AS6031F1

Flow Firmware – Description and Application Guide

AS6031F1 Application Note

Revision: 2

Release Date: 2022-12-0202

Document Status: Production

Content Guide

AS6031F1 is a variant of the AS6031 ultrasonic flow converter that comes with a protected flow firmware by ScioSense already programmed into the NVRAM. Based on these algorithms and together with the appropriate calibration and operation parameters the chip is ready to do the complete flow and volume calculation as well as error handling on chip. This brings great advantage with respect to power consumption of the whole system.

In this application note we describe the operation of the firmware in principle, the base parameters as stored in the firmware data and the output data provided in the RAM.

1 Introduction

This application note describes the ScioSense firmware version as implemented in the ultrasonic flow converter AS6031F1. It describes the functionality of the firmware, the organization and format of the input data as well as of the output data.

Why should we have the flow calculation in the AS6031? Flow calculation in ultrasonic flow meters is a complex task, using many inputs from the analog frontend like time-of-flight, pulse width and amplitude. The conversion from time-of-flight (ToF) to flow demands linearization and calibration. For short- and long-term operation stability, a complex error handling is needed, which is based on hardware error flags but also data analysis. And all this should be achieved with lowest power consumption.

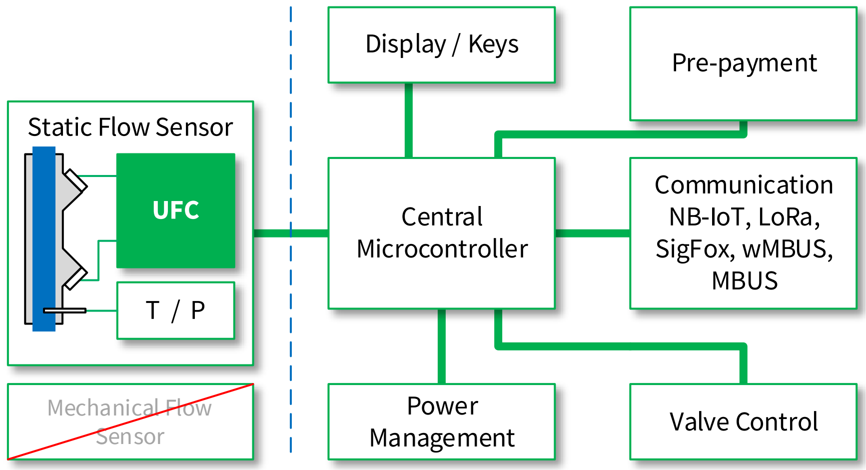


Figure 1: Ultrasonic Flow Meter Block diagram

ScioSense designed a dedicated firmware that covers all this and provides several advantages to the user:

- Fast time to market. The user can focus on other tasks like spool piece design, characterization and calibration. No need to program the AS6031.
- Lowest power due to optimized CPU and reduced communication with the external µP. AS6031F1 runs autonomously, so the external µP can sleep most of the time.
- High flexibility in the choice of the external µP. Ideally. Users can continue with their existing platforms, e.g., those being used for mechanical meters, or they select the most appropriate controller available from the shelf.

- The specification for the central µP with respect to connectivity, display and data management will be very volatile, while the flow calculation will be a constant. A split is therefore recommended.
- Flow and volume are available as digital data via SPI. Alternatively, the flow can be output via pulse interface, which can be compatible to mechanical meters.

The ScioSense firmware always aims at operation in flow meter mode, where the chip measures autonomously and indicates the availability of results by an interrupt to the external microcontroller. The configuration is stored as part of the firmware data and copied into the configuration registers after power-on reset. When it is disturbed or interrupted, it will resume nominal operation by a watchdog reset, and when its configuration is modified, it will restore it latest after one hour. Some configurations can't be changed at all but will be controlled by the firmware, see section 5.13 for details.

The right way to stop the firmware and change configurations, for example for testing, is to switch off post processing and disable the watchdog (else a reset happens within typically 13 s). Then any configuration can be tested without firmware operation. For tests with active firmware the desired configuration should be stored in firmware data, see section 9.3.1. With a few exceptions, a modified configuration can be written to the chip directly while the firmware operates. But note that the firmware will restore its stored configuration at any full hour of the built-in real time clock **SRR_TS_HOUR** and **SRR_TS_MIN_SEC** (0x0E6 and 0x0E7).

2 Firmware Features

AS6031F1 comes with ScioSense firmware that does the full flow and temperature calculation. There is no need for the user to write code by himself. Only firmware data need to be adopted with respect to operation limits and linearization coefficients according to the specific sensor. An external controller can read the measurement results like flow and temperature from specified memory cells over SPI on demand. The measurement is running permanently without any need of external control. As an option, users of course can add their own code like additional filtering, averaging or adoption to production process.

Typical properties of the firmware are:

- Flow measurement
- Full linearization and calibration, either using a standard piece-wise linear correction or ScioSense proprietary method.
- Temperature measurement for cold water meters derived from speed of sound (up to 60 °C).
- Temperature measurement using resistive sensors 0.5 to 5 kΩ with built-in PT1000 characteristic, procedures for 2- and 4-wire measurement and a calculation accuracy in 4-wire mode of 10 mK.
- Optional internal chip temperature measurement.
- Zero flow detection e.g., down to 0.5 l/h for Q3 = 4000 l/h.
- Full communication (input and output) over SPI. Flow volume and error can be provided also over standard two-wire pulse interface.
- Prepared for two-point calibration (zero flow and high flow @ room temperature) in series production.

- First-hit level (FHL) regulation, various modes.
- Phase shift detection and phase jump correction for ToF
- Bubble detection and error detection included to recognize wrong measurements.
- Entry points for custom code.

Hardware related:

- Firmware NVRAM usage (of 4 kB available): 2.9 kB. 546 B for phase shift.
- RAM usage (of 176 x 32-bit words):
  - 109 words permanently used, including 27 words frontend data buffer (FDB).
  - 21 words free / unused 19 words free for temporary use.
- Firmware data usage (of 128 x 32 words):
  - 11 words configuration (always) plus protected ones.
  - 96 words used, including 16 words for bootloading.
  - 7 words free, up to 27 depending on linearization & calibration.
- Expected total current consumption including the flow calculation and 2 V/V gain at 3.0 V (example):
  - About 13 µA @ 8 Hz flow measuring rate, about 5 µA @ 2 Hz flow measuring rate.

For any information related to the hardware of the AS6031 like configuration registers, system handling registers, error flags and general operating principle please refer to:

AS6031 datasheet SC-000853-DS.

2.1 Memory Allocation

The ScioSense firmware uses memory in the following way:

- The firmware user code (FWU) is the main program, located in the lower part of the NVRAM. It is open sources and can be customized. It is limited in size by the ScioSense code. Register PROTECT_LIM shows the limit, e.g., 1216 (address 0x4C0) bytes in case of ScioSense applied firmware revision A1A10314.
- The applied ScioSense code (FWA) calls subroutines in the user code (FWU). FWA takes roughly 3 k of NVRAM. This part is closed and write/read protected.
- Many support functions are coded in ROM. This increases the efficiency of the programming. In addition, the ROM starts with a check for the CPU request and a bootloader section.
- The firmware data hold the individual data, namely the non-linear correction coefficients, the 2-point calibration coefficients, other firmware variables and parameters as well as the configuration of the chip.
- In the RAM, the original front end buffer data are stored as well as the final data calculated by the flow firmware. Of course, several cells are used for filter functions and temporary variables.

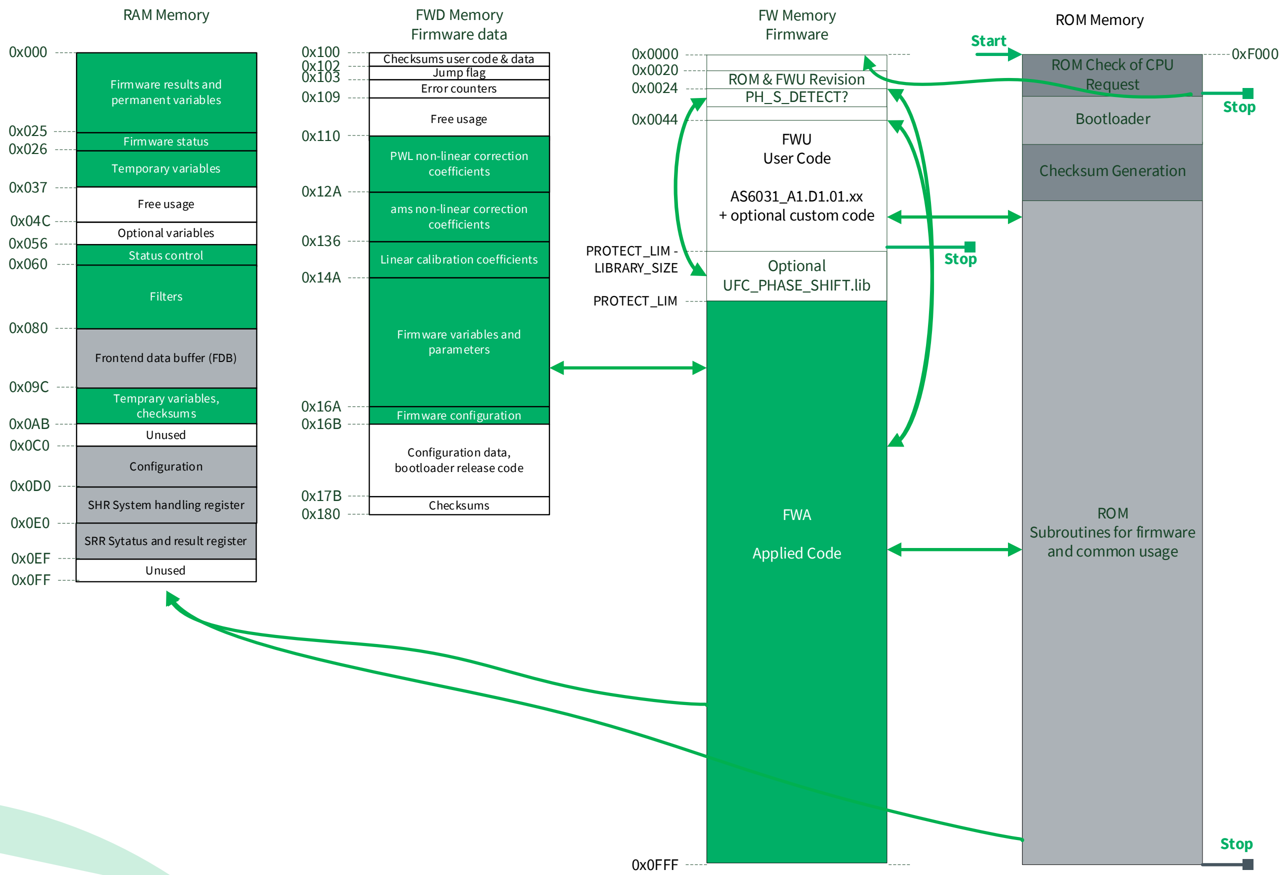


Figure 2: Memory allocation

2.2 Major Program Structure

The CPU gets called with every measurement. The main procedures are:

- First, the reason for the call is checked. ROM routine ROM_CPU_CHK is called, which checks the CPU request flags.
- Next, bootloader flag and checksum generations flags are checked by ROM routines.
- From there, the open user code starts at address 4, with checking firmware data cell 2 for bit 0. If this is zero, then only the user code is executed, and the CPU stops at the end. If this bit is 1 then the applied ScioSense code is started.
- The applied code has four indirect jumps into the open user code.
  - At the very beginning
  - After error handling
  - After postprocessing
  - For general purpose handling
- Following the first jump at the beginning, the applied firmware does some basic initialization and restoration as well as basic operation check routines. Subtasks of these check-and-resume routine are:
  - Refresh permanent RAM data by an hourly recall. Solves issues by corrupted data.
  - Check initialization and restore if needed.
  - Keep data and flags of running operation.
  - Check whether ROM is powered.
  - Check initialization and re-initialize if needed.
  - Break infinite loops.
  - Restore constants after a recall.
  - Restore the operation state after a recall.
- Before postprocessing the applied code jumps into the open part for the phase jump detection and correction at address 36. This part is executed only when the library is included.

The main post processing routines do the following:

- Low-level error handling (hardware level).
  - Handle and clear all hardware error flags / refresh copy in firmware error register.
  - Set / reset error counters.
  - Erase / replace invalid results.
  - Invalid flow results are replaced by valid predecessors (up to 8).
  - Invalid TM results are cleared immediately, with calculation results replaced by 0x7FFFFFFF (last valid results kept elsewhere).
- Post-processing subroutines

The whole process is controlled by interrupt requests issued by the chip's measurement rate generator (MRG) and the task sequencer (TS).

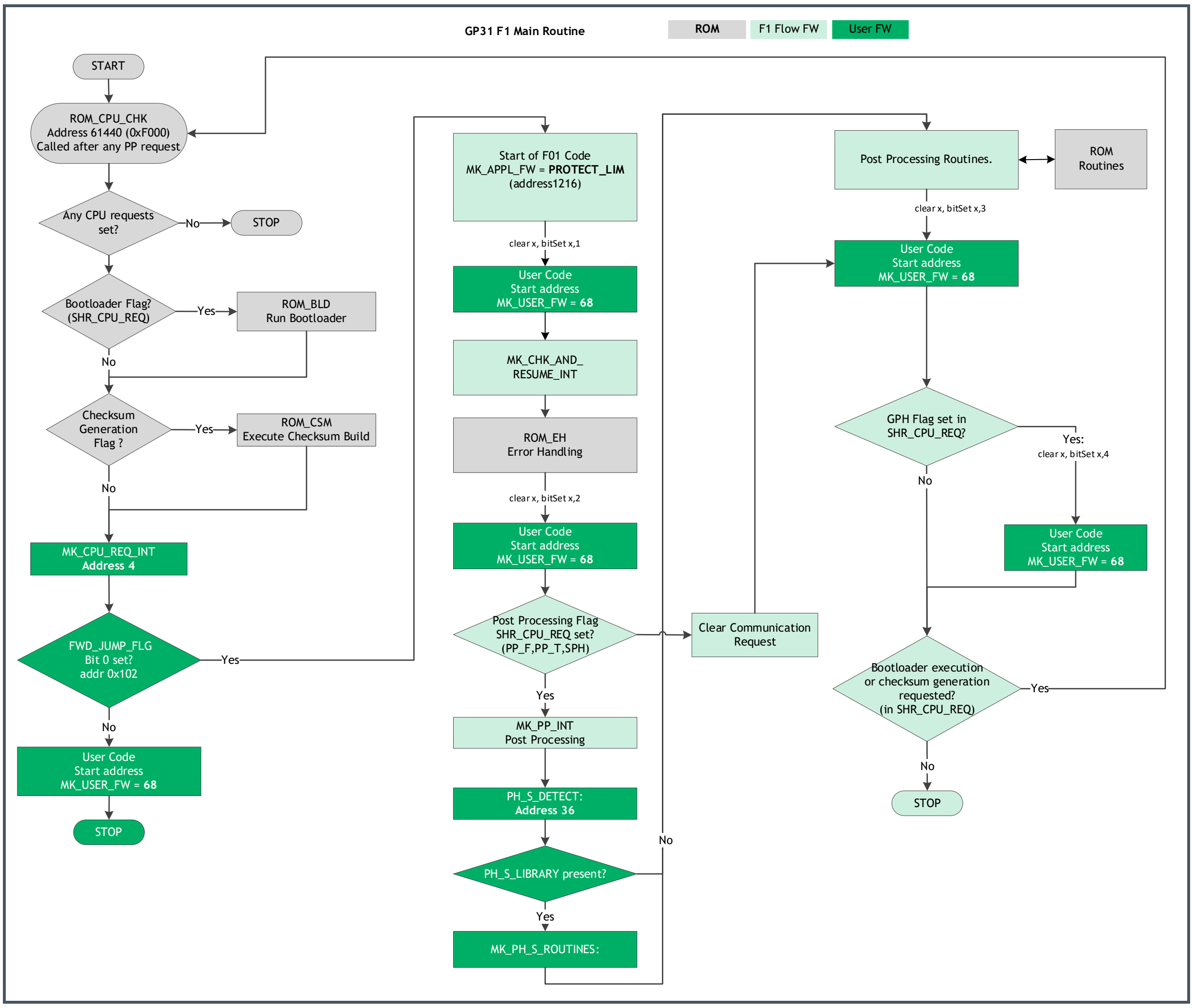


Figure 3: Flow chart, top level

2.3 Post Processing Subroutines

The most important routines are the post processing routines, therefore some more details on those:

- Temperature calculation
  - The sum TOF_UP + TOF_DOWN is used to calculate the speed of sound. The physical relation between speed of sound and temperature is then used to calculate the temperature, limited to the range < 60 °C.
  - Calculating temperatures based on the RDC measurements with the internal sensors and the external cold and hot sensors. This includes the compensation measurements in 2-wire and 4-wire mode.
- Flow calculation.
  - First hit level monitoring and regulation, based on pulse width ratio or amplitude variation.
  - Filtering (rolling (8) average of DIF_TOF and SUM_TOF).

