROM_SAVE_FLOW_VOLUME* / *ROM_SAVE01_FLOW_VOLUME*** ***ROM_SAVE1_FLOW_VOLUME* / *ROM_SAVE11_FLOW_VOLUME*** ***ROM_SAVE2_FLOW_VOLUME* / *ROM_SAVE21_FLOW_VOLUME***			
Description	These routines are used to calculate the flow volume of one measurement cycle from the present flow (usually in l/h) and store it cumulatively to flow volume (usually in cubic meter). Operation: *FLOW_LPH* * *VOLUME_FACTOR* (V.F.) -> flow (e.g. in cubic meters) per cycle -> accumulate (add to/subtract from) the flow volume (integer and fractional part) Calculation steps *for VOLUME_FACTOR* for liter -> l/h: a) *FLOW_LPH*/3600 -> *FLOW_LPS* b) *FLOW_LPS* * *MEAS_RATE_INV* -> *FLOW* in liter per meas. cycle c) Flow in liter per meas. cycle/1000 -> Flow in cubic meter per cycle d) Flow in cubic meter per cycle -> add it to the Flow Volume (integer and fractional part) Actual calculation of the *VOLUME_FACTOR* from configuration data: *VOLUME_FACTOR* = [(TS_CM + 1) * TS_CT * TOF_RATE] / [1024*3600*1000] Here the following configuration parameters are used: - TS_CM : Cycle mode (Task sequencer) - TS_CT: Cycle time (Task sequencer) - TOF_RATE: Time of Flight Rate - HS_CLK =4 MHz The actual decision on units is done by the user through defining the appropriate input scaling (l/h or something else) and VOLUME_FACTOR.			
Difference between the routine calls: The routines operate either on usual RAM or on firmware data (FWD) RAM, which is useful for regular permanent storage. Their input comes from X, Y or RAM, and can have different meaning (see below).	ROM routine name	RAM region:	input from:	parameter meaning:
	[1]*ROM_SAVE_FLOW_VOLUME*	RAM	RAM	Flow & V.F.
	[2]*ROM_SAVE1_FLOW_VOLUME*	RAM	X,Y	Flow & V.F.
	[3]*ROM_SAVE2_FLOW_VOLUME*	FWD	X,Y	Flow & V.F.
	[5]*ROM_SAVE01_FLOW_VOLUME*, [6]*ROM_SAVE11_FLOW_VOLUME*	RAM	X,Y	Volume

ROM routine names	*ROM_SAVE_FLOW_VOLUME / ROM_SAVE01_FLOW_VOLUME* *ROM_SAVE1_FLOW_VOLUME / ROM_SAVE11_FLOW_VOLUME* *ROM_SAVE2_FLOW_VOLUME / ROM_SAVE21_FLOW_VOLUME*			
	[7]*ROM_SAVE21_FLOW_VOLUME,*	FWD	X,Y	Volume
Prerequisite depending on actual call, see above	The RAM cells *RAM_R_FLOW_VOLUME_INT* and *RAM_R_FLOW_VOLUME_FRACTION* must contain the flow volume of previous measurements. [1,4]RAM cells used for input must contain the right parameter (see above)			
Input parameters / register values: The meaning of input depends on the actual call, see above.	[2,3]X or [4]*FWD_R_FLOW_LPH* or [1]*RAM_R_FLOW_LPH*: the present flow value (fd 16, usually in l/h) or [5,6,7,8]X: the additional flow volume (fd 32, usually cubic meters) [2,3]Y or [4]*FWD_R_VOLUME_FACTOR* or [1]*FWD_R_VOLUME_FACTOR*: *VOLUME_FACTOR* (fd 44; note that this is usually a very small number, so the upper 12 fractional digits are zero and "above" the actual data word) or [5,6,7,8]Y = X			
Output/Return value Output may be in usual RAM or FWD, depending on actual call, see above.	64-bit Volume Flow result in RAM Addresses [1,2,5,6]*RAM_R_FLOW_VOLUME_INT* (integer, usually in cubic meters) *RAM_R_FLOW_VOLUME_FRACTION* (fd 32, usually in cubic meters) or [3,4,7,8]*FWD_R_FLOW_VOLUME_INT* (integer, usually in cubic meters) *FWD_R_FLOW_VOLUME_FRACTION* (fd 32, usually in cubic meters)			
Temporary RAM	-			
Permanent RAM depending on actual call, see above	[1,2,5,6]*RAM_R_FLOW_VOLUME_INT*, *RAM_R_FLOW_VOLUME_FRACTION* or [3,4,7,8]*FWD_R_FLOW_VOLUME_INT, FWD_R_FLOW_VOLUME_FRACTION*; [1]*RAM_R_FLOW_LPH, RAM_R_VOLUME_FACTOR* or [4]*RAM_R_FLOW_LPH, RAM_R_VOLUME_FACTOR*			
Routines used	-			
Unchanged registers	(all registers X, Y, Z and R are in use)			

15.5.4 Temperature Measurement

ROM routine name	*ROM_TEMP_POLYNOM*
Description	This routine calculates the temperature of a PT sensor in °C using the polynomial approximation Temperature T = {[(*PT_COEFF2* * *PT_RATIO*) + *PT_COEFF1*] * *PT_RATIO* } + *PT_COEFF0* where *PT_RATIO* = PT_RES / R0 of the PT sensor *PT_COEFF2* = 10.115 (fd 16) *PT_COEFF1* = 235.57 (fd 16) *PT_COEFF0* = -245.683 (fd 16) This polynomial resembles the inverted R(T)-polynomial for PT (according to IEC 60751:2008) within 3mK accuracy between 0°C and 100°C.
Prerequisite	-
Input parameters / register values	X: PT_RATIO (fd 16)