- Zero flow detection (stage 1: zero flow will skip flow calculation for power saving reasons).
  - Flow speed calculation (linear and non-linear correction).
  - Mean (16) and rolling (16) average.
  - Zero flow detection (stage 2).
  - Volume calculation.
- High-speed clock calibration.
- Amplitude measurement and calibration.
- Pulse interface data generation.

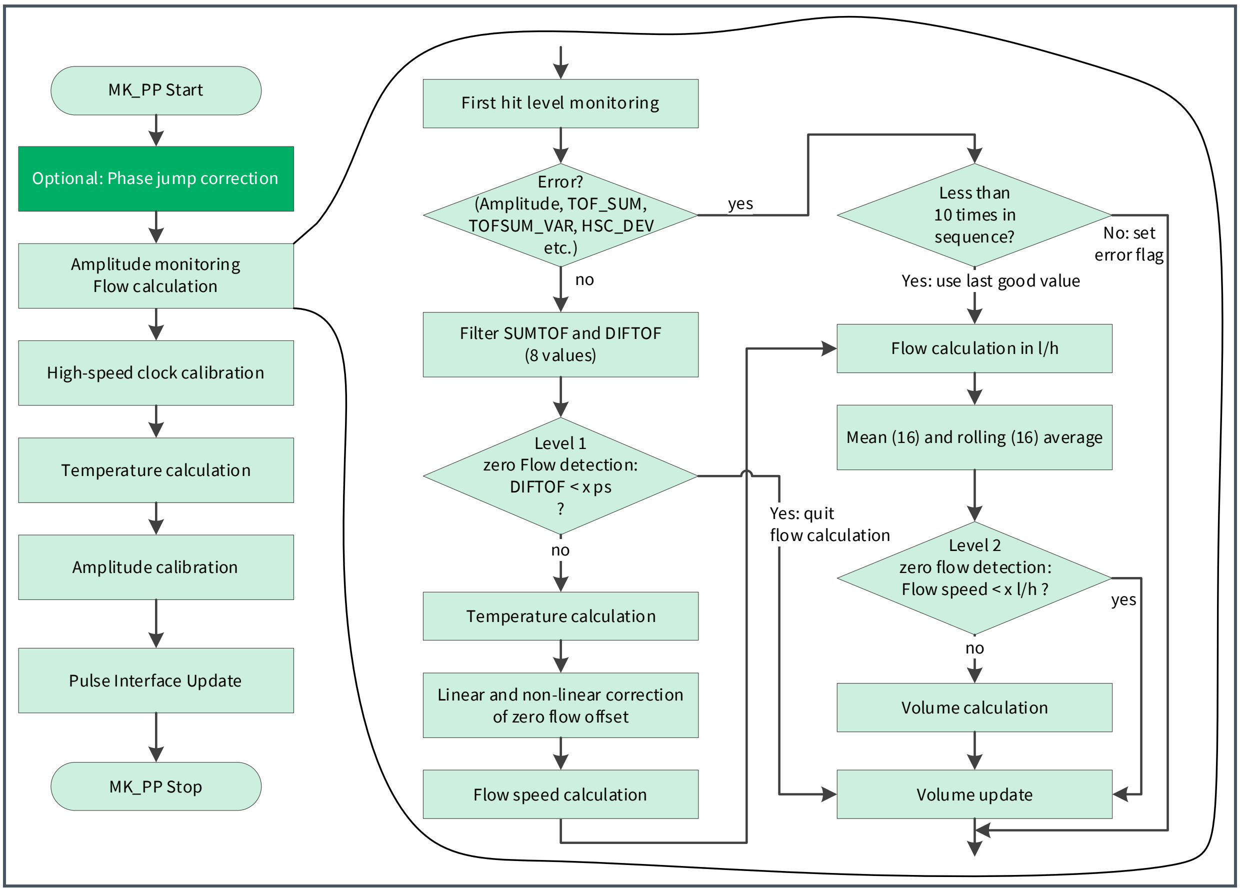
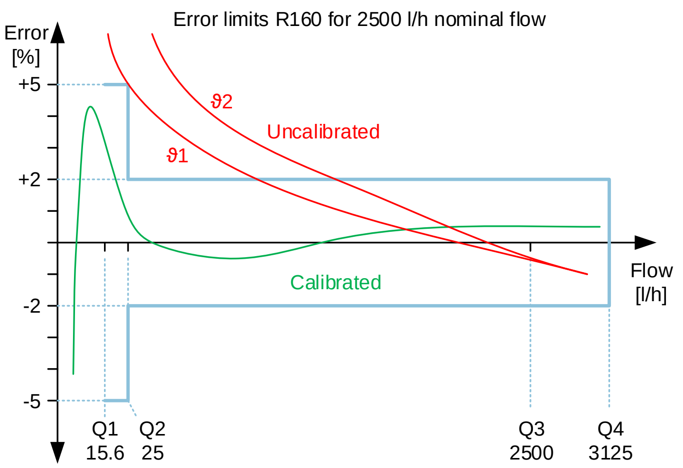


Figure 4: Main post processing routine

The phase jump detection and correction of the TOF UP and TOF DOWN are implemented in an open user code snippet and a related library file. See section 4 for details.

3 Linearization and Calibration

The firmware converts the difference in time-of-flight up and down (DIFTOF) into a flow and volume information. Ideally, the relation between DIFTOF and flow speed would be linear, but in addition to a temperature dependance of the linear relation, there is also a non-linear deviation that is related to the hydraulics of the specific sensor design. The F01 firmware concept assumes that the non-linear behavior is more or less the same over one or more production lots and that the individual meters can be corrected linearly at two points.



Q1 Minimum flow rate
Q2 Transitional flow rate
Q3 Nominal flow rate
Q4 Maximum flow rate

Figure 5: Non-linear correction

Linearization is done in three major steps:

1. Offset correction. This is necessary to get a correct sum of time-of-flight up and down (SUMTOF) which then is used to calculate the speed of sound and the temperature, which is then used for the temperature correction.
2. Linear correction over 2 points. This is the rough, linear correction for the individual meter, with offset and slope values for tree temperature ranges.
3. Non-linear correction. The firmware offers two options for this: a classical piecewise linear correction (PWL) or a ScioSense specific one that corrects for deviations to one direction at low flow very effectively.

The non-linear coefficients are generated during development phase and typically fix for a product or at least a production batch. In production, the coefficients for the 2-point linear correction and the offset are updated individually. But even individual PWL coefficients could be adjusted with some additional computing.

All calibration parameters are generated by the calibration engine (new engine in preparation). Cells number 58 and 62 to 73 have to be adapted to individual spool pieces by 2-point calibration. Note that the address range for PWL coefficients can be selected through B0 of ***FWD_FW_CONFIG*** to optimize memory space needs. Usually only one calibration method is used, and the coefficients of the other one are not stored. But to some extend it is possible (if there is still enough space for all PWL coefficients) to have both types of calibrations stored, such that a simple comparison can be done just by switching between ScioSense and PWL method.

In step1 the parameters for a correct offset and in the following the speed of sound and temperature calculation are set. The distances with flow and with no flow describe the mechanical design. The SUMTOF offset is collected as a first measurement in 2-point calibration. The DIFTOF at high flow is used by the ScioSense algorithm only.

In step 2 the DIFTOF is corrected in a linear manner, means by an offset O(T) and a scaling factor F(T). For the offset, the calibration engine output is base offsets O(TCx) and slopes (S_O(TCx) for three temperatures and allows a precise offset interpolation:

$O(T) = O(TCx) + S\_O(TCx)*(TCx - TC_{measured}) / f_{ref}$

A simplified approach would use only one offset and slope factor for the complete temperature range.

The measured DIFTOF is corrected by this offset O(T).

In step 3, this DIFTOF is further corrected by the non-linear parameter, e.g. with PWL by subtracting the PWL output.

In step 4, the scaling factor from the linear 2-point correction is applied. The procedure for the scaling factor is similar, base factors F(TCx) and slopes (S_F(TCx) for three temperatures allow a precise scale factor interpolation:

$F(T) = F(TCx) + S\_F(TCx)*(TCx - TC_{measured}) / f_{ref}$

The DIFTOF is corrected by this offset slope F(T), multiplied with the speed of sound and divided by the active length to get the flow velocity.

3.1 Piece Wise Linear (PWL)

The PWL coefficients are generated by means of the calibration engine that can be downloaded from https://downloads.sciosense.com/AS6031/. This engine offers some flexibility with respect to the number of calibration points, but total number and size is limited by the available space foreseen in the firmware data. By default, the address range 16 to 41 is foreseen. The starting address is set in cell 106, bits 7:0, **PWL_ADR**. Without restrictions it can be set to 7. It can be set also to 3, but then the error counters cannot be activated. The range stops normally at address 41. But also the range for the ScioSense coefficients can be used which ends at address 53. In total we have 25 to 50 32-bit words available.

The format of the coefficients can be set to 4, 8, 16 and 32 bits. With 32 bit values the number is quite limited, so 16 bit or even 8 is the choice for high numbers of coefficients, of course with loss in precision. This format is defined in the calibration engine.

In general, the organization of the PWL coefficients in the firmware data is defined in the first word of PWL firmware data, ***FWD_COEFF_ORG***:

- Bits [31:30]: DIF_TOF coefficient size, 4, 8, 16 or 32 bits.
- Bits [29:28]: 8-bit temperature values, position after decimal point, 0, 1, 2 or 3.
- Bits [23:16]: Number of temperature points
- Bits [7:0]: Number of DIF_TOF points

Note: The PWL linearization algorithm does not cover handling of negative coefficients. When the calibration engine output includes negative coefficients, all coefficients have to be increased by the absolute value of the smallest negative coefficient so that all are positive. Those corrected coefficients can then be written into the firmware data. The value of the smallest negative value is put into FWD cell 88, ***FWD_CORR_COEFF***.

DIFTOF	T1	T2	T3	T4		Minimum:		DIFTOF	T1	T2	T3	T4
0	19	29	40	50		-1,0568938		0	19	29	40	50
22	1,7922297	1,113182	0,73147691	1,6010345				22	2,849123	2,170075	1,788371	2,657928
46	1,82892941	1,486948	1,30713696	-0,2465813	--->	Add 1,0568938 --->		46	2,885823	2,543842	2,364031	0,810312
99	2,1624493	1,510421	1,55300598	-1,0568938				99	3,219343	2,567315	2,6099	0
211	2,20838017	1,438918	0,6066375	1,1991559				211	3,265274	2,495812	1,663531	2,25605
450	0	0	0	0				450	1,056894	1,056894	1,056894	1,056894

Figure 6: Example negative coefficients

FWD_CORR_COEFF = $T_{ref}\ 2^{16}/-1.0568938 = -277.058$ = 0xFFFFFEEA

T_{ref} = Time period of high-speed clock

4 Phase Shift Detection & Jump Correction

4.1 Functionality

AS6031 has the possibility to fire a split burst with a phase shift of typically $^{3}/_{4}$ T between initial and ending sequence.

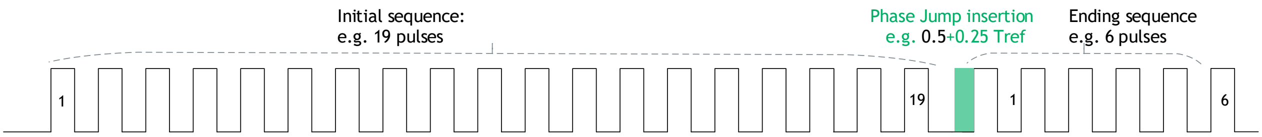


Figure 7: Split fire burst

The idea is to have the jump after the hits that are used for ToF measurement. The content of the frontend data buffer is adjusted and shows only the ultrasonic TOF sum of all values Up and Down, **FDB_US_TOF_SUM_OF_ALL_x**. The individual ToF in the RAM show the values for the very first three hits measured, the three hits following the original measurement and four hits that are exposed to the phase shift.

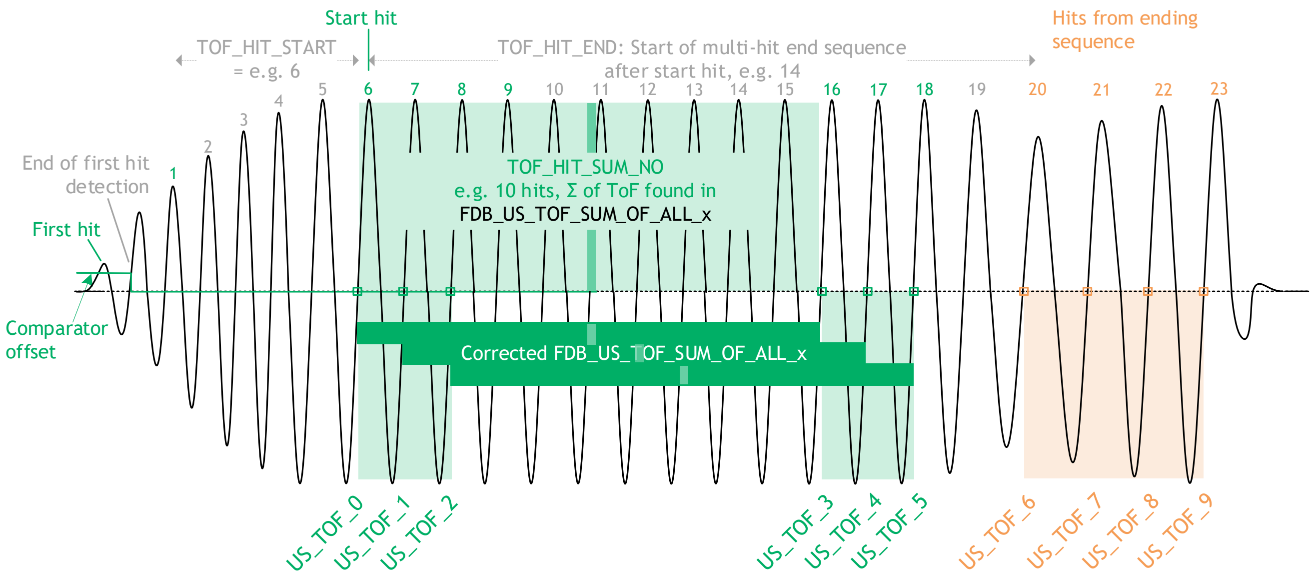


Figure 8: Phase shift detection and jump correction

The hits of the ending sequence are used to find the position of the phase jump. Therefore, they are compared with the theoretically expected time stamps without phase jump. US_TOF_3 is taken as basis and multiples of the period are added for the expected times. The differences are temporarily stored in RAM cells RAM_Vxx_PH_S_xx_x. The biggest delta is the taken as a reference and used to decide whether the position of this jumped into one or the other direction.

The ToF result can be corrected mathematically. In the normal case, the result from the front-end data buffer is corrected by US_TOF_1 and US_TOF_4 to the center position. In case of a phase jump it shifted towards US_TOF_0/ US_TOF_3 or US_TOF_2/ US_TOF_5.

Normal:						Jump right:		Jump left:	
37	RAM_V2F_PH_S_DOWN_0	000006B2	0.0038147	6.538	ns	0.294	ns	42.362	ns
38	RAM_V30_PH_S_DOWN_1	00002AF8	0.0038147	41.962	ns	6.966	ns	177.998	ns
39	RAM_V31_PH_S_DOWN_2	0000B48E	0.0038147	176.323	ns	42.023	ns	317.951	ns
3A	RAM_V32_PH_S_DOWN_3	00014502	0.0038147	317.391	ns	176.693	ns	340.317	ns

Figure 9: Example of temporary phase deviations

The major parameter in the firmware data to be defined by the application is the parameter ***FWD_PH_S_THRESHOLD***. This is the absolute value of phase-shift threshold value (no negative values here). The format is fd16 in multiples of the reference period (value/2^{16} * T_{ref}[ns]).

In the example above this threshold could be e.g. 80ns. If the value of ***RAM_V31_PH_S_DOWN_2*** falls below that value then there was a jump to the right. If ***RAM_V30_PH_S_DOWN_1*** is bigger than that threshold there was a shift to the left.

The measured TOF results in FDB 0x080 and 0x084 will be corrected in the following manner:

- No jump: SUM_OF_ALL_U = SUM_OF_ALL_U - TOF_0_U + TOF_3_U
- Jump right: SUM_OF_ALL_D = SUM_OF_ALL_D
- Jump left: SUM_OF_ALL_D = SUM_OF_ALL_D - TOF_0_U + TOF_3_U - TOF_1_D + TOF_4_D

4.2 Implementation

The phase shift detection and jump correction routine is not hard-coded but needs to be implemented in the open user code. The necessary line in the code will be
`#include "UFC_PHASE_SHIFT.lib"`.

Sample code is provided by ScioSense as a library file. The ScioSense flow firmware jumps to address 36 to execute the phase shift routine in advance to the further flow calculation.

The flow firmware always jumps to label PH_S_DETECT. If the phase-shift library has been included the code inside the #ifdef PH_S_LIBRARY will get compiled and executed at each jump. If the phase-shift library hasn't been included, then the code line jsubret will get compiled and the code will return from jump immediately.

```
org    32
       equal1 FW_ROMVERSION_REV
       equal  FW_VERSION           ; Defined at the beginning of this file

org 36
PH_S_DETECT:
#ifdef PH_S_LIBRARY
       jsub   MK_PH_S_ROUTINES     ; jump to call every phase-shift routine
       jsubret
#else
       jsubret
#endif

org 64
MK_USER_FW:
```

The library routine has sub routines for

- Initialization (TOF_HIT_NO, period)
- Consistency check for the individual ToF results
- Calculation of the deviations
- Check for the presence of a phase jump
- Correction of the TOF_SUM, overwriting the original results in the frontend data buffer, addresses 0x080 and 0x084.

Note: In case the library is not included the code jumps back immediately for normal flow calculation. But the address range 36 to 68 is reserved so that it is possible to have custom code, other than the jump to the phase shift routine.

5 Firmware Input Data (FWD)

Most of the values in various cells of the firmware data should be customized or at least checked. Section 4.1 gives a complete overview of all firmware data parameters. The quickest start with ScioSense firmware may be checking the parameter table there and use it to customize the template firmware data file provided by ScioSense. If necessary, more detail information can be found in the following sections, which discuss the overview in different segments, ordered for their particular function with focus on the relevant bits.

The major register for operation control is ***FWD_FW_CONFIG*** (cell number 106). Its control bits are summarized in section 5.2.

5.1 Parameters in ScioSense Firmware Data (FWD)

The following list is a complete overview of FWD memory cell usage by the ScioSense firmware. For the format definition, see the appendix on notational conventions.

All variable names and address definitions can also be found in file AS6031_Flow_FW_A1A1.h, contained in the evaluation package. For the physical addresses, add the cell number to base address 0x100.

Table 1: Firmware data (FWD)

Cell[1]	Variable Name	Function	Description *default value (if applicable)*	
0	*FWD_FWU_CS*		Checksums	User code
1	*FWD_FWDU_CS*			User data
2	*FWD_JUMP_FLAG*		Flag to trigger jump from ScioSense code to address 68 of the user code. Bit 0 = 0: Direct jump to address 68 Any Bit = 1: Indirect jump to address 68	
3	*FWD_ERROR_COUNT_CONF1*	(optional) Error counters	Define error flag positions to be counted in error counter 1	
4	*FWD_ERROR_COUNT_CONF2*		Define error flag positions to be counted in error counter 2	
5	*FWD_ERROR_COUNT_21*		Temporary storage of error counts 2 (B3, B2) and 1 (B1, B0) Set to 0x00000000 if error counters are used	
6	*FWD_ERROR_COUNT_43*		Temporary storage of error counts 4 - each error (B3, B2) and 3 - hardware errors (B1, B0) Set to 0x00000000 if error counters are used	
7	*FWD_ERROR_COUNT_INV21*		Inverse of 0x105 for constant checksum	

[1] Address = Cell number + offset 0x100

8	*FWD_ERROR_COUNT_INV43*		Inverse of 0x106 for constant checksum	
9 to 15	*NOT USED*		Not used	
16 to 41	*PWL COEFFICIENT TABLE*	Non-linear correction	(optional) Typical position of PWL calibration coefficients table, generated by cal. engine	
42 to 53	*SCIOSENSE COEFFICIENT TABLE*		(optional) Fixed position for ScioSense calibration coefficients table, generated by cal. engine	
54 to 73	*LINEAR COEFFICIENTS TABLE*		Linear coefficients table, all generated by cal. engine; values in cells # 58 and 62 – 73 have to be adapted to individual spool pieces by 2-point calibration	
54	*FWD_R_TEMP_TC1*	Linear correction (2-point)	Temperatures for linear calibration in °C Format fd16	1^{st}
55	*FWD_R_TEMP_TC2*			2^{nd}
56	*FWD_R_TEMP_TC3*			3^{rd}
57	*FWD_R_TEMP_TC4*			4^{th}
58	*FWD_R_TOF_OFFSET*		Offset time for SUMTOF (fd16*T_{ref})	
59	*FWD_TOF_DIFF_CAL*		DIFTOF at high flow calibration point (fd16*T_{ref})	
60	*FWD_DIST_WITH_FLOW*		Ultrasonic sound path length along flow in m (fd24)	
61	*FWD_DIST_NO_FLOW*		Ultrasonic sound path length w/o flow in m (fd24)	
62	*FWD_R_ZERO_OFFSET_TC2*		Zero flow DIFTOF O(TC) Format fd32	at TC2
63	*FWD_R_ZERO_OFFSET_TC3*			at TC3
64	*FWD_R_ZERO_OFFSET_TC4*			at TC4
65	*FWD_R_O_SLOPE_TC12*		Zero flow slope S_O(TC) Format fd16	between TC1 and TC2
66	*FWD_R_O_SLOPE_TC23*			between TC2 and TC3
67	*FWD_R_O_SLOPE_TC34*			between TC3 and TC4
68	*FWD_R_F_SLOPE_TC12*		Proportionality factor slope S_F(TC) Format fd16	between TC1 and TC2
69	*FWD_R_F_SLOPE_TC23*			between TC2 and TC3
70	*FWD_R_F_SLOPE_TC34*			between TC3 and TC4

71	*FWD_R_F_OFFSET_TC2*		Proportionality factor F(TC) Format fd16	at TC2
72	*FWD_R_F_OFFSET_TC3*			at TC3
73	*FWD_R_F_OFFSET_TC4*			at TC4
74	*FWD_SOUND_VEL_MAX*	Medium related	Maximum of speed of sound in m/s Default value for water 0x00061400 Format fd8	
75	*FWD_1_BY_A*		Medium constant, used for T calculation on basis of the speed of sound. Default value for water 0x002CA2E2	
76	*FWD_CONST_C*		Medium constant, used for T calculation on basis of the speed of sound. Default value for water 0x000F6C3A	
77	*FWD_THETA_MAX*		B3/B2/B1: Temperature at maximum speed of sound in °C, format fd8. B0: minimal speed of sound, format fd5. Default value for water 0x004A002B	
78	*FWD_LONG_TERM_ERROR*		Number of low AM measurements before hardware failure / no-water checks are done Proposed value 0x00000020	
79	*FWD_FHL_USER*		B1/B0: trusted FHL ratio (option B), format fd16, or B0: absolute trusted FHL, LSB=0.88 mV, format fd0.	
80	*FWD_TOF_SUM_DELTA*		FHL method 3: Nominal time difference (raw TDC units) in SUMTOF between operating and trusted FHL	
81	*FWD_TOFSUM_VAR_LIM*		Error limit for deviation of SUMTOF from former average (raw TDC units)	
82	*FWD_HSC_DEV*		Firmware data memory with maximum permissible deviation in the HS clock from 4 LS clock periods (250 or 125 ns) * 2^{16}	
83	*FWD_ERR_INTERRUPT*		Define error flag positions that issue an interrupt. . Same bits as RAM_R_FW_ERR_FLAGS1	
84	*FWD_AM_DIFF_LIM*		Error limit for deviation between currently measured amplitude UP and DOWN in mV, format fd16	
85	*FWD_R_AM_MIN*		Minimum allowed amplitude in mV, format fd16, must be set to a realistic value, lower than in normal operation, but higher than in no water/hardware error condition	
86	*FWD_PW_NOM*		Nominal pulse width ratio with fd7 format. Set to zero to switch off pulse width regulation (0x54 before)	

87	***FWD_PW_DEV***		Error limit for deviation between currently measured UP and DOWN pulse width, format fd7
88	***FWD_CORR_OFFSET***		Offset correction in case of negative PWL coefficients; negative value in raw TDC format fd16
89	***FWD_TOF_RATE_FACTOR***		Factor for TOF rate scaling in zero flow case
90	***FWD_FLOW_AVG_FACTOR***		2^N number of flow values for averaging; this factor *16 determines the total number of samples for long term averaged flow, as used for the zero-flow decision.
91	***FWD_R_PULSE_PER_LITER***		Pulse interface: Number of pulses per liter
92	***FWD_R_PULSE_MAX_FLOW***		Pulse interface & maxflow error limit: maximum permissible flow in l/h.
93	***FWD_NEG_FLOW_LIMIT***		Cut-off limit for negative flow in l/h, format fd16; positive values are ignored
94	***FWD_R_TOF_DIFF_LIMIT***		Minimum limit for DIFTOF values in raw TDC units. At lower |DIFTOF|, temporary zero flow is assumed and no calculation is done
95	***FWD_ZERO_FLOW_LIMIT***		Zero flow limit in l/h, format fd16: When the absolute of the long-term average flow is smaller, long term zero flow is assumed and the TOF rate is scaled by ***FWD_TOF_RATE_FACTOR***
96	***FWD_CAL_PTR_OFFSETR***	Temperature measurement	Reference branch offset resistance in internal reference in Ω, format fd16; typical value 0 (calibrate if desired)
97	***FWD_EXT_REF_VAL***		Value of reference resistor R_{ref} in Ω, format fd16; typical value for PT1000: 1kΩ =0x03E80000
98	***FWD_PT_INT_SLOPE***		Internal sensor resistance slope in (K/Ω)*Rref; format fd13; typical value 0x0029F000 (calibrate if desired)
99	***FWD_PT_INT_NOM***		Internal sensor nominal resistance with fd 16; typical value 0x03C20000 (calibrate if desired)
100	***FWD_PTC_RATIO_INV***		Nominal ratio of reference resistor to PT cold sensor resistance at 0°C; format fd16; typical value 1 = 0x00010000
101	***FWD_PTH_RATIO_INV***		Nominal ratio of reference resistor to PT hot sensor resistance at 0 °C; format fd16; typical value 1 = 0x00010000
102	***FWD_HSC_CLOCK***	Phase shift	HSC clock in Hz, used as reference in the clock calibration 0x003D0900 = 4 MHz, 0x007A1200 = 8 MHz 0 is equivalent to 4 MHz

103	*FWD_MH_RLS_DLY*		Nominal value for release delay; calculated only when FWD_MH_RLS_DLY_LIM ≠0
104	*FWD_PH_S_THRESHOLD*		Absolute value of phase-shift threshold value. Cannot be a negative number, fd16 * T_{ref}(ns)
105	*FWD_MH_RLS_DLY_LIM*		Multihit release delay limits (min B1, B0. & max B3, B2.); average is initial value; 1 LSB = 62.5 ns
106	*FWD_FW_CONFIG*		ScioSense firmware configuration register
107	*FWD_FW_RLS*		Boot loader release. 0xABCD7654 activates the bootloading process after start-up
108	*FWD_R_CD*		Watchdog disable code - Without ScioSense firmware, 0x48DBA399 disables the watchdog while any other code enables it. - With ScioSense firmware, the watchdog can't be disabled, but 0x00000000 disables configuration restore after recall. Reset after POR should be applied. - Proposed value 0x95659C6A
109	*FWD_IFC_CTRL*	Initial values for configuration registers	Configuration data for **CR_IFC_CTRL**
110	*FWD_GP_CTRL*		Configuration data for **CR_GP_CTRL**
111	*FWD_USM_OPT*		Configuration data for **CR_USM_OPT**
112	*FWD_IEH*		Configuration data for **CR_IEH**
113	*FWD_CPM*		Configuration data for **CR_CPM**
114	*FWD_MRG_TS*		Configuration data for **CR_MRG_TS**
115	*FWD_TPM*		Configuration data for **CR_TPM**
116	*FWD_USM_PRC*		Configuration data for **CR_USM_PRC**; ScioSense firmware interprets B3 as TOF_RATE;
117	*FWD_USM_FRC*		Configuration data for **CR_USM_FRC**
118	*FWD_USM_TOF*		Configuration data for **CR_USM_TOF**
119	*FWD_USM_AM*		Configuration data for **CR_USM_AM**
120	*FWD_TRIM1*		Configuration data for **CR_TRIM1**; Set to 0x94A0C46C
121	*FWD_TRIM2*		Configuration data for **CR_TRIM2**; Set to 0x401100C7

122	*FWD_TRIM3*		Configuration data for **CR_TRIM3**; Set to 0x00A7400F	
123	*NOT USED*			
124	*FWD_R_FWD1_CS*	Checksums	Checksum firmware data 1	These checksums are stored for comparison to values calculated by the chip. Deviations in comparison cause checksum errors. Can be calculated by the UFC Evaluation Software.
125	*FWD_R_FWD2_CS*		Checksum firmware data 2	
126	*FWD_R_FWU_CS*		Checksum firmware code user	
127	*FWD_R_FWA_CS*		Checksum firmware code ScioSense	

5.2 Parameters & Flags for General Operation Control

The most important register is ***FWD_FW_CONFIG*** (firmware configuration), which controls most options for firmware operation.