ROM routine name	*ROM_TEMP_POLYNOM*
Output/Return value	X: Temperature in °C (fd 16)
Temporary RAM	-
Permanent RAM	-
Routines used	*ROM_FORMAT1_64_TO_32BIT*
Unchanged registers	(all registers X, Y, Z and R are in use)

ROM routine name	*ROM_TEMP_LINEAR_FN*
Description	This routine is used to calculate the temperature of any sensor as a linear function of sensor resistance using the nominal resistance and sensor slope. Applied formula: Temperature T = (Sensor Resistance at T[°C] – Nominal resistance) / *RAM_R_VAF_REF_RES_VAL* * Sensor slope
Prerequisite	-
Input parameters / register values	X: Nominal resistance (fd 16) Y: Sensor slope (fd 16) Z: Sensor resistance (fd 16) *RAM_R_VAF_REF_RES_VAL*: Reference Resistance (fd 16)
Output/Return value	X: Temperature (fd 16)
Temporary RAM	-
Permanent RAM	*RAM_R_VAF_REF_RES_VAL*
Routines used	*ROM_FORMAT1_64_TO_32BIT*
Unchanged registers	(all registers X, Y, Z and R are in use)

ROM routine name	*ROM_TM_SUM_RESULT*
Description	In sensor temperature measurement, each single time measurement is repeated after some fixed delay time. Averaging these results eliminates a possible 50/60 Hz disturbance. This routine sums up all duplicate measurements (from the frontend data buffer in cells for measurement 1 and 2) and stores it in the frontend data buffer (in the cells of measurement 1). The routine works for all 2-wire or 4-wire temperature measurement results, it reads the configuration from CR_TM.
Prerequisite	The routine should be called directly after a temperature measurement.
Input parameters / register values	All frontend data buffer (FDB) cells (addresses 0x80 – 0x9B)
Output/Return value	The added results overwrite the original measurements in the first measurement FDB cells (addresses 0x80- 0x84 and 0x8A - 0x92)
Temporary RAM	-
Permanent RAM	-
Routines used	-
Unchanged registers	(all registers X, Y, Z and R are in use)

15.5.5 Interface Communication

ROM routine name	*ROM_I2C_ST*
Description	I2C Start Byte Transfer: Initiate an I2C read or write operation, depending on preceding i2crw-command (1=read, 0=write).
Prerequisite	I2C slave device address must be defined in CR_PI_I2C. Read or write direction must be defined by command i2crw (1=read, 0=write).
Input parameters / register values	-
Output/Return value	SRR_MSC_STF contains a flag to indicate I2C acknowledge (Bit I2C_ACK).
Temporary RAM	-
Permanent RAM	-
Routines used	-
Unchanged registers	X, Y, Z, R

ROM routine name	*ROM_I2C_BT*
Description	I2C Byte Transfer: Read or write one byte of data over I2C, depending on preceding i2crw-command (1=read, 0=write).
Prerequisite	I2C slave device address must be defined in CR_PI_I2C. Read or write direction must be defined by command i2crw (1=read, 0=write).In write case, R must point to the desired input RAM cell, and usually the bytedir and bytesel command must be used to select the desired byte part of the 4Byte-word in the RAM cell (use bytedir 0 and bytesel 4, 5, 6 or 7).
Input parameters / register values	R is not changed but used as pointer to the data register by the chip hardware in write case (see “Prerequisite”).
Output/Return value	SRR_MSC_STF contains a flag to indicate I2C acknowledge (Bit I2C_ACK). In read case, SRR_E2P_RD contains the transferred byte. For storing the received byte in a 4Byte RAM cell, use the bytedir and bytesel command to select the desired byte position (use bytedir 1 and bytesel 4, 5, 6 or 7, and the or command to add a new byte to a partly filled 4Byte-word).
Temporary RAM	-
Permanent RAM	-
Routines used	-
Unchanged registers	X, Y, Z, R

ROM routine name	*ROM_I2C_LT*
Description	I2C Byte Transfer: Read or write the last transmitted byte of data over I2C, depending on preceding i2crw-command (1=read, 0=write). The routine sends the stop signal at the end of transmission.
Prerequisite	I2C slave device address must be defined in CR_PI_I2C. Read or write direction must be defined by command i2crw (1=read, 0=write).In write case, R must point to the desired input RAM cell, and usually the bytedir and bytesel command must be used to select the desired byte part of the 4Byte-word in the RAM cell (use bytedir 0 and bytesel 4, 5, 6 or 7).
Input parameters / register values	R is not changed but used as pointer to the data register by the chip hardware in write case (see “Prerequisite” below).