Table 2: Firmware data (FWD)

Addr: 0x16A Cell: 106		FWD_FW_CONFIG (Firmware Configuration)		
Bit	Bit Name	Default	Format	Bit Description
7:0	**PWL_ADDR**	0b10	UINT [7:0]	(optional) Start address of PWL coefficients table in FWD (without leading address bit 8, which is always 1 in FWD addresses)[2]
13:8	**PWL_EXP**	0b000001	UINT [5:0]	(optional) Exponent of scaling factor for PWL coefficients: scale up each value by 2^(**PWL_EXP**)[2]
14	**NOT USED**	0b0	BIT	Set to zero
15	**BNR_FWCONF_PWL**	0b0	BIT	Enable PWL calibration
16	**BNR_FWCONF_2MAX_NOZERO**	0b0	BIT	0: Set flow to zero when exceeding 2x maximum flow and signal error 1: Flow remains even when exceeding 2x maximum flow, but error flag is set
17	**BNR_FWCONF_FHL_ZEROFLOW**	0b0	BIT	0: With FHL regulation active, disable zero flow state (always assume full flow as long as FHL is considered not ok) 1: Apply zero flow state independently of FHL regulation

[2] Generated by calibration engine

18	**BNR_FWCONF_FILTER**	0b0	BIT	Enable IIR filter for HSC factor and AM values
19	**BNR_FWCONF_NOENF _RECALL**	0b0	BIT	Don't enforce recall setting (recall still happens as configured) 0: recall settings are enforced to 1 s, 5 s,10 min or 1 h
20	**BNR_FWCONF_IND_RECALL0**	0b0	BIT	To be changed to 1 by FW after initialization due to recall; corresponds to bit 28, for constant checksums
21	**BNR_FWCONF_NO_PI_ERR**	0b0	BIT	0: Signal error on pulse interface as configured in bit 22 1: Never signal error on pulse interface
22	**BNR_FWCONF_PI_ERROR**	0b0	BIT	0: Don't signal no-water/hardware error as error on pulse interface 1: Also signal no-water/hardware error as error on pulse interface
23	**BNR_FWCONF_ERR**	0b0	BIT	Activate average error counters 0: No average error counters 1: Average error counters active
25:24	**BNR_FWCONF_FHL**	0b00	BIT	Configuration of FHL regulation methods 00: Method 1, fixed FHL 01: Method 2, consistency check against trusted FHL 10: Method 3, consistency check against trusted FHL with offset time 11: Method 4, consistency check against special configuration
26	**BNR_FWCONF_ FHL_RATIO**	0b0	BIT	Configuration for FHL regulation option B 0: Interpret FHL-values as fixed voltage 1: Interpret FHL-values as ratio to measured amplitude
27	**BNR_FWCONF_VLIM**	0b0	BIT	0: Disable control of speed of sound limits 1: Enable control of speed of sound limits
28	**BNR_FWCONF_IND _RECALL**	0b0	BIT	Activate recall initialisation, to be changed to 0 after initialization due to recall 0: No recall initialization due or no initialization configured 1: Initialization after recall Note: for constant checksum generation set this to 1 (correlates to bit 20)
29	**BNR_FWCONF_VOL**	0b0	BIT	Flow volume storage protection: flow volume is stored in three pairs of RAM cells, such that a correction can be applied. When all three data are different then an error flag is raised.

				0: Don't apply 1: Apply[3]
30	**BNR_FWCONF_TSENS**	0b0	BIT	0: Use temperature value from flow meas. for calibration coefficients 1: Use temperature value from hot sensor meas. for calibration coefficients
31	**BNR_FWCONF_ACAM**	0b0	BIT	0: Apply ScioSense calibration method 1: Disable ScioSense calibration method

5.3 Parameters for Hardware

The most convenient way to define and test the best hardware configuration is by using the ScioSense UFC Evaluation Software. From there the configuration data can be directly included into the firmware data file. With the right setting of the bootloader, the configuration will then be automatically loaded from the firmware data after a power-on reset.

A few remarks are necessary:

- The TOF rate is not stored in AS6031 configuration registers. The firmware uses B3 of cell 116 (*FWD_USM_PRC*) to define TOF rates above 1. TOF rate 0 is not permitted with unmodified ScioSense firmware.
- FHL values are not stored in AS6031 configuration registers. For FHL setting and regulation see section 7.
- With the ScioSense firmware the watchdog is always enabled. Cell 108 (*FWD_R_CD*), normally reserved for the watchdog disable code (0x48DBA399), has a different function.
  - 0x00000000 in cell 108: watchdog always enabled, the firmware does not refresh the configuration and other permanent settings hourly. This is helpful during development, when the configuration under test is frequently varied. With this setting a reset should be applied after a power-on reset.
  - 0x95659C6A in cell 0x108: recommended for production.
- One more configuration setting is enforced by the firmware, no matter what is written in CR_MRG_TS or *FWD_MRG_TS*: The checksum timer is always set to "hourly", such that every full hour a data recall is done. This restores the RAM part of the NVRAM memory. With cell 108 unequal 0x00000000, the firmware also restores the configurations and other permanent settings after recall. This is a safety measure in case RAM data was corrupted for any reason.
- In combination with ScioSense firmware the temperature measurement rate should be reduced, setting TM_RATE > 1.

Note: ScioSense firmware sues GPIO5 and GPIO6 for diagnostics. They cannot be used otherwise.

[3] Turn this bit off when the stored volume should be changed intentionally

5.4 Parameters for Medium

Four parameters are used to describe the velocity of sound in pure water. They also define acceptable velocity limits for error handling. These parameters should not be changed when operating with water. For other media, please contact support.

Table 3: Firmware data (FWD)

Cell	Variable Name	Description *default value (if applicable)*
74	***FWD_SOUND_VEL_MAX***	Maximum of speed of sound in m/s Default value for water 0x00061400 Format fd8
75	***FWD_1_BY_A***	Medium constant Default value for water 0x002CA2E2
76	***FWD_CONST_C***	Medium constant Default value for water 0x000F6C3A
77	***FWD_THETA_MAX***	B3/B2/B1: Temperature at maximum speed of sound in °C, format fd8. B0: minimal speed of sound, format fd5. Default value for water 0x004A002B

5.5 Parameters for Calibration

All calibration parameters are generated by the calibration engine, see calibration engine user guide SC-001279-UG. Cells number 58 and 62 to 73 have to be adapted to individual spool pieces by 2-point calibration. Note that the address range for PWL coefficients can be selected through B0 of ***FWD_FW_CONFIG*** to optimize memory space needs. Usually only one calibration method is used, and the coefficients of the other one are free for other use. But it is also possible to have both types of calibrations stored and even enabled.

In step1 the parameters for a correct offset and in the following the speed of sound and temperature calculation are set. The distances with flow and with no flow describe the mechanical design. The SUMTOF offset is collected as a first measurement in 2-point calibration. The DIFTOF at high flow is used by the ScioSense algorithm only.

Table 4: Firmware data step 1

Cell	Variable Name	Function	Description *default value (if applicable)*
58	***RAM_TOF_OFFSET***		Offset time for SUMTOF in raw TDC units
59	***FWD_TOF_DIFF_CAL***		DIFTOF at high flow calibration point in raw TDC units
60	***FWD_DIST_WITH_FLOW***		Ultrasonic sound path length along flow in m (fd24)
61	***FWD_DIST_NO_FLOW***		Ultrasonic sound path length w/o flow in m (fd24)

In step 2 the DIFTOF is corrected in a linear manner, means by an offset O(T) and a scaling factor F(T). For the offset, the calibration engine output is base offsets O(TCx) and slopes (S_O(TCx) for three temperatures and allows a precise offset interpolation:

$O(T) = O(TCx) + S\_O(TCx)*(TCx - TC_{measured}) / f_{ref}$

A simplified approach is using only one offset and slope factor for the complete temperature range.

The measured DIFTOF is then corrected by this offset.

For the scaling factor the output is similar, base factors F(TCx) and slopes (S_F(TCx) for three temperatures and allows a precise offset interpolation:

$F(T) = F(TCx) + S\_F(TCx)*(TCx - TC_{measured}) / f_{ref}$

This scaling factor is applied to the finally calculated flow value.

Table 5: Firmware data linear correction[4]

Cell	Variable Name	Function	Description *default value (if applicable)*	
54	***FWD_R_TEMP_TC1***	Linear correction (2-point)	Temperatures for linear calibration in °C Format fd16	1st
55	***FWD_R_TEMP_TC2***			2nd
56	***FWD_R_TEMP_TC3***			3rd
57	***FWD_R_TEMP_TC4***			4th
62	***FWD_R_ZERO_OFFSET_TC2***		Zero flow DIFTOF O(TC) Format fd32	at TC2
63	***FWD_R_ZERO_OFFSET_TC3***			at TC3
64	***FWD_R_ZERO_OFFSET_TC4***			at TC4
65	***FWD_R_O_SLOPE_TC12***		Zero flow slope S_O(TC) Format fd16	between TC1 and TC2
66	***FWD_R_O_SLOPE_TC23***			between TC2 and TC3
67	***FWD_R_O_SLOPE_TC34***			between TC3 and TC4
68	***FWD_R_F_SLOPE_TC12***		Proportionality factor slope S_F(TC) Format fd16	between TC1 and TC2
69	***FWD_R_F_SLOPE_TC23***			between TC2 and TC3
70	***FWD_R_F_SLOPE_TC34***			between TC3 and TC4
71	***FWD_R_F_OFFSET_TC2***		Proportionality factor F(TC) Format fd16	at TC2
72	***FWD_R_F_OFFSET_TC3***			at TC3
73	***FWD_R_F_OFFSET_TC4***			at TC4

In advance to the flow calculation, a non-linear correction is applied to the DIFTOF. The choice is between the ScioSense proprietary method and a classical piecewise linear correction. The major choices are made in the firmware configuration register, see section 5.2.

- PWL_ADDR
  (optional) Start address of PWL coefficients table in FWD (without leading address bit 8, which is always 1 in FWD addresses)
- PWL_EXP
  (optional) Exponent of scaling factor for PWL coefficients: scale up each value by $2^{\wedge \mathrm{PWL\_EXP}}$

[4] All those parameters are generated by the calibration engine. See User Guide UG403 for details.

- BNR_FWCONF_TSENS
  0: Use temperature value from flow meas. for calibration coefficients
  1: Use temperature value from hot sensor measurement for calibration
- BNR_FWCONF_PWL
  0: Apply ScioSense calibration method
  1: Enable PWL calibration method

All the non-linear correction coefficients will be generated by the calibration engine.

5.6 Parameters for Zero Flow and Negative Flow Operation

In zero flow case, the ScioSense firmware can switch to an operation mode with reduced measurement rate and thus reduced current consumption. The following four parameters control this operation: The raw DIFTOF value in cell 94 is a limit for skipping calculation, with an absolute DIFTOF below that value no calculation is done and the flow is considered zero. In contrast, the limit in cell 95 is given as a flow, for example 0.5 l/h. If the actual averaged flow is below this value, the chip switches to zero flow operation: The TOF rate is increased by the factor defined on cell 89. In other words, the measurement rate is reduced by this factor. As soon as the average flow exceeds the value in cell 95, the chip switches back to full operation. Or, at the first single measurement (not averaged) which exceeds the value in cell 95 by a factor of 8, the chip also immediately returns to full operation. The flow averaging factor in cell 90 defines the range of averaging (actual averaging filter length is 16 times the value in cell 90, which must be a power of 2). This way the level of remaining noise after averaging can be controlled. A value of 0x00000010 = 16 is proposed, which means average flow is calculated as the arithmetic mean over the last 16 * 16 = 256 measurements.

If a negative flow is defined in cell 93, any reverse flow exceeding this value is not added to flow volume but stored in cells 0x04F to 0x051 (see section 6.1). If no reverse flow should be counted, this value is typically set to a small negative flow. Don't set it too small, else negative values from noise at zero flow will not average out to zero but will leave some positive offset in flow volume.

If a negative flow limit is used, cells 0 and 1 are internally utilized for intermediate values and can't be used for other purposes.

Table 6: Firmware data zero and negative flow

Cell	Variable Name	Function	Description default value (if applicable)
89	*FWD_TOF_RATE_FACTOR*		Factor for TOF rate scaling in zero flow case
90	*FWD_FLOW_AVG_FACTOR*		Number of flow values for averaging is $2^{FWD\_FLOW\_AVG\_FACTOR}$; $2^{FWD\_FLOW\_AVG\_FACTOR}$ * 16 determines the total number of samples for long term averaged flow, as used for the zero-flow decision (average+mean).
93	*FWD_NEG_FLOW_LIMIT*		Cut-off limit for negative flow in l/h, format fd16; positive values are ignored

94	***FWD_R_TOF_DIFF_LIMIT***		Minimum limit for DIFTOF values in raw TDC units. At lower |DIFTOF|, temporary zero flow is assumed, and no calculation is done
95	***FWD_ZERO_FLOW_LIMIT***		Zero flow limit in l/h, format fd16: When the absolute of the long-term average flow is smaller, long term zero flow is assumed and the TOF rate is scaled by ***FWD_TOF_RATE_FACTOR***

5.7 Parameters for Error Handling

Proper error handling is an important feature since situations where no normal measurement is possible will appear regularly, may it be no-water or bubbles. In addition, suitable error handling increases operation reliability in long term operation, which is typical for flow meters. Due to its importance, this topic is discussed in chapter 7 in detail. The current section just lists the parameters which have influence on error handling. Note that all error handling activities can be switched off either by setting the limit to zero or, when this is impossible (like in cells 74, 77 and 92), through a bit setting in ***FWD_FW_CONFIG***. See end of this section for details.

The parameters in cells 74 and 77 (fixed for water) define upper and lower limits for the acceptable speed of sound. Calculated results beyond these limits indicate wrong measurements and are thus ignored, the chip keeps the last valid result instead.

Cell 81 defines a variation limit for the current SUMTOF, compared to the average of the last 8 measurements. SUMTOF changes slowly with water temperature, such that a sudden change in value can be considered wrong. A typical reason for jumping SUMTOF values is detecting a wrong first hit (see chapter 7). It is therefore a good idea to set cell 81 to, for example, half a period of the measurement frequency, to detect such kinds of errors (at 1 MHz measurement frequency for example 0x0001FFFF = 0.5 µs). Setting the cell to zero switches off this error detection.

Cell 82 defines an absolute variation limit for the high-speed clock (HSC) calibration factor. It can be used as a quality check for the HSC oscillator ceramic, or to control if the chip is properly configured, e.g., when using an 8 MHz HSC. The limit is in raw TDC units, given as deviation from a measurement of 4 low speed clock periods (which is 4 * 30.52 µs, and as raw value at 4 MHz HSC, 0x01E84800); 0x0009C400 = 2 % permissible deviation at 4MHz HSC is a reasonable choice. Setting the cell to zero switches off this error detection.