ROM routine name	*ROM_I2C_LT*
Output/Return value	SRR_MSC_STF contains a flag to indicate I2C acknowledge (Bit I2C_ACK). In read case, SRR_E2P_RD contains the transferred byte. For storing the received byte in a 4Byte RAM cell, use the bytedir and bytesel command to select the desired byte position (use bytedir 1 and bytesel 4, 5, 6 or 7, and the or command to add a new byte to a partly filled 4Byte-word).
Temporary RAM	-
Permanent RAM	-
Routines used	-
Unchanged registers	X, Y, Z, R

ROM routine name	*ROM_I2C_DWORD_WR*
Description	This routine is used to write a 4 byte-word of data to the I2C slave device, starting at a given memory address. The device address for the I2C device is taken automatically from I2C slave address of CR_PI_I2C. The routine starts with switching on the HSC and switches it off after transmission. It thus causes a high current consumption and has a long runtime. In power critical applications, its use should thus be restricted. Timing properties of the I2C slave should also be considered (e.g. long storing times after some data transmission).
Prerequisite	-
Input parameters / register values	X: 16-bit memory address (start) where the data has to be written Y: 4 bytes of data
Output/Return value	The four bytes from Y are written to the I2C slave, starting at the address given in X. In case of an error, the transmission was not acknowledged by the I2C slave device and bit BNR_I2C_ABORT of *RAM_R_FW_STATUS* is set.
Temporary RAM	-
Permanent RAM	*RAM_R_VA1_I2CADDR, RAM_R_VA2_I2CDATA, RAM_R_FW_STATUS*
Routines used	*ROM_I2C_ST, ROM_I2C_BT, ROM_I2C_LT*
Unchanged registers	Z

ROM routine name	*ROM_I2C_BYTE_WR*
Description	This routine is used to write a single of data to the I2C slave device to given memory address. The device address for the I2C device is taken automatically from I2C slave address of CR_PI_I2C. The routine starts with switching on the HSC and switches it off after transmission. It thus causes a high current consumption and has a long runtime. In power critical applications, its use should thus be restricted. Timing properties of the I2C slave should also be considered (e.g. long storing times after some data transmission).
Prerequisite	-
Input parameters / register values	X: 16-bit address where the data has to be written Y: 1 byte of data (B0 of the 32 bit-word in Y is transferred)
Output/Return value	Byte B0 from Y is written to the I2C slave to the address given in X. In case of an error, the transmission was not acknowledged by the I2C slave device and bit BNR_I2C_ABORT of *RAM_R_FW_STATUS* is set.
Temporary RAM	-
Permanent RAM	*RAM_R_VA1_I2CADDR, RAM_R_VA2_I2CDATA, RAM_R_FW_STATUS*
Routines used	*ROM_I2C_ST, ROM_I2C_BT, ROM_I2C_LT*
Unchanged registers	Z

ROM routine name	*ROM_I2C_DWORD_RD*
Description	This routine is used to sequentially read 4 data bytes from the I2C slave device, starting at a given memory address. The device address for the I2C device is taken automatically from I2C slave address of CR_PI_I2C. The routine starts with switching on the HSC and switches it off after transmission. It thus causes a high current consumption and has a long runtime. In power critical applications, its use should thus be restricted. Timing properties of the I2C slave should also be considered.
Prerequisite	-
Input parameters / register values	X: 16-bit memory address (start) where the data is read.
Output/Return value	X: 4 bytes of read data In case of an error, the transmission was not acknowledged by the I2C slave device and bit BNR_I2C_ABORT of *RAM_R_FW_STATUS* is set.
Temporary RAM	-
Permanent RAM	*RAM_R_VA1_I2CADDR, RAM_R_FW_STATUS*
Routines used	*ROM_I2C_ST, ROM_I2C_BT, ROM_I2C_LT*
Unchanged registers	Z

ROM routine name	*ROM_I2C_BYTE_RD*
Description	This routine is used to sequentially read one single byte from the I2C slave device from a given memory address. The device address for the I2C device is taken automatically from I2C slave address of CR_PI_I2C. The routine starts with switching on the HSC and switches it off after transmission. It thus causes a high current consumption and has a long runtime. In power critical applications, its use should thus be restricted. Timing properties of the I2C slave should also be considered.
Prerequisite	-
Input parameters / register values	X: 16-bit memory address where the data is read.
Output/Return value	X: Single data byte, stored in byte B0 of X In case of an error, the transmission was not acknowledged by the I2C slave device and bit BNR_I2C_ABORT of *RAM_R_FW_STATUS* is set.
Temporary RAM	-
Permanent RAM	*RAM_R_VA1_I2CADDR, RAM_R_FW_STATUS*
Routines used	*ROM_I2C_ST, ROM_I2C_BT, ROM_I2C_LT*
Unchanged registers	Z

15.5.6 Housekeeping

ROM routine name	*ROM_CPU_CHK*
Description	Check kind of CPU request: This routine is called by hardware design after any Post Processing (PP) request. It checks the system handling register SHR_CPU_REQ and calls the requested routines.
Prerequisite	-

ROM routine name	*ROM_CPU_CHK*
Input parameters / register values	Flag settings in SHR_CPU_REQ
Output/Return value	The routine directly calls firmware (MK_CPU_REQ), boot loader (*ROM_BLD*) or checksum generation (*ROM_CSM*) using goto. These routines generally return, after execution, to the start of *ROM_CPU_CHK* to see if SHR_CPU_REQ has changed in the meantime.
Temporary RAM	(depending on called routines)
Permanent RAM	(depending on called routines)
Routines used	MK_CPU_REQ, *ROM_BLD, ROM_CSM*
Unchanged registers	(all registers X, Y, Z and R are in use)
Call Address	61440 / 0xF000