Cells 84 and 87 define two variation limits between current results from TOF_UP and TOF_DOWN measurements, the permissible difference in measured amplitudes and in measured pulse width ratios, respectively. In combination with the SUMTOF deviation limit of cell 81, these cells are used for bubble detection. Their values depend on measurement noise, so they should be chosen based on representative measurements with the actual flow meter system. The reliability of bubble detection depends strongly on a reasonable setting of these parameters. Values from the template firmware data file are, for example, 0x00200000 for cell 84, meaning 32 mV amplitude variation, and 0x0000000C for cell 87, meaning an absolute difference of pulse width ratios of 0.094. Setting one of the cell values to zero switches off the corresponding error detection.

Cell 85 defines the low amplitude limit (also see cell 78 above). The value should be chosen such that it is always exceeded by every acceptable measurement. When a lower amplitude appears the ScioSense firmware will signal a no water/hardware error after the number of measurements defined in cell 78. Setting the cell to zero switches off this error detection.

Cell 92 defines the maximum flow for the pulse interface, given as integer in l/h. It is also used to set a limit of 2x this maximum flow for calculated flow, to keep the possible influence of undetected measurement mistakes on flow volume limited. This parameter should not be set to zero, the flow limitation can be switched off, if desired, through bit 16 of ***FWD_FW_CONFIG***.

Finally, hardware error flags as provided by register **SRR_ERR_FLAG** are also in use. Configuration register **CR_IEH** defines which of these error flags are activated. Firmware data cell 112 is copied to **CR_IEH** through the bootloader, so this cell must contain the desired configuration setting. For a proper function of the checksum error flags, four checksum values for comparison must be provided in cells 124 to 126. The simplest way to get the right values is to load the final firmware data file into the UFC Evaluation Software's download window. Checksums for firmware data will be immediately calculated there (cells 124 and 125) as well as the checksum for the firmware code user part, if it is also opened (cell 126). The checksum for the firmware code ScioSense part should be taken from ScioSense template firmware data file (get it into the download window by "FWA manual entry").

Table 7: Firmware data error handling

Cell	Variable Name	Function	Description *default value (if applicable)*
74	***FWD_SOUND_VEL_MAX***	Medium related	Maximum of speed of sound in m/s Default value for water 0x00061400 Format fd8
77	***FWD_THETA_MAX***		B3/B2/B1: Temperature at maximum speed of sound in °C, format fd8. B0: minimal speed of sound, format fd5. Default value for water 0x004A002B
78	***FWD_LONG_TERM_ERROR***		Number of low AM measurements before hardware failure / no-water checks are done Proposed value 0x00000020
81	***FWD_TOFSUM_VAR_LIM***		Error limit for deviation of SUMTOF from former average (raw TDC units)
82	***FWD_HSC_DEV***		Error limit for HSC calibration in raw TDC units (deviation time from reference measurement of 4 LSC periods); format fd16

84	*FWD_AM_DIFF_LIM*		Error limit for deviation between currently measured amplitude UP and DOWN in mV, format fd16	
85	*FWD_R_AM_MIN*		Define error flag positions that issue an interrupt	
87	*FWD_PW_DEV*		Error limit for deviation between currently measured UP and DOWN pulse width, format fd7	
92	*FWD_R_PULSE_MAX_FLOW*		Pulse interface & maxflow error limit: maximum permissible flow in l/h divided by 2.	
112	*FWD_IEH*		Configuration data for **CR_IEH**	
124	*FWD_R_FWD1_CS*	Checksums	Checksum firmware data 1	These checksums are stored for comparison to values calculated by the chip. Deviations in comparison cause checksum errors. Can be calculated by the UFC Evaluation Software.
125	*FWD_R_FWD2_CS*		Checksum firmware data 2	
126	*FWD_R_FWU_CS*		Checksum firmware code user	
127	*FWD_R_FWA_CS*		Checksum firmware code ScioSense	

The following bits in ***FWD_FW_CONFIG*** define error handling: Bit 27 enables the control of calculated speed of sound, and bit 16 limits to double the maximum flow. Bits 21 and 22 define if the pulse interface signals errors, and if no-water/hardware error should be signaled as error, too.

- BNR_FWCONF_2MAX_NOZERO
  0: Set flow to zero when exceeding 2x maximum flow and signal error
  1: Flow remains even when exceeding 2x maximum flow
- BNR_FWCONF_NO_PI_ERR
  0: Signal error on pulse interface as configured in bit 22
  1: Never signal error on pulse interface
- BNR_FWCONF_PI_ERROR
  0: Don't signal no-water/hardware error as error on pulse interface
  1: Also signal no-water/hardware error as error on pulse interface
- BNR_FWCONF_VLIM
  0: Disable control of speed of sound limits
  1: Enable control of speed of sound limits

5.8 Parameters for Error Counters & Interrupts

Details on error counters are given in section 8.3. If average error counters 1 and 2 are enabled by ***FWD_FW_CONFIG*** (see further below), they are configured by firmware data cells 3 (counter 1) and 4 (counter 2). There is no basic difference between these two counters. To define which error flags should be counted, set the corresponding bits in cell 3 or 4 according to the bits defined in ***RAM_R_FW_ERR_FLAGS*** (see section 0). The results for peak average error count/hour will appear in ***RAM_ERROR_COUNT_21*** in the upper or the lower two Byte, respectively. With average error counters enabled, another RAM cell ***RAM_ERROR_COUNT_43*** will also contain counts for any error (upper two Byte) and hardware errors (lower two Byte). Then, FWD cells 5 and 6 must be set to 0x00000000. These cells count the errors during one hour between the regular hourly recalls, which reset them to the stored 0x00000000. The two result RAM cells store the peak value reached in each of these hourly counters.

Example: Counter 1 should count bubble events and TOF timeouts, counter 2 should count no-water events. Set cell 3 to 0x00024000 and cell 4 to 0x00008000 and activate error counters. During the first hour after activation, the numbers in ***RAM_ERROR_COUNT_21*** and ***RAM_ERROR_COUNT_43*** will rise according to the actual events - each measurement cycle with the configured error(s) is one event. Then they will remain at their maximum values until, in another hour, more events appeared. In long term, these cells will contain the long-term peak value of an hourly average count. An example result may be 0x00EA0132 in cell ***RAM_ERROR_COUNT_21*** (234 times “no-water” and 306 bubble events and/or TOF timeouts) and 0x014C0121 in cell **RAM_ERROR_COUNT_43** (totally 332 events, of which 289 where hardware errors – probably TOF timeouts, so there were probably 17 bubble events in comparison).

Cell 83 defines the errors which may issue a synchronous firmware interrupt in a similar way as the error counter configurations above:
Set the desired bits according to error flag positions in ***RAM_R_FW_ERR_FLAGS*** (see 6.1.3) and switch on the synchronous firmware interrupt in **CR_IEH** and cell 112, respectively. This should be used to issue an irregular interrupt at a user-defined error condition. A typical application of such an interrupt would be if AS6031F1 communicates its results only rarely to the system's microcontroller, but special events should be recognized immediately. For example, writing 0x00008000 in cell 83 will issue an irregular interrupt on the interface as soon as no-water/hardware error is detected.

Table 8: Firmware data error counters & interrupts

Cell	Variable Name	Function	Description *default value (if applicable)*
3	***FWD_ERROR_COUNT_CONF1***	(optional) Error counters	Define error flag positions to be counted in error counter 1 (see section 8.3)
4	***FWD_ERROR_COUNT_CONF2***		Define error flag positions to be counted in error counter 2 (see section 8.3)
5	***FWD_ERROR_COUNT_21***		Temporary storage of error counters 2 (B3, B2) and 1 (B1, B0) (see section 8.3); Set to *0x00000000* if error counters are used
6	***FWD_ERROR_COUNT_43***		Temporary storage of error counter 4 - each error (B3, B2) and 3 - hardware errors (B1, B0) (see section 8.3); Set to *0x00000000* if error counters are used
83	***FWD_ERR_INTERRUPT***		Define error flag positions that issue an interrupt. Same bits as RAM_R_FW_ERR_FLAGS1

112	***FWD_IEH***		Configuration data for **CR_IEH**

Bit 28 in ***FWD_FW_CONFIG*** switches average error counters on or off:

- BNR_FWCONF_ERR
  0: Disable average error counters
  1: Enable average error counters

5.9 Parameters for FHL Regulation

First hit level (FHL) regulation is a major function of the ScioSense firmware and is described in detail in section 6. The different FHL regulation methods are set with bits 24 and 35 of ***FWD_FW_CONFIG***, see below. The following parameters are used with the different FHL regulation methods:

Cell 79 ***FWD_FHL_USER*** defines the "trusted" FHL for methods 1 to 4. The trusted FHL is a level where a well-defined first hit is determined at any operation condition. According to the chosen setting in bit 26 of ***FWD_FW_CONFIG***, this value is either interpreted as absolute voltage (only Byte 0 is read in this case), or the two lower Bytes are interpreted as a ratio of the measured amplitude. The latter can be used to compensate production tolerances and aging, but it may cause errors when the amplitude measurement is wrong (note: measurements which have recognized errors are never used to calculate an amplitude or to regulate FHL). Example: A value of 0x00001755 in cell 79 may be interpreted as 9.1 % of the amplitude (bit 26 = 1), say 36.5 mV of 400 mV receive amplitude, or as 74.8 mV (bit 26 = 0; 0x55*0.88 mV).

Cell 80 ***FWD_TOF_SUM_DELTA*** defines in method 3 the nominal SUMTOF difference between normal operation FHL and the trusted FHL value. It is typically a multiple of measurement frequency periods. Method 3 is used when the well-defined first hit, determined through operation with the trusted FHL, is not suitable for stable long-term operation. The trusted FHL is then only used to check consistency of the currently operating FHL with a nominal time distance, as given in cell 80, between the current first hit and the one defined by the trusted FHL.

The SUMTOF variation limit in cell 81 ***FWD_TOFSUM_VAR_LIM*** was explained in section 3.4 above. It has an important function in FHL regulation methods 2 and 3: Consistency of the operating FHL with the trusted FHL is checked by regularly switching to the trusted FHL. As long as the measured SUMTOF does not deviate more than the value in cell 81 from the expected value, the operating FHL is considered consistent, and operation is proceeded without changing the current FHL.

The nominal pulse width ratio (PWR) value in cell 86 ***FWD_PW_NOM*** switches on PWR regulation. It is good practice to define an optimal PWR value which provides optimal clearance between those FHL values where the detected first wave changes (this happens at peak amplitudes of the neighboring waves). PWR regulation takes place every 32 measurements, but with 16 measurements offset to FHL regulation, to minimize mutual influences. Setting cell 86 to zero switches off PWR regulation.

Cell 107 ***FWD_R1_FHL_VALUE*** is start and fallback value of the FHL, always given in absolute mV. AS6031F1 always starts with this FHL value when switched on, before starting to regulate. In case of long-term error, when no FHL regulation is done, the firmware regularly switches to this FHL value to check if a valid measurement may be possible.

Table 9: Firmware data FHL

Cell	Variable Name	Function	Description *default value (if applicable)*
79	***FWD_FHL_USER***		B1/B0: trusted FHL ratio (option B), format fd16, or B0: absolute trusted FHL, LSB = 0.88 mV, format fd0.
80	***FWD_TOF_SUM_DELTA***		FHL method 3: Nominal time difference (raw TDC units) in SUMTOF between operating and trusted FHL
81	***FWD_TOFSUM_VAR_LIM***		Error limit for deviation of SUMTOF from former average (raw TDC units)
86	***FWD_PW_NOM***		Nominal Pulse width for FHL option A, fd7
88	***FWD_CORR_OFFSET***		Offset correction in case of negative PWL coefficients; negative value in raw TDC format fd16
107	***FWD_R1_FHL_VALUE***		Start / fallback value of FHL, LSB = 0.88 mV;

Most of all, bits 24 and 25 of ***FWD_FW_CONFIG*** configure the FHL regulation method, as described in the table. Bit 26 determines if the FHL value given in FWD cell 79 is interpreted as a full amplitude ratio or an absolute voltage. With bit 23 it can be chosen if FHL regulation should happen all 32 measurements or only in error case. Regulating only in error case may increase operation stability, but it has the disadvantage that first an actual error must happen before FHL is being corrected. Finally, bit 17 configures if zero flow state is switched off while regulating FHL. This significantly speeds up retuning to the desired operation state after detecting an inconsistent FHL, but it comes at the cost of increased power consumption (which should be negligible, since in stable operation FHL should always be consistent with the desired operation).

- BNR_FWCONF_FHL_RATIO: Configuration for FHL regulation option B
  0: Interpret FHL-values as fixed voltage
  1: Interpret FHL-values as ratio to measured amplitude
- BNR_FWCONF_FHL: Configuration of FHL regulation methods
  00: Method 1, fixed FHL
  01: Method 2, consistency check against trusted FHL
  10: Method 3, consistency check against trusted FHL with offset time
  11: Method 4, consistency check against special configuration
- BNR_FWCONF_TESTMODE: Configuration for FHL regulation option C
  0: Enter FHL test mode only in error case
  1: Enter FHL test mode regularly (each 32 measurements)

- BNR_FWCONF_FHL_ZEROFLOW
  0: With FHL regulation active, disable zero flow state (always assume full flow as long as FHL is considered not ok)
  1: Apply zero flow state independently of FHL regulation

5.10 Parameters for Multihit Release Delay

FWD_MH_RLS_DLY holds the nominal value for release delay with 1 LSB = 7.8125 ns. This value is copied into the system handling registers **SHR_USM_RLS_DLY_U** and **SHR_USM_RLS_DLY_D** during initialization.

Bit 19th, USM_RLY_MODE, of the CR_USM_PRC register defines whether the release delay window will be regulated. If 0, it means that only FHL for derivation of the start release delay is used. If 1, it means that the release delay window or a combination with FHL is used.

The multihit release delay correction sets the release delay window for UP and DOWN with of $T_{ref}/4$ in advance to the first hit. If the first hit level FHL ≠ 0, it increases this distance to 2.25 periods (filter mode). The updated new release delays are compared with the limits set in FWD_MH_RLS_DLY_LIM. The format of this is shifted by 3 bits so that the two limits fit into the data word.

Table 10: Firmware data phase shift detection

Cell	Variable Name	Function	Description *default value (if applicable)*
103	***FWD_MH_RLS_DLY***		Nominal value for release delay, 1 LSB = 7.8125 ns; Copied into SHR_USM_RLS_DLY_U and … _D during initialization
105	***FWD_MH_RLS_DLY_LIM***		Multihit release delay limits (min B1, B0. & max B3, B2.); average is initial value; 1 LSB = 62.5 ns
116	***FWD_USM_PRC***		Bit 19 defines whether RLS_DLY correction is active.

5.11 Parameters for Phase Shift Detection

A few parameters need to be set for the correct detection of the right position of the phase shift and consequently the correction of the ToF.

Table 11: Firmware data phase shift detection

Cell	Variable Name	Function	Description *default value (if applicable)*

104	*FWD_PH_S_THRESHOLD*		Absolute phase-shift threshold value. Cannot be a negative number. fd16 * T_{ref}(ns)

5.12 Parameters for Temperature Measurement

Temperature measurement with ScioSense firmware is either with one or two PT1000 or PT500 sensors, or the measurement using the internal sensor. Other types of external sensors require different reference curves and may be used with modified firmware.