ROM routine name	*ROM_USER_RAM_INIT*
Description	This ROM routine is used to initialize the entire user RAM with 0 as default value.
Prerequisite	-
Input parameters / register values	-
Output/Return value	All 176 user RAM cells (addresses 0x00 - 0xAF) are initialized to 0
Temporary RAM	All 176 user RAM cells
Permanent RAM	-
Routines used	-
Unchanged registers	Y, Z

15.5.7 High-speed Oscillator

ROM routine name	*ROM_SCALE_WITH_HSC*
Description	Routine to scale the input parameter with the HS Clock Calibration factor (*RAM_R_HSC_SCALE_FACT*) Scaled output = Input / *RAM_R_HSC_SCALE_FACT*
Prerequisite	*RAM_R_HSC_SCALE_FACT* must have the valid HS Clock Calibration factor (Use *ROM_HSC_CALIB* routine)
Inputs	X - Parameter to be scaled (any format, integer value < 2^{30})
Output	X - Scaled parameter (same format as input X)
Temporary RAM	-
Permanent RAM	*RAM_R_HSC_SCALE_FACT*
Routines used	-
Unchanged registers	Z

ROM routine name	*ROM_PP_HSC_CALIB*
Description	
Prerequisite	

ROM routine name	*ROM_PP_HSC_CALIB*
Input parameters / register values	
Output/Return value	
Temporary RAM	
Permanent RAM	
Routines used	
Unchanged registers	

15.5.8 Configuration

ROM routine name	*ROM_RECFG_TOF_RATE*
Description	Routine to reconfigure TOF_RATE generator for less measurements, depending on the parameter N: New TOF_RATE = Original TOF_RATE * N The actual measurement is done only every Nth time. The routine manipulates SHR_TOF_RATE for this adjustment. It is only usable for TOF_RATES up to 31.
Prerequisite	-
Input parameters / register values	X - Factor N for lowering the TOF_RATE
Output/Return value	Z - Original TOF_RATE value from SHR_TOF_RATE Register TOF_RATE bits in SHR_TOF_RATE Register are changed (see description). Bit BNR_TOF_RATE_REDUCED is set in *RAM_R_FW_STATUS* register to indicate that the TOF_RATE was reconfigured.
Temporary RAM	-
Permanent RAM	RAM_R_FW_STATUS
Routines used	-
Unchanged registers	(all registers X, Y ,Z and R are in use)

15.5.9 Mathematics

ROM routine name	*ROM_FORMAT1_64_TO_32BIT*
description	Routine to format a 64-bit value (in Y and X) into a 32 bit result with 16 integer + 16 fractional bits This can be used to format 64 bit multiplication results with 32 integer + 32 fractional bits into a usual fd 16 word. The MSB of the integer part in Y defines the sign, as if Y and X would be a single 64 bit word. This routine has the same function as *ROM_FORMAT_64_TO_32BIT*, but is essentially faster at the cost of one temporary RAM cell.
Prerequisite	-
Input parameters / register values	Y: Higher 32 bits of the value (integer part with maximum 16 significant bits, signed !) X: Lower 32 bits of value (fractional part, unsigned !)

ROM routine name	*ROM_FORMAT1_64_TO_32BIT*
Output/Return value	X: 32-bit result with 16 Integer + 16 fractional bits (fd 16)
Temporary RAM	*RAM_R_V1F_SHIFT*
Permanent RAM	-
Routines used	-
Unchanged registers	Z

ROM routine name	*ROM_DIV_BY_SHIFT*
Description	Routine to perform the division of a value Y by X, where X=2^N is an integer power of two. Result = Y/X = Y/2^N
Prerequisite	-
Input parameters / register values	X - Divisor (denominator) = 2^N value (integer) Y - Dividend (numerator) (any format)
Output/Return value	Y - Result of division (same format as input Y)
Temporary RAM	-
Permanent RAM	-
Routines used	-
Unchanged registers	Z, R

ROM routine name	*ROM_SQRT*
Description	This routine is used to evaluate the square root using the Newton method accurately for values in the range (196 <= X <= 5476). sqrt(x) is calculated by iterating the following steps 1. Choose GUESS = 32; Iteration counter = 3 2. Find x/GUESS (1st division by shift and normal division for next 2 iterations) 3. Average of GUESS and x/GUESS 4. GUESS <--- Average ; Decrement iteration counter 5. Repeat steps 2-4 till counter = 0 6. Square Root = Last GUESS value When used in flow temperature calculation, values of X between 196 = (14^2) and 5476 = (74^2) result in temperature errors < 1°C. This X range is equivalent to a temperature range of 60°C to 0°C.
Prerequisite	-
Input parameters / register values	X : Radicand (fd 16, 196 <= X <= 5476)
Output/Return value	: Square root of input X (fd 16)
Temporary RAM	*RAM_R_V2F_SQRT_X, RAM_R_V2E_SQRT_Y*
Permanent RAM	-
Routines used	-
Unchanged registers	(all registers X, Y, Z and R are in use)

ROM routine name	*ROM_LINEAR_CORRECTION / ROM_LINEAR1_CORRECTION*
Description	Linear interpolation of a coefficient over any parameter (here: temperature), knowing the coefficient value at two points and given the current value of the parameter (stored in *RAM_R_VA3_CURRENT_THETA*) Applied formula: Result = slope * (*RAM_R_VA3_CURRENT_THETA* - Parameter@Point1) + offset = X * (RAM_R_VA3_CURRENT_THETA - Y) + Z When the coefficient value is known at two parameter points, slope and offset can be calculated as slope = (Coefficent@Point2 - Coefficent@Point1) / (Parameter@Point2 - Parameter@Point1) offset = Coefficent@Point1 The routine has an alternative call address *ROM_LINEAR1_CORRECTION*, where the RAM cell of the current parameter value can be freely chosen.
Prerequisite	-
Input parameters / register values	X : Slope between the two points (fd 16) Y: Parameter@Point1 (fd 16) Z: Offset (fd 16) *RAM_R_VA3_CURRENT_THETA* current parameter (temperature) (fd 16) With alternative call *ROM_LINEAR1_CORRECTION*: R: Pointer to RAM cell with current parameter value
Output/Return value	X: Coefficient corrected linearly over temperature
Temporary RAM	-
Permanent RAM	*RAM_R_VA3_CURRENT_THETA* (none when *ROM_LINEAR1_CORRECTION* is used)
Routines used	*ROM_FORMAT1_64_TO_32BIT*
Unchanged registers	(all registers X,Y,Z and R are in use)

ROM routine name	*ROM_FIND_SLOPE*
Description	This routine is used to find the slope between two points, given the coefficient and corresponding parameter values at the two points (for example two correction factors over two temperatures). The routine is used as preparation for any linear interpolation. Basically, all input- and output-values have the same format. Due to internal calculations, the format must be chosen such that the four leading bits of the parameters are zero, and at least 12 leading bits of the resulting slope are zero, too. Otherwise the result will be wrong.
Prerequisite	Parameter interval (*RAM_R_VA5_FLOWVAR*_1 – Z) must be numerically larger than (Y - X)/23, to avoid overflow in an internal division.
Input parameters / register values	X : Coefficient at point 1 (any format, typically 16 fd) Y: Coefficient at point 2 (same format as X) Z: Parameter at point 1 (same format as X, 4 leading bits must be 0) *RAM_R_VA5_FLOWVAR_1*: Parameter at point 2 (same format as Z)
Output/Return value	X : Slope (same format as input X; 12 leading bits are always 0)
Temporary RAM	-
Permanent RAM	RAM_R_VA5_FLOWVAR_1
Routines used	-
Unchanged registers	(all registers X, Y, Z and R are in use)

15.6 Amplitude Calculation

The raw data of the amplitude measurement and amplitude calibration measurement are stored in the frontend data buffer:

FDB_US_AM_U ≡ $AM_{Up}[ns]$

FDB_US_AM_D ≡ $AM_{Down}[ns]$

FDB_US_AMC_VH ≡ $AMC_{high}[ns]$

FDB_US_AMC_VL ≡ $AMC_{\text{low}}\ [ns]$

The calibrated amplitudes in Volt are calculated according following formulas:

Equation 1:

$$V_{Up}[mV] \ = AMC_{Gradient}\left[\frac{mV}{ns}\right] \times AM_{Up}[ns] \ - \ AMC_{Offset}\left[ns\,\frac{mV}{ns}\right]$$

$$V_{Down}[mV] \ = AMC_{Gradient}\left[\frac{mV}{ns}\right] \times AM_{Down}[ns] \ - \ AMC_{Offset}\left[ns\,\frac{mV}{ns}\right]$$

With

$$AMC_{Gradient}\left[\frac{mV}{ns}\right] \ = \frac{V_{Cal}[mV]}{AMC_H[ns] - AMC_L[ns]};\ V_{Cal} \ = typ.{}^{V_{ref}}\!/_{2} = 350mV$$

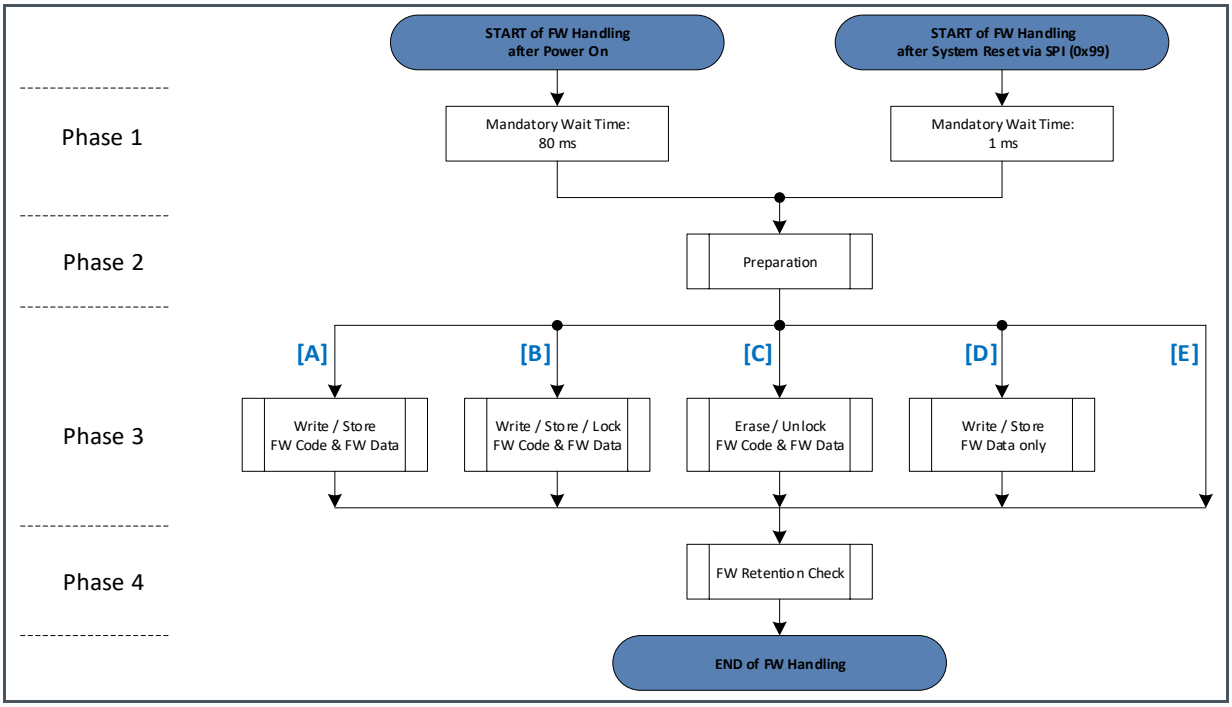
$$AMC_{Offset}[mV] \ = \ (2 \times AMC_L[ns] - AMC_H[ns]) \times AMC_{Gradient}\left[\frac{mV}{ns}\right]$$