For external PT sensors, only three parameters are needed:

- Cell 97 contains the reference resistor, for example 1 kΩ = 0x03E80000 for PT1000
- Cells 104 and 105 contain the 0 °C resistance ratios to reference for cold and hot sensor, respectively, these values are normally 1 = 0x0001000.

The internal sensor is not very accurate and may be operated with the nominal values given in the table below. For higher accuracy, the values for nominal resistance and resistance temperature slope in cells 103 and 100, respectively, should be calibrated (simplest calibration would be changing cell 103 to get the known temperature, changing cell 100 and even the offset value cell 96 would require the knowledge of measurements at one more temperature).

Table 12: Firmware data temperature

Cell	Variable Name	Function	Description *default value (if applicable)*
96	*FWD_CAL_PTR_OFFSETR*	Temperature measurement	Reference branch offset resistance in internal reference in Ω, format fd16; typical value 0x00000000 (calibrate if desired)
97	*FWD_EXT_REF_VAL*		Value of reference resistor R_{ref} in Ω, format fd16; typical value for PT1000: 1 kΩ =0x03E80000
98	*FWD_PT_INT_SLOPE*		Internal sensor resistance slope in (K/Ω)*Rref; format fd13; typical value 0x0029F000 (calibrate if desired)
99	*FWD_PT_INT_NOM*		Internal sensor nominal resistance in Ω, format fd16; typical value 0x03C20000 (calibrate if desired)
100	*FWD_PTC_RATIO_INV*		Nominal ratio of reference resistor to PT cold sensor resistance at 0 °C; format fd16 typical value 1 = 0x00010000

101	*FWD_PTH_RATIO_INV*		Nominal ratio of reference resistor to PT hot sensor resistance at 0 °C; format fd16; typical value 1 = 0x00010000

Bit 30 in FWD_FW_CONFIG determines if the sensor temperature measurement should be used to select calibration coefficients:

- BNR_FWCONF_TSENS
  0: Use temperature value from flow meas. for calibration coefficients
  1: Use temperature value from hot sensor measurement for calibration coefficients

5.13 Parameters for Pulse Interface

The outputs used for pulse interface are defined by hardware configuration in configuration register **CR_GP_CTRL**[5]. It is enabled and configured in **CR_PI_E2P**. However, most configuration settings in this register are being overwritten by the ScioSense firmware to simplify pulse interface configuration. FWD cells 91 and 92 define the number of pulses per liter and the permissible maximum flow, and the pulse interface settings are chosen such that a maximum pulse width is generated while not producing double pulses. The number of pulses per liter should be chosen such that a minimum pulse period of 10 ms is not undercut and that it does not exceed 100 pulses per second at maximum flow.

Cell 92 holds the maximum flow divided by 2. It is also used to limit flow values in error case (measured flow never higher than double maximum flow, see section 5.4). E.g., if the content is 3000 l/h, then any flow above 6000 l/h set the error flag.

ScioSense firmware can be configured to signal error conditions over the pulse interface, too (see section 8.5). In case of error, the pulse output goes permanently high, and the direction output toggles with the TOF measurement frequency. This creates no additional pulse count and can be easily identified by some external readout device or master controller. With bit 22 of the firmware configuration register ***FWD_FW_CONFIG*** it can in addition be chosen if a no-water/hardware error situation should also be signaled as error or not.

The load resistance applied to the pulse interface can have a significant influence on current consumption, since in case of high flow, pulses would permanently drive current through this load. With a load impedance of 1 MΩ, the maximum additional current consumption is in the range of 1.5 µA.

Table 13: Firmware data pulse interface

Cell	Variable Name	Function	Description *default value (if applicable)*

[5] See datasheet SC-000853-DS

91	*FWD_R_PULSE_PER_LITER*	Initial values	Pulse interface: Number of pulses per liter
92	*FWD_R_PULSE_MAX_FLOW*		Pulse interface & maxflow error limit: maximum permissible flow in l/h divided by 2.
109	*FWD_PI_E2P*		Configuration data for **CR_PI_E2P**
110	*FWD_GP_CTRL*		Configuration data for **CR_GP_CTRL**

- Bit 21, BNR_FWCONF_NO_PI_ERR, determines if error should be signaled over the pulse interface at all.
- Bit 22, BNR_FWCONF_PI_ERROR, configures if no-water/hardware error should be signaled as an error over the pulse interface. AB error is signaled over the pulse interface by setting the pulse port to “permanently high” while the direction port toggles.

5.14 Optional Parameters

Several FWD cells are unused or optionally unused (if some function is disabled). They may be used for any custom purpose. Parameters used in sensor temperature measurement (see section 5.10) are generally available for other purposes if this function is not activated.

3	***FWD_ERROR_COUNT_CONF1***
4	***FWD_ERROR_COUNT_CONF2***
5	***FWD_ERROR_COUNT_21***
6	***FWD_ERROR_COUNT_43***
7	***FWD_ERROR_COUNT_INV21***
8	***FWD_ERROR_COUNT_INV43***
16 to 41	PWL coefficient table
42 to 53	ScioSense coefficient table
96	***FWD_CAL_PTR_OFFSETR***
97	***FWD_EXT_REF_VAL***
100	***FWD_PT_INT_SLOPE***
103	***FWD_PT_INT_NOM***
104	***FWD_PTC_RATIO_INV***
105	***FWD_PTH_RATIO_INV***

6 Firmware Output Data (RAM)

6.1 Variables and Results in RAM

All results as well as all messages (operation mode messages or error flags) are stored in specific RAM cells.

The AS6031F1 has a total of 176 32-bit RAM memory cells[6]. This volatile memory block is used by firmware as well as by the frontend data buffer. In addition, any memory cell of FWD (address 0x100 to 0x17F) can be used as volatile RAM cell, too – keeping in mind that these cells are overwritten at recalls by their corresponding flash memory content. The following section lists the complete RAM usage of the ScioSense firmware as well as the bit definitions in ***RAM_R_FW_STATUS*** and ***RAM_R_FW_ERR_FLAGS***. The subsequent section summarizes the cells of FWD which are used as RAM by the ScioSense firmware.

The following list is a complete overview of RAM memory cell usage by the ScioSense firmware.

All variable names and address definitions can also be found in the header file.

Table 14: RAM content[7]

Addr.		Variable name	Description	Format
0x000	Final results from flow calculation	***RAM_R_FLOW_VOLUME_INT***	Signed integer part of total volume of water flow in cubic meters	fd0
0x001		***RAM_R_FLOW_VOLUME _FRACTION***	Unsigned factional part of total volume of water flow in cubic meters	fd32
0x002		***RAM_R_FLOW_LPH***	Presently calculated flow volume (l/h), unfiltered	fd16
0x003		***RAM_FILTERED_FLOW_LPH***	Filtered flow volume (l/h), 2's complement for negative value	fd16
0x004		***RAM_R_THETA***	Temperature (°C) calculated from SUMTOF	fd16
0x005		***RAM_SOUND_VEL***	Velocity of sound (m/s)	fd8
0x006		***RAM_FLOW_SPEED***	Calculated speed of flow (m/s)	fd16
0x007		***RAM_R_TOF_DIFF***	Current DIFTOF in raw TDC units	fd16
0x008		***RAM_R_TOF_SUM***	Current SUMTOF in raw TDC units	fd16
0x009 to 0x01E		Permanent variables and parameters for calculations	These variables are used to transfer former results and settings to the current flow calculation. Do not overwrite.	-

[6] See datasheet SC-000853-DS
[7] 2's complement for negative value

Addr.		Variable name	Description	For-mat
0x01C		*RAM_R_AM_MIN_RAW*	Minimal acceptable signal amplitude, calculated from latest calibration values for direct comparison to raw amplitude measurement value.	-
0x01D		*RAM_R_AMC_GRADIENT*	Latest amplitude calibration gradient value	-
0x01E		*RAM_R_AMC_OFFSET*	Latest amplitude calibration offset value	-
0x01F		*RAM_R_V1F_COEFF_ADR*	Temporary variable for PWL coefficients table address	-
0x020		*RAM_R_PTC_TEMPERATURE*	Cold sensor temperature (°C)	fd16
0x021		*RAM_R_PTH_TEMPERATURE*	Hot sensor temperature (°C)	fd16
0x022		*RAM_PTC_RES*	Cold sensor resistance (Ω)	fd16
0x023		*RAM_R_PTH_RES*	Hot sensor resistance (Ω)	fd16
0x024		*RAM_R_PT_INT _TEMPERATURE*	Internal sensor temperature (°C)	fd16
0x025	Status information	*RAM_R_FW_STATUS*	Firmware status bits, see 6.1.1	bits
0x026		*RAM_FLOW_COUNTER*	Internal counter for flow values after last average update	int
0x027		*RAM_R_FW_ERR_FLAGS*	Firmware and hardware error flags, see 0 Note that most of these flags are not permanent and erased with each new measurement (flow or sensor temperature).	bits
0x028		*RAM_R_FHL_ERR_CTR*	Internal error counter, increased with every error which prevents flow calculation and reset to zero at every correct measurement. At values below 4, corrupted measurements are replaced by valid preceding results. At values of 4 and above, measurement failure is signalized, flow is set to zero and bubble detection is checked.	int
0x029 0x02A		Temporary parameters		
0x02B		*RAM_V2B_PH_S_UP_0*	Phase-shift time 0 UP in multiples of T_{ref}	fd16
0x02C		*RAM_V2C_PH_S_UP_1*	Phase-shift time 1 UP in multiples of T_{ref}	fd16
0x02D		*RAM_V2D_PH_S_UP_2*	Phase-shift time 2 UP in multiples of T_{ref}	fd16
0x02E		*RAM_V2E_PH_S_UP_3*	Multiple temporary use by UFC_PHASE_SHIFT.lib	fd16

Addr.		Variable name	Description		For-mat
0x02F		***RAM_V2F_PH_S_DOWN_0***	Multiple temporary use by UFC_PHASE_SHIFT.lib		fd16
0x030		***RAM_V30_PH_S_DOWN_1***	Multiple temporary use by UFC_PHASE_SHIFT.lib		fd16
0x031		***RAM_V31_PH_S_DOWN_2***	Multiple temporary use by UFC_PHASE_SHIFT.lib		fd16
0x032		***RAM_V32_PH_S_DOWN_3***	Multiple temporary use by UFC_PHASE_SHIFT.lib		fd16
0x033		***RAM_V33_PH_S_ERR_FLAGS***	Error Flags for Phase-Shift		
0x034		***RAM_PERIOD_UPD***	Period time in UP and DOWN, HSC calibrated value		-
0x035		***RAM_AM_AV_UP***	Output of amplitude IIR filter, copied also to FDB		
0x036		***RAM_AM_AV_DOWN***	Output of amplitude IIR filter, copied also to FDB		
0x037 to 0x04C		NOT USED			
0x04D	Error counters	***RAM_ERROR_COUNT_21***	(optional) Peak hourly error count of counters 2 (B3/B2) and 1 (B1/B0), as configured in 0x103 and 0x104 Free in case error counters are not activated		int
0x04E		***RAM_ERROR_COUNT_43***	(optional) Peak hourly error count of counters 4 (B3/B2; all errors) and 3 (B1/B0; hardware defined errors) Free in case error counters are not activated		int
0x04F	Negative flow	***RAM_NEG_FLOW***	Presently calculated negative flow volume (l/h), unfiltered[7]	(optional) Flow values for negative flow exceeding the limit given in FWD cell 93 Flow which is not considered in flow volume. Unused if no limit is set	fd16
0x050		***RAM_NEG_FLOW_VOLUME _INT***	Integer part of total volume of negative water flow in cubic meters[7]		fd0
0x051		***RAM_NEG_FLOW_VOLUME _FRACTION***	Fractional part of total volume of negative water flow in cubic meters		fd32
0x052	Redundant storage for flow	***RAM_C1_FLOW_VOLUME _INT***	Safety copy 1 of flow volume, integer part	(optional) These four cells contain, if configured, 2x2 safety copies of ***RAM_R_FLOW_VOLUME_INT*** and ***RAM_R_FLOW_VOLUME_FRACTION***.	fd0
0x053		***RAM_C2_FLOW_VOLUME _INT***	Safety copy 2 of flow volume, integer part		fd0
0x054		***RAM_C1_FLOW_VOLUME _FRACTION***	Safety copy 1 of flow volume, fractional part		fd32

Addr.		Variable name	Description		For-mat
0x055		***RAM_C2_FLOW_VOLUME_FRACTION***	Safety copy 2 of flow volume, fractional part	Volume values are considered valid when at least two of these three values are identical, else the error flag ***BNR_VOL_ERR*** is raised.	fd32
0x056		***RAM_FEP_STF***	Copy of recent **SRR_FEP_STF**, to keep front end status after clear of **SRR_FEP_STF**		bits
0x057	Error- and cycle counters	***RAM_LOW_AM_ERR_CTR***	Counter for consecutive low AM cases; this counter is compared to ***FWD_LONG_TERM_ERROR*** to do a no-water / hardware check	These error counters count the number of measurements with error, they are cleared at every valid measurement	int
0x058		***RAM_TS_ERR_CTR***	Counter for consecutive Task sequencer timeout errors		int
0x059		***RAM_R_TM_ERR_CTR***	Counter for consecutive TM errors		int
0x05A		***RAM_R_USM_ERR_CTR***	Counter for consecutive USM errors		int
0x05B		***RAM_R_AM_ERR_CTR***	Counter for consecutive AM errors		int
0x05C		***RAM_CYCLE_COUNTER***	This counter controls the slow FHL changes Bits [4:0]: cycle counter 1-16; bit [5] indicates if PWR or FHL regulation is due at value 16		int
0x05D	Permanent Clock related	***RAM_R_HSC_SCALE_FACT***	HS Clock scaling factor, deviation from nominal frequency		fd24
0x05E		***RAM_R_TDC_T_BY_2L***	this variable stores TDC_PERIOD/(2*DIST_WITH_FLOW) fd0 including HSC calibration		fd39
0x05F		***RAM_R_TDC_CLK***	real HSC frequency: (4 MHz: 0x003D0900 or 8 MHz: 0x007A1200) / ***RAM_R_HSC_SCALE_FACT***		fd0
0x060 to 0x06F		***RAM_R_ROLAVG_1 ...*** ***RAM_R_ROLAVG_16***	Rolling average filter for flow, starting at 0x060: ***RAM_R_ROLAVG_1***	These variables are used for filtering over	

Addr.		Variable name	Description	For-mat
0x070 to 0x077		***RAM_ROLAVG_DIFTOF_1 ... RAM_ROLAVG_DIFTOF_8***	Rolling average filter for DIFTOF, starting at 0x070: ***RAM_ROLAVG_DIFTOF_1*** — some measurement cycles. Don't modify !	
0x078 to 0x07F		***RAM_ROLAVG_SUMTOF_1 ... RAM_ROLAVG_SUMTOF_8***	Rolling average filter for SUMTOF, starting at 0x078: ***RAM_ROLAVG_SUMTOF_1***	
0x080	Frontend data buffer	***FDB_US_TOF_ADD_ALL_U / FDB_TM_PP_M1***	TOF Sum Up of all the configured hits / t_{pp}: Offset delay comp. value of Meas. 1	These variables contain all measurement results of the measurement frontend. The cells may be overwritten and reused after valuation. ScioSense firmware usually doesn' t modify these cells. This table just lists the variable names for results for TOF and TM measurements, for details please see Manual Vol. 1, 3.2 and 3.3
0x081		***FDB_US_PW_U / FDB_TM_PTR_RAB_M1***	US Pulse Width Ratio Up / t_{RAB}: Reference Impedance value of Meas. 1	
0x082		***FDB_US_AM_U / FDB_TM_PTC_CAB_M1***	US Amplitude Value Up / t_{CAB}: PT Cold Impedance value of Meas. 1	
0x083		***FDB_US_AMC_VH / FDB_TM_PTH_HAB_M1***	US Amplitude Calibrate Value High / t_{HAB}: PT Hot Impedance value of Meas. 1	
0x084		***FDB_US_TOF_ADD_ALL_D / FDB_TM_PTR_RA_M1***	TOF Sum Down of all the configured hits / t_{RA}: 1st Offset resistance value of Meas. 1	
0x085		***FDB_US_PW_D / FDB_TM_PP_M2***	US Pulse Width Ratio Down / t_{pp}: Offset delay comp. value of Meas. 2	
0x086		***FDB_US_AM_U / FDB_TM_PTR_RAB_M2***	US Amplitude Value Down / t_{RAB}: Reference Impedance value of Meas. 2	
0x087		***FDB_US_AMC_VL / FDB_TM_PTC_CAB_M2***	US Amplitude Calibrate Value Low / t_{CAB}: PT Cold Impedance value of Meas. 2	
0x088		***FDB_US_TOF_0_U / FDB_TM_PTH_HAB_M2***	Ultrasonic TOF Up Value 0 / t_{HAB}: PT Hot Impedance value of Meas. 2	
0x089		***FDB_US_TOF_1_U / FDB_TM_PTR_RA_M2***	Ultrasonic TOF Up Value 1 / t_{RA}: 1st Offset resistance value of Meas. 2	
0x08A		***FDB_US_TOF_2_U / FDB_TM_PTR_4W_RB_M1***	Ultrasonic TOF Up Value 2 / t_{RB}: Reference 2nd Offset res. val. of Meas. 1	
0x08B		***FDB_US_TOF_3_U / FDB_TM_PTC_4W_CA_M1***	Ultrasonic TOF Up Value 3 / t_{CA}: PT Cold 1st Offset res. value of Meas. 1	
0x08C		***FDB_US_TOF_4_U / FDB_TM_PTC_4W_CB_M1***	Ultrasonic TOF Up Value 4 / t_{CB}: PT Cold 2nd Offset res. value of Meas. 1	

Addr.		Variable name	Description	Format
0x08D		*FDB_US_TOF_5_U / FDB_TM_PTC_4W_AC_M1*	Ultrasonic TOF Up Value 5 / t_{AC}: PT Cold 3^{rd} Offset res. value of Meas. 1	
0x08E		*FDB_US_TOF_6_U / FDB_TM_PTC_4W_BC_M1*	Ultrasonic TOF Up Value 6 / t_{BC}: PT Cold 4^{th} Offset res. value of Meas. 1	
0x08F		*FDB_US_TOF_7_U / FDB_TM_PTH_4W_HA_M1*	Ultrasonic TOF Up Value 7 / t_{HA}: PT Hot 1^{st} Offset res. value of Meas. 1	
0x090		*FDB_US_TOF_0_D / FDB_TM_PTH_4W_HB_M1*	Ultrasonic TOF Down Value 0 / t_{HB}: PT Hot 2^{nd} Offset res. value of Meas. 1	
0x091		*FDB_US_TOF_1_D / FDB_TM_PTH_4W_AH_M1*	Ultrasonic TOF Down Value 1 / t_{AH}: PT Hot 3^{rd} Offset res. value of Meas. 1	
0x092		*FDB_US_TOF_2_D / FDB_TM_PTH_4W_BH_M1*	Ultrasonic TOF Down Value 2 / t_{BH}: PT Hot 4^{th} Offset res. value of Meas. 1	
0x093		*FDB_US_TOF_3_D / FDB_TM_PTR_4W_RB_M2*	Ultrasonic TOF Down Value 3 / t_{RB}: Reference 2^{nd} Offset res. val. of Meas. 2	
0x094		*FDB_US_TOF_4_D / FDB_TM_PTC_4W_CA_M2*	Ultrasonic TOF Down Value 4 / t_{CA}: PT Cold 1^{st} Offset res. value of Meas. 2	
0x095		*FDB_US_TOF_5_D / FDB_TM_PTC_4W_CB_M2*	Ultrasonic TOF Down Value 5 / t_{CB}: PT Cold 2^{nd} Offset res. value of Meas. 2	
0x096		*FDB_US_TOF_6_D / FDB_TM_PTC_4W_AC_M2*	Ultrasonic TOF Down Value 6 / t_{AC}: PT Cold 3^{rd} Offset res. value of Meas. 2	
0x097		*FDB_US_TOF_7_D / FDB_TM_PTC_4W_BC_M2*	Ultrasonic TOF Down Value 7 / t_{BC}: PT Cold 4^{th} Offset res. value of Meas. 2	
0x098		*FDB_TM_PTH_4W_HA_M2*	t_{HA}: PT Hot 1^{st} Offset res. value of Meas. 2	
0x099		*FDB_TM_PTH_4W_HB_M2*	t_{HB}: PT Hot 2^{nd} Offset res. value of Meas. 2	
0x09A		*FDB_TM_PTH_4W_AH_M2*	t_{AH}: PT Hot 3^{rd} Offset res. value of Meas. 2	
0x09B		*FDB_TM_PTH_4W_BH_M2*	t_{BH}: PT Hot 4^{th} Offset res. value of Meas. 2	
0x09C to 0x0AF		Temporary parameters	These variables are used by firmware or ROM for temporary results. They can be used temporarily for custom codes, too.	
0x0A8	Checksum	*RAM_R_VA8_FWD1_CS*	Checksum Firmware Data 1 / Note that these checksums are	fd0

Addr.		Variable name	Description		For-mat
0x0A9		*RAM_R_VA9_FWD2_CS*	Checksum Firmware Data 2	calculated by the chip and only available after the calculation process has run (part of the bootloading process). Values here may be overwritten by firmware later	fd0
0x0AA		*RAM_R_VAA_FWU_CS*	Checksum Firmware Code User		fd0
0x0AB		*RAM_R_VAB_FWA_CS*	Checksum Firmware Code ScioSense		fd0

6.1.1 Firmware Status Register (RAM_R_FW_STATUS)

All bits of this register besides the first nine are controlled by the firmware. They are listed here for information and should normally not be changed from outside. Bits 0, 1 and 3 to 8 are copied from frontend status SRR_FEP_STF.

Table 15: RAM_R_FW_STATUS Register

Addr: 0x025		RAM_R_FW_STATUS		
Bit	**Bit Name**	**Default**	**Format**	**Bit Description**
0	**BNR_HCC_UPD**	0b0	BIT	Copy of current state in **SRR_FEP_STF** 0: No request 1: High speed clock calibration updated
1	**BNR_TM_UPD**	0b0	BIT	Copy of current state in **SRR_FEP_STF** 0: No request 1: Temperature measurement updated
2	NOT USED			
3	**BNR_FLOW_FILT_INIT_DONE**	0b0	BIT	0: Flow filter not initialized 1: Flow filter initialized
4	**BNR_US_U_UPD**	0b0	BIT	Copy of current state in **SRR_FEP_STF** 0: No request 1: TOF_UP measurements available
5	**BNR_US_D_UPD**	0b0	BIT	Copy of current state in **SRR_FEP_STF** 0: No request 1: TOF_DOWN measurements available
6	**BNR_US_TOF_UPD**	0b0	BIT	Copy of current state in **SRR_FEP_STF** 0: No request 1: TOF measurement updated

7	**BNR_TOF_EDGE**	0b0	BIT	Copy of current state in **SRR_FEP_STF** 0: Positive TOF edge 1: Negative TOF edge
8	**BNR_AM_UPD**	0b0	BIT	Copy of current state in **SRR_FEP_STF** 0: No request 1: Amplitude measurement updated
9	**BNR_AMC_UPD**	0b0	BIT	Copy of current state in **SRR_FEP_STF** 0: No request 1: Amplitude calibration updated
10	**BNR_MEAS_FAILURE_ALERT**	0b0	BIT	0: No measurement failure signalized 1: Measurement failure: More than 4 measurement errors in sequence
11	**BNR_FILTER_INIT_DONE**	0b0	BIT	0: TOF filters not initialized 1: TOF filters initialized
12	**BNR_CHKSUM_DUE_AUX**	0b0	BIT	This bit supports checksum building indication
13	**BNR_CHKSUM_DUE**	0b0	BIT	This bit indicates that checksum building is due after the next identified recall
14	**BNR_AVG_TOF_FOR_FLOW**	0b0	BIT	0: Accumulating flow values for long term average 1: Full number of flow values for long term average reached
15	**BNR_TOFSUM_DIV_MODE**	0b1	BIT	0: Division by shifting 1: Normal division
16	**BNR_HSC_SCALE_EN**	0b1	BIT	Set HSC cal. according to configuration 0: Disable high speed clock calibration in firmware 1: Enable high speed clock calibration in firmware
17	**BNR_PI_UPD_REQ**	0b0	BIT	0: No pulse interface (PI) update 1: PI update is requested – the PI gets new flow volume added
18	**BNR_NON_ZERO_FLOW**	0b0	BIT	0: Zero flow possible 1: No zero flow - instantaneous flow is above 8*zero flow limit, or zero flow state is disabled due to regulations
19	**BNR_THETA_OUT_OF_RANGE**	0b0	BIT	0: Normal operation 1: Temperature for calibration is outside calibration table
20	**BNR_FWI_DONE**	0b1	BIT	0: No firmware initialization 1: Firmware initialization done
21	**BNR_I2C_ABORT**	0b0	BIT	(set by ROM routine) 0: No problems with I2C transactions

				1: Some I2C transaction was aborted with NOT ACKNOWLEDGE
22	**BNR_TEMP_REFRSH**	0b0	BIT	0: Normal operation 1: Refresh of registers after recall needed
23	**BNR_PI_ERR_1ST_DIR**	0b0	BIT	Temporary storage of pulse interface direction when starting error signal
24	**BNR_TEST_MODE_ADJ**	0b0	BIT	Indicate active PWR adjustment 0: Normal operation 1: FHL adjustment for nominal PWR is active
25	**BNR_TEST_MODE_EP**	0b0	BIT	Indicate no-water / hardware error test mode 0: Normal operation 1: Test mode to check for no-water / hardware error
26	**BNR_EN_FHL_MONITORING**	0b1	BIT	0: Disable FHL regulation (for temporary tests) 1: Enable FHL regulation (enforced at init or recall)
27	**NOT_USED**	0b0	BIT	Not used, 1 after init or recall
28	**NOT_USED**	0b0	BIT	Not used
29	**BNR_TOF_RATE_REDUCED**	0b0	BIT	0: TOF rate as originally configured in ***FWD_USM_PRC*** /B3 1: TOF rate changed for zero flow as given in ***FWD_TOF_RATE_FACTOR***
30	**BNR_TOF_DIFF_NEGATIVE**	0b0	BIT	0: Current DIFTOF is positive after zero-flow correction 1: Current DIFTOF is negative after zero-flow correction
31	**BNR_TEST_MODE_FHL**	0b0	BIT	Indicate active FHL regulation 0: Normal operation 1: FHL regulation is active to control first hit level or recover from error.

6.1.2 Phase Shift Status Flag Register (RAM_V33_PH_S_STATUS_FLAGS)

The phase shift detection and correction use several status flags that indicate the outcome of the detection routine.

*Table 16: **RAM_R_V33_PH_S_STATUS_FLAGS1** Register*

Adder: 0x027		RAM_R_FW_ERR_FLAGS		
Bit	**Bit Name**	**Default**	**Format**	**Bit Description**
0	**BNR_PH_S_U_JUMP_PH_LEFT**	b0	BIT	Jump in the left direction for the UP hits
1	**BNR_PH_S_U_JUMP_PH_RIGHT**	b0	BIT	Jump in the right direction for the UP hits
2	**BNR_PH_S_U_JUMP_ONE_PH**	b0	BIT	Jump of one phase for the UP hits
3	**BNR_PH_S_U_JUMP_TWO_PH**	b0	BIT	Jump of two phases for the UP hits
4	**BNR_PH_S_D_JUMP_PH_LEFT**	b0	BIT	Jump in the left direction for the DOWN hits
5	**BNR_PH_S_D_JUMP_PH_RIGHT**	b0	BIT	Jump in the right direction for the DOWN hits
6	**BNR_PH_S_D_JUMP_ONE_PH**	b0	BIT	Jump of one phase for the DOWN hits
7	**BNR_PH_S_D_JUMP_TWO_PH**	b0	BIT	Jump of two phases for the DOWN hits
8 to 13	NOT USED	b0	BIT	
14	**BNR_PH_S_U_TOF01_CONS**	b0	BIT	UP_TOF 0&1 inconsistency flag
15	**BNR_PH_S_D_TOF01_CONS**	b0	BIT	DOWN_TOF 0&1 inconsistency flag
16	**BNR_PH_S_U_TOF03_CONS**	b0	BIT	UP_TOF 0&3 inconsistency flag
17	**BNR_PH_S_D_TOF03_CONS**	b0	BIT	DOWN_TOF 0&3 inconsistency flag
18 to 21	NOT USED	b0	BIT	
22	**BNR_PH_S_CONS_ERR**	b0	BIT	Flag indicating possible inconsistency between hits TOF_0 and TOF_1 or TOF_0 and TOF_3.
23	**BNR_PH_S_DLY_LIM_MIN_EXC**	b0	BIT	Flag indicating that the release delay window MIN limit is exceeded
24	**BNR_PH_S_DLY_LIM_MAX_EXC**	b0	BIT	Flag indicating that the release delay window MAX limit is exceeded

25	**BNR_PH_S_CONF_WRG**	b0	BIT	Flag indicating wrong configuration; stops phase-shift routines execution if set
26	**BNR_PH_S_VALID_WRG**	b0	BIT	Flag indicating a validation warning
27	**BNR_PH_S_JUMP_DET_ERR**	b0	BIT	Flag indicating jump detection error (eg. both left and right jumps detected)
28 to 29	NOT USED	b0	BIT	
30	**BNR_PH_S_DONE**	b0	BIT	Flag indicating that the phase-shift routines were finished
31	**BNR_PH_S_PARAMS_INITIALIZED**	b0	BIT	Flag indicating that the parameters were initialized (only used in case of NO flow firmware inside the chip)

6.1.3 Firmware Error Flag Register (RAM_R_FW_ERR_FLAGS1)

Most bits of this register are temporary, unless otherwise noted: They indicate an error at the time it appears and clear when it vanishes. Bits [31:16] are copies of the hardware error flags in **SRR_ERR_FLAG**. While the original bits [15:0] in **SRR_ERR_FLAG** are cleared after firmware evaluation, to be able to recognize new errors, their copies in bits [31:16] of **RAM_R_FW_ERR_FLAGS** are kept until the next measurement cycle and can thus be read out.