15.7 FW Handling Procedures

For FW handling, following procedures can be distinguished:

[A]	Write / Store	FW Code	&	FW Data
[B]	Write / Store / Lock	FW Code	&	FW Data
[C]	Erase / Unlock	FW Code	&	FW Data
[D]	Write / Store			FW Data only
[E]	Verify	FW Code	&	FW Data

Figure 171:
Firmware handling



FW handling procedures needs 3 or 4 phases to be properly performed:

Phase 1:	Wait time	(dependent on start option)
Phase 2:	Preparation	(common for all procedures)
Phase 3:	FW Update	(different for procedures [A], [B], [C], [D])
Phase 4:	FW Retention Check	(common for all procedures)

Phase 3 is not needed for procedure [E].

START of any FW handling procedure is independent from current status of **AS6031**: Measure cycle mode can be either in disabled or in enabled state.

After END of any FW handling procedure the **AS6031** is in IDLE state.
Then flow meter mode can be triggered by executing a system reset (**RC_SYS_RST**) or a system init (**RC_SYS_INIT**).
The following recommended FW handling procedures are timer based as well based on interrupt handling.

15.7.1 Phase1: Initial Wait Time

Initial wait time depends on start option:

- Power On
- System Reset via SPI

Figure 172:
Start with Power On

Step	Description	SPI Opcodes & Data
1	Power On of AS6031	
2	Mandatory wait time: At least 80 ms	

The wait time after Power On is derived by release of internal power on resets.
After this wait time SPI communication is possible with AS6031.

Figure 173:
Start with System Reset

Step	Description	SPI Opcodes & Data
1	Execute System Reset by sending RC_SYS_RST	0x99
2	Mandatory wait time: At least 1 ms	

15.7.2 Phase 2: Preparation

The preparation phase is common for all procedures and sets device in a stable idle state before starting a FW update.

Figure 174:
Preparation

Step	Description	SPI Opcodes & Data
1	Request Bus Master	0x88

Step	Description	SPI Opcodes & Data
2	Disable Watchdog by writing code to **CR_WD_DIS**	0x5A 0xC0 0x48DBA399
3	Set RESTART_EN by writing code to **CR_TRIM2** *Sets also recommended CPU_SPEED*	0x5A 0xCD 0x40100000
4	Disable BG, & Post Processing settings and set Measure Cycle Time to max. value by writing code to **CR_MR_TS**	0x5A 0xC6 0x00001000
5	Execute Supervisor Init by sending **RC_SV_INIT** *Clears task sequencer and sets MCT = OFF*	0x9C
6	**Mandatory** wait time: At least **1** ms	
7	Request Dummy Measurement Task *To clear pending HCC/ZCC calibration due to SV_INIT*	0xDA 0x00
8	**Mandatory** wait time: At least **1** ms	
9	Clear Interrupt, Error & FEP Status Flag by writing to **SHR_EXC**	0x5A 0xDD 0x00000007
10	Execute Reset Flag Clear in SYS_STATUS by sending **RC_RF_CLR**	0x89
11	Enable important interrupt & error flags in **CR_IEH**[19,17,15,14,13,12] for following FW transactions	0x5A 0xC4 0x000AF000
12	Release Bus Master	0x87
13	Perform Recall of Firmware Data via **FWD_RECALL** (bit 20) in **SHR_RC**	*See subroutine "Perform FW Transaction"*
14	Perform Recall of Firmware Code via **FWC_RECALL** (bit 19) in **SHR_RC**	*See subroutine "Perform FW Transaction"*
IDLE state reached & Ready for FW Update		

Remarks:

- Step 14
  Can be skipped if FW update **[D]** (Write / Store of FW Data only) is performed subsequently

15.7.3 Phase 3: FW Update

To fulfill specified NVRAM parameters "Data Retention" and "Endurance", Vcc supply has to be in the range of 3.0 – 3.6 V when FW is updated.

[A]: Write / Store FW Code & FW Data
[B]: Write / Store / Lock FW Code & FW Data

Figure 175:
[A] or [B]: Write / Store / (Lock) FW Code & FW Data only

Step	Description	SPI Opcodes & Data
1	Get range of FW user code by reading **SRR_FWU_RNG**	0x7A 0xEC *read data (Bit [31:0])*
2	Write Firmware Code User **FWCU** to FWC addresses 32….[FWU_RNG-1]	0x5C 0x00 0x20 0xXX 0xXX …..
3	Write Firmware Data User **FWDU** to FWD addresses 2….119	0x5B 0x02 0xXXXXXXXX 0xXXXXXXXX …….
4	After calculation: Write expected checksums for FWCU & FWDU to FWD addresses 0….1: **FWCU_CS_EXP**, **FWDU_CS_EXP**	0x5B 0x00 0xXXXXXXXX 0xXXXXXXXX
5	**[A]:** Write / Store FW Code & FW Data Perform Store of Firmware Code & Data via **FW_STORE_ALL** (bit 16) in **SHR_RC**	*See subroutine "Perform FW Transaction"*
	[B]: Write / Store / Lock FW Code & FW Data Perform Store of Firmware Code & Data with Lock via **FW_STORE_LOCK** (bit 17) in **SHR_RC**	*See subroutine "Perform FW Transaction"*

Remarks:

- Step 1 & 2
  Writing FW Code up to [FWU_RNG-1] is not required in any case. For a one-time customer update (e.g. in production flow), the Firmware Code can be written up to last address of customer code only, as user area is filled with 0x00 up to [FWU_RNG-1] when delivered to customer.
- Step 4
  Writing checksums separately facilitates the generation of checksums before in steps 2 & 3 while code and data are transferred. If checksums are generated before transferring of code and data, steps 3 & 4 can be combined to 1 block transfer: 0x5B 0x00 0xXXXXXXXX 0xXXXXXXXX ……..
- Step 5
  This step is the only difference between FW update **[A]** and **[B]**. All steps before are common.

Figure 176:
[C]: Erase / Unlock FW Code & FW Data

Step	Description	SPI Opcodes & Data
1	Perform Erase of Firmware Code & Data via **FW_ERASE** (bit 18) in **SHR_RC**	*See subroutine "Perform FW Transaction"*

Figure 177:
[D]: Write / Store FW Data only

Step	Description	SPI Opcodes & Data
1	Write Firmware Data User **FWDU** to FWD addresses 2….119	0x5B 0x02 0xXXXXXXXX 0xXXXXXXXX …….
2	After calculation: Write expected checksum for FWDU to FWD address 1: **FWDU_CS_EXP**	0x5B 0x01 0xXXXXXXXX
3	Perform Store of Firmware Data via **FWD_STORE** (bit 22) in **SHR_RC**	*See subroutine "Perform FW Transaction"*

Remarks:

The previous recommended sequences for **[A]** – **[D]** allow minimum write access times by addressing user parts only (FWCU & FWDU). Alternatively, whole memory space can be addressed without any effect on applied FW sections (FWCA & FWDA)

15.7.4 Phase 4: FW Retention Check

Figure 178:
FW retention check

Step	Description	SPI Opcodes & Data
1	Perform Recall of Firmware Code via **FWC_RECALL** (bit 19) in **SHR_RC**	*See subroutine "Perform FW Transaction"*
2	Perform Recall of Firmware Data via **FWD_RECALL** (bit 20) in **SHR_RC**	*See subroutine "Perform FW Transaction"*
3	Initialize checksum error flags in **SHR_GPO** SHR_GPO[18:12] == b1111111	0x5A 0xD3 0x0007F000

Step	Description	SPI Opcodes & Data
4	Execute Checksum Generation by sending **RC_FW_CHKSUM**	0xB8
5	Check that FW checksum has been finished by reading **CHKSUM_FNS** (bit 3) in **SRR_IRQ_FLAG**	*See subroutine "Interrupt Handling"*
6	Retention check by reading **SHR_GPO** PASS: SHR_GPO[18:12] == b0000000 FAIL: SHR_GPO[18:12] != b0000000	0x7A 0xD3 *read data (Bit [31:0])*

Remarks:

- Steps 1 & 2
  Can be skipped if FW procedure **[E]** (Verify of FW Code & Data) is performed (no FW update before)

15.7.5 Common Subroutines

Figure 179:
Interrupt handling

Step	Description	SPI Opcodes & Data
a	Wait on interrupt INTN	
b	Check interrupt flag on reading **SRR_IRQ_FLAG**(bit x) *0: not needed* 1: FW_TRANS_FNS *2: not needed* 3: CHKSUM_FNS 4-7: *not needed*	0x7A 0xE0 *read data (Bit x of 32)*
c	Clear interrupt flag register by sending **RC_IF_CLR**	0x8D

Interrupt Handling requires that appropriate interrupt flag in **CR_IEH** is enabled in advance.

Figure 180:
Preform firmware transaction

Step	Description	SPI Opcodes & Data
a	Enable FW Transaction by writing release code to **SHR_RC_RLS**	0x5A 0xDF 0x50F5B8CA

Step	Description	SPI Opcodes & Data
b	Execute FW Transaction by writing code to **SHR_RC**(bit x) 16: FW Store All 17: FW Store & Lock 18: FW Erase & Unlock 19: FW Code Recall 20: FW Data Recall 21: *not needed* 22: FW Data Store	0x5A 0xDE *write data (Bit x of 32)*
c	Check that FW transaction has been finished by reading **FW_TRANS_FNS** (bit 1) in **SRR_IRQ_FLAG**	*See subroutine "Interrupt Handling"*

15.7.6 Optional Status Check

Figure 181:
Preform optional status check

Step	Description	SPI Opcodes & Data
1	Prepare & perform short bootload sequence for updating FW revisions by writing code to **CR_TRIM3, FWD_ACR, SHR_CPU_REQ**	0x5A 0xCE 0x00000000 0x5B 0x6B 0x00000000 0x5A 0xDC 0x00000001
2	**Mandatory** wait time: At least **1** ms	
3	Get FW Range, FWU Revision, FWA Revision by reading **SRR_FWU_RNG, SRR_FWU_REV, SRR_FWA_REV**	0x7A 0xEC *read data (Bit [31:0])* *read data (Bit [31:0])* *read data (Bit [31:0])*
4	Check status of Watchdog and Lock State by reading **SRR_MSC_STF**: Bit 15: Watchdog State Bit 2: Lock State	0x7A 0xEA *read data (Bit [31:0])*
5	Check different status bits by reading System Status Bit 7: Error Flag Bit 5: Communication Fail Bit 4: Measure Cycle Timer State	0x8F *read data (Bit [7:0])*

Remarks:

- Steps 1 & 2 are mandatory for step 3
- Steps 4 & 5 can be performed separately

15.8 Measurement Start in Time Conversion Mode

A measurement start in time conversion mode typically requires that “Autoconfig Release Code” is disabled. Further interactions via remote interface have to be performed as follows (greyed actions as described above in flow diagram):

Figure 182:
Measurement Start

Step	Description	Opcodes & Data
1	Wait on interrupt INTN	
2	Check interrupt flag on reading **SRR_IRQ_FLAG**(bit 2) 2: BLD_FNS	0x7A 0xE0 *read data (Bit 2 of 32)*
3	Clear interrupt flag register by sending **RC_IF_CLR**	0x8D
4	Write Configuration Data to CR addresses: 0x0C0….0x0CB SHR addresses: 0x0D0…0x0D2 / 0x0DA…0x0DB	0x5A 0xC0 0xXXXXXXXX ……. 0x5A 0xD0 0xXXXXXXXX …….
5	Set Measure Cycle Timer On	0x8B
6	Check if Cycle Timer is on with RC_RD_STATUS, bit 4 MCT_STATE	0x8F *read byte*

16 Known Errors

16.1 Amplitude Measurement

- Description:

AM measurement gets corrupted or fails completely at extreme operating parameters.

- Workaround:
  4 actions needed:

Figure 183:
Workaround

Actions		
HW implementation on PCB: GIPO4 connected to GPIO1		
GPIO4	CR_GP_CTRL[19:16]	b0100
GPIO1	CR_GP_CTRL[7:4]	b1111
TI_EN_MODE	CR_TRIM3[19]	b1

17 Revision Information

Document Status	Product Status	Definition
Product Preview	Pre-Development	Information in this datasheet is based on product ideas in the planning phase of development. All specifications are design goals without any warranty and are subject to change without notice
Preliminary Datasheet	Pre-Production	Information in this datasheet is based on products in the design, validation or qualification phase of development. The performance and parameters shown in this document are preliminary without any warranty and are subject to change without notice
Datasheet	Production	Information in this datasheet is based on products in ramp-up to full production or full production which conform to specifications in accordance with the terms of ScioSense B.V. standard warranty as given in the General Terms of Trade
Datasheet (discontinued)	Discontinued	Information in this datasheet is based on products which conform to specifications in accordance with the terms of ScioSense B.V. standard warranty as given in the General Terms of Trade, but these products have been superseded and should not be used for new designs

Changes from previous version to current revision v1-00	Page
First officially released datasheet	all

- Page and figure numbers for the previous version may differ from page and figure numbers in the current revision.
- Correction of typographical errors is not explicitly mentioned.

18 Legal Information

Copyrights & Disclaimer

Devices sold by ScioSense B.V. are covered by the warranty and patent indemnification provisions appearing in its General Terms of Trade. ScioSense B.V. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein. ScioSense B.V. reserves the right to change specifications and prices at any time and without notice. Therefore, prior to designing this product into a system, it is necessary to check with ScioSense B.V. for current information. This product is intended for use in commercial applications. Applications requiring extended temperature range, unusual environmental requirements, or high reliability applications, such as military, medical life-support or life-sustaining equipment are specifically not recommended without additional processing by ScioSense B.V. for each application. This product is provided by ScioSense B.V. “AS IS” and any express or implied warranties, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose are disclaimed.

ScioSense B.V. shall not be liable to recipient or any third party for any damages, including but not limited to personal injury, property damage, loss of profits, loss of use, interruption of business or indirect, special, incidental or consequential damages, of any kind, in connection with or arising out of the furnishing, performance or use of the technical data herein. No obligation or liability to recipient or any third party shall arise or flow out of ScioSense B.V. rendering of technical or other services.

RoHS Compliant & ScioSense Green Statement

RoHS Compliant: The term RoHS compliant means that ScioSense B.V. products fully comply with current RoHS directives. Our semiconductor products do not contain any chemicals for all 6 substance categories, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, RoHS compliant products are suitable for use in specified lead-free processes.

ScioSense Green (RoHS compliant and no Sb/Br): ScioSense Green defines that in addition to RoHS compliance, our products are free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material).

Important Information: The information provided in this statement represents ScioSense B.V. knowledge and belief as of the date that it is provided. ScioSense B.V. bases its knowledge and belief on information provided by third parties and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. ScioSense B.V. has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. ScioSense B.V. and ScioSense B.V. suppliers consider certain information

to be proprietary, and thus CAS numbers and other limited information may not be available for release.