*Table 17: **RAM_R_FW_ERR_FLAGS1** Register*

Adder: 0x027		RAM_R_FW_ERR_FLAGS1		
Bit	**Bit Name**	**Default**	**Format**	**Bit Description**
0	**BNR_HS_CALIB_FAIL**	b0	BIT	0: HS Clock deviation within limit, 1: HS Clock Calibration not done because the deviation is too large
1	**BNR_AMP_DIFF_TOO_HIGH**	b0	BIT	Check up / down amplitude difference 0: Amplitude difference of current meas. is within limits, 1: Amplitude difference of current measurement exceeds limits
2	**BNR_AMP_VAL_TOO_LOW**	b0	BIT	0: Measured amplitude up or down is above minimum 1: Measured amplitude up or down is below minimum
3	**BNR_PW_DIFF_NOT_OK**	b0	BIT	Check up and down PWR difference 0: PWR difference of current measurement is within limits

				1: PWR difference of current measurement exceeds limits
4	BNR_SUMTOF_DEV	b0	BIT	Check deviation of new SUMTOF from average 0: Deviation from former SUMTOF average is below limits 1: Deviation from former SUMTOF average exceeds limits
5	BNR_FHL_NOT_OK	b0	BIT	0: Current FHL is consistent with configuration 1: Current FHL is inconsistent with config., FHL regulation is active
6	BNR_MEAS_NOT_OK	b0	BIT	0: Measurement is o.k. (after short-term correction, if necessary) 1: Measurement is considered wrong (> 4 errors sequentially); Flow is forced to zero until the next error-free measurement occurs
7	RESERVED	b0	BIT	RESERVED
8	BNR_FLOW_BT_2MAX	b0	BIT	0: Current flow is not exceeding 2*maximum flow, see section 5.7 1: Current flow exceeds 2*maximum flow and is forced to zero
9	BNR_FLOW_LT_NEGLIM	b0	BIT	0: Flow is not below negative limit 1: Flow is below negative limit and stored separately, see section 5.6
10	BNR_VOL_ERR	b0	BIT	0: No error on stored flow volume (or no volume control) 1: Stored flow volume had an unrecoverable error; this bit is not cleared automatically, it needs to be cleared remotely
11	BNR_PH_S_FW_VALID_WRG	b0	BIT	Validation warning from the phase-shift routines
12	BNR_PH_S_FW_JUMP_DET_ERR	b0	BIT	Jump detection error, detected conflicting jump reports
13	BNR_VEL_ERROR	b0	BIT	0: Calculated velocity of sound is within limits 1: Calculated velocity of sound is outside limits; former value is used
14	BNR_BUBBLE	b0	BIT	0: Normal operation 1: Bubble detected: bits 1,3 or 4 are set after at least 4 meas. errors
15	BNR_NO_WATER	b0	BIT	(see section 6.1 and bit 2 below) 0: No no-water/hardware error situation detected

				1: No-water/hardware error detected after long-term **AMP_VAL_TOO_LOW**
16	**BNR_TDC_TMO_FW** [8]	b0	BIT	Copy of bit 0 of **SRR_ERR_FLAG** before clear Error flag TDC timeout: Flag is set when TDC timeout was reached while waiting for a signal (after 4.096 ms)
17	**BNR_TOF_TMO_FW** [8]	b0	BIT	Copy of bit 1 of **SRR_ERR_FLAG** before clear Error flag TOF timeout: Flag is set when configured TOF timeout was reached while waiting for a signal, see **CR_USM_PRC**
18	**BNR_AM_TMO_FW** [8]	b0	BIT	Copy of bit 2 of **SRR_ERR_FLAG** before clear Error flag amplitude measurement timeout: Flag is set when TDC timeout was reached during amplitude measurement, amplitude value is invalid
19	**BNR_TM_OC_ERR_FW** [8]	b0	BIT	Copy of bit 3 of **SRR_ERR_FLAG** before clear Error Flag Temperature measurement open circuit: Flag is set when the threshold voltage for temperature measurement is not reached within the configured discharge time, which indicates too high resistance / open circuit at one sensor.
20	**BNR_TM_SC_ERR_FW** [8]	b0	BIT	Copy of bit 4 of **SRR_ERR_FLAG** before clear Error Flag Temperature Measurement Short Circuit: Flag is set when the threshold voltage for temperature measurement is reached in less than 1µs, which indicates too low resistance / short circuit at one sensor.
21	**BNR_ZCC_ERR** [8]	b0	BIT	Copy of bit 5 of **SRR_ERR_FLAG** before clear Error Flag Zero Cross Calibration: Flag is set when zero cross calibration did not converge and may have failed. See datasheet for details
22	**BNR_LBD_ERR** [8]	b0	BIT	Copy of bit 6 of **SRR_ERR_FLAG** before clear Error Flag Low Battery Detect: Flag is set when the supply voltage Vcc drops below the limit (see **CR_CPM**). firmware signals this, in addition, as LOW = 0 V on GPIO6 (normally HIGH = Vcc).
23	**BNR_USM_SQC_TMO** [8]	b0	BIT	Copy of bit 7 of **SRR_ERR_FLAG** before clear Error Flag Ultrasonic Sequence Timeout: Flag is set when the ultrasonic measurement sequence is longer than the configured pause time. Note that without pause the flag is always set. See **CR_USM_PRC**

24	**BNR_TM_SQC_TMO**[8]	b0	BIT	Copy of bit 8 of **SRR_ERR_FLAG** before clear Error Flag Temperature Sequence Timeout: Flag is set when the temperature measurement sequence is longer than the configured pause time. Note that without pause the flag is always set. See **CR_TM**
25	**BNR_TSQ_TMO**[8]	b0	BIT	Copy of bit 9 of **SRR_ERR_FLAG** before clear Error Flag Task Sequencer Timeout: Flag is set when the task sequencer was not idle when the subsequent measurement cycle started. Some processes may not have finished correctly or need more time for execution.
26	**BNR_E2C_ACK_ERR**[8]	b0	BIT	Copy of bit 10 of **SRR_ERR_FLAG** before clear Error Flag EEPROM Acknowledge: Flag is set when EEPROM communication signalled an acknowledge error, communication failed
27	**NOT_USED**	b0	BIT	
28	**BNR_CS_FWD1_ERR**[8]	b0	BIT	Copy of bit 12 of **SRR_ERR_FLAG** before clear Error Flag FWD1 Checksum: Flag is set when the checksum of FWD1 calculated by AS6031 is not the same as stored in ***FWD_R_FWD1_CS***
29	**BNR_CS_FWD2_ERR**[8]	b0	BIT	Copy of bit 13 of **SRR_ERR_FLAG** before clear Error Flag FWD2 Checksum: Flag is set when the checksum of FWD2 calculated by AS6031 is not the same as stored in ***FWD_R_FWD2_CS***
30	**BNR_CS_FWU_ERR**[8]	b0	BIT	Copy of bit 14 of **SRR_ERR_FLAG** before clear Error Flag FWU Checksum: Flag is set when the checksum of FW user code calculated by AS6031 is not the same as stored in ***FWD_R_FWU_CS***
31	**BNR_CS_FWA_ERR**[8]	b0	BIT	Copy of bit 15 of **SRR_ERR_FLAG** before clear Error Flag FWA Checksum: Flag is set when the checksum of FW ScioSense code calculated by AS6031 is not the same as stored in ***FWD_R_FWA_CS***

6.2 Firmware Variables in RAM-part of FWD

The following cells of FWD are used as RAM by the ScioSense firmware or modified after recall. Note that FWD is an NVRAM, which consists of a volatile RAM part and a FLASH part for permanent storage.

[8] In AS6031_AS6040_REG_A1.h the definitions of bits 15 to 31 are based on the copies in SRR_ERR_FLAG, e.g. BNR_TDC_TMO_FW = 0 (bit 0 of SRR_ERR_FLAG, instead of 16)

The RAM can be used as usual, but at a recall the data from the FLASH part is transferred to RAM and overwrites its former content. For a complete list of FWD cells, see chapter 5.

Table 18: Firmware Variables in FWD

FWD Cell #	FWD Cell Address	Variable Name	Description and (if applicable) default value	Format
0	0x100	*FWD_R_FLOW_ VOLUME_INT*	(optional): Integer part of negative flow volume in cubic meters - internally used if a negative flow limit is defined in cell # 93.	fd0
1	0x101	*FWD_R_FLOW_ VOLUME_FRACTION*	(optional): Fractional part of negative flow volume in cubic meters - internally used if a negative flow limit is defined in cell # 93.	fd32
5	0x105	*FWD_ERROR_COUNT_21*	(optional) Temporary storage of error counts 2 (B3, B2) and 1 (B1, B0) (see section 8.3 Set to 0x00000000 if error counters are used	int
6	0x106	*FWD_ERROR_COUNT_43*	(optional) Temporary storage of error counts 4 - each error (B3, B2) and 3 - hardware errors (B1, B0) (see section 8.3); Set to 0x00000000 if error counters are used	int
108	0x16C	*FWD_R_CD*	Watchdog disable code Without ScioSense firmware, 0x48DBA399 disables the watchdog while any other code enables it. With ScioSense firmware, the watchdog can't be disabled, but 0x00000000 disables configuration restore after recall. A reset should be applied after a POR. Proposed stored value 0x95659C6A	bits

6.3 Reading Data

With the firmware running autonomously, the customer has the following choices to read data[9]:

- Typically, the desired results from the firmware are read out over SPI interface. Access to these RAM cells is possible after any measurement and is done independent of the chip's firmware – it requires no programming work on the chip. Communication over those interfaces can be organized in various ways:
- When operating without firmware, new measurement results must always be read out before the next measurement. This is typically done using an interrupt issued by the chip after each measurement, which of course generates a high amount of data traffic.

[9] The unusual case of permanent sensor temperature measurements (TM_RATE = 1) needs special treatment for reliable communication with the chip when operating with ScioSense firmware. Please contact ScioSense when you want to operate the chip with this setting.

- With firmware, the data traffic can be minimized since the firmware stores and evaluates intermediate measurement results. Thus, a firmware can provide processed results after even long time periods without communication. Then the decision on how and when the controller retrieves data from AS6031F1 depends, among others, on the following criteria:
  - Is it required to have the data in external controller always up to date? Which update rate would be acceptable?
  - Is it possible to retrieve new data on request (for example when results are requested from outside)? How much delay time is acceptable for such a process?
  - How should an error case be handled? How is a permanent and error-free result data storage guaranteed? How much data loss is acceptable in case of serious errors?
  - These factors have to be balanced against the power consumption costs caused by communication and by waking up the external controller.
- Particular error messages can be configured to issue an interrupt to initiate communication (see 5.8). While many error events are handled by the firmware, some (unexpected ones) should be monitored by the external controller, for example checksum errors which indicate chip memory failure. The external controller should also monitor the chip's real time clock to identify preceding reset events, for example watchdog resets (even though unexpected). See chapter 8 for more details.
- Alternatively, a AS6031F1 based flow meter can directly replace a mechanical device when using the built-in standard pulse interface for flow volume counting. External data display, storage and further evaluation is done in the same way as with mechanical devices, in the simplest way by a counter. The pulse interface can signal errors, too.

In general, permanent storage of result data must be done outside AS6031. It may, however, be acceptable to update for example flow volume data at a low rate (say every hour) and to accept the mistake caused by some (unexpected and presumably extremely rare) fatal error event which may erase the stored flow volume on AS6031. The last resort of AS6031 in case of a fatal error is a watchdog reset, after which all stored measurement data will be initialized to zero, including flow volume. This will only happen if it is the only way the chip can return to normal operation (serious data corruption after a power drop, or chip operation blocked by remote commands; note that in both cases the stored data is already getting wrong).

With all these considerations in mind, some proposals for stable and efficient communication between chip and controller can be made:

- Reduce communication to a low level for low power consumption.
- But store important results outside and at an acceptable rate.
- Use error-triggered interrupts to inform the controller quickly about errors (see section 5.8)
- Synchronize communication with chip operations; else wrong data may be read out:
  - Use interrupts to synchronize: Either regular interrupts like "End of Task Sequencer" (configurable) or, for arbitrarily reduced communication rates, a communication request over remote command. The ScioSense firmware will answer with a synchronous firmware interrupt when ready for communication. The ScioSense firmware will in addition reset the communication request.
  - Or by keeping track of chip operations over time and accessing the chip only during IDLE. To synchronize internal and external clocks, the task sequencer time in **SRR_TS_TIME**

(0x0E9) can be used. This register contains the time elapsed since the last task sequencer trigger.

– Or access data at any time, but control the validity of results, e.g. by writing and reading a pair of RAM cell at beginning and end of communication (writing one value to a RAM cell and directly reading it back is not a valid test, since the buffer of the interface may just return the last written value; write two values sequentially and read them back as validity check).

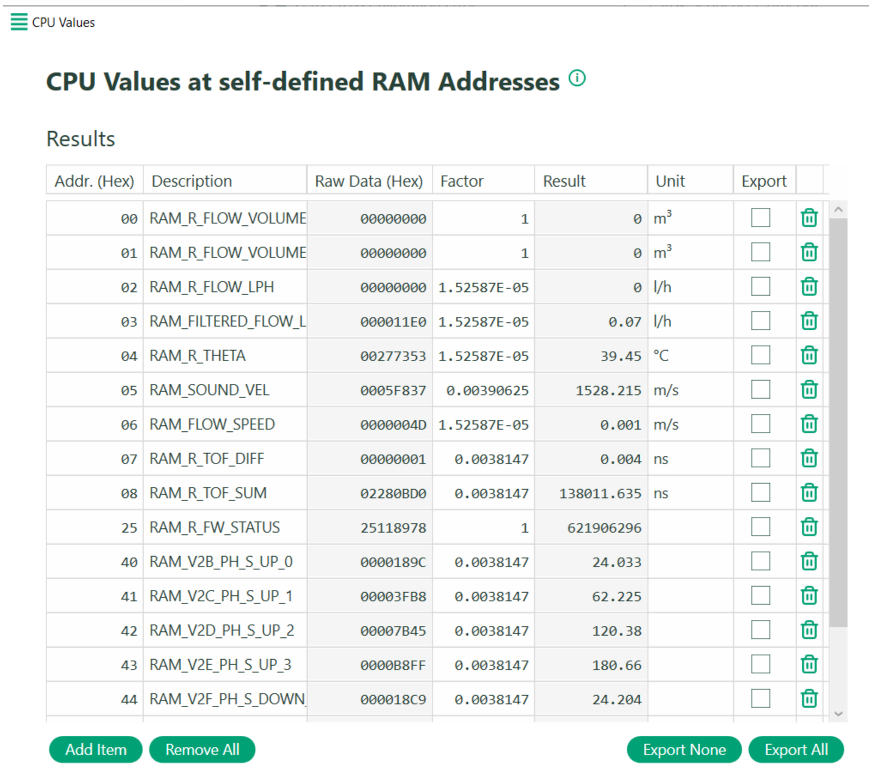
Despite all security remarks on possibly losing volatile data on AS6031, the stored data is secured to a high degree, as long as sufficient operation voltage is available. Typically, stored RAM data remains down to operation voltage levels of about 2 V. The ScioSense firmware signals low voltage by a "low" on GPIO6, which can be evaluated from outside even when chip communication is not possible anymore. The most important measurement result, the flow volume, can be secured against data corruption by an optional redundant storage (**BNR_FWCONF_VOL**).

6.3.1 Reading via GUI

The most important results are directly displayed in the UFC evaluation software. Select menu item CPU to open the CPU values window and menu item Monitoring to open the flags page (with option for a separate window, too). ScioSense provides a UFC project file that already includes the right settings and descriptions for the CPU results and the flags.

The results can also be displayed in a graph over time by clicking "Open CPU Graph".

The software is flexible and allows to display content of any RAM cell.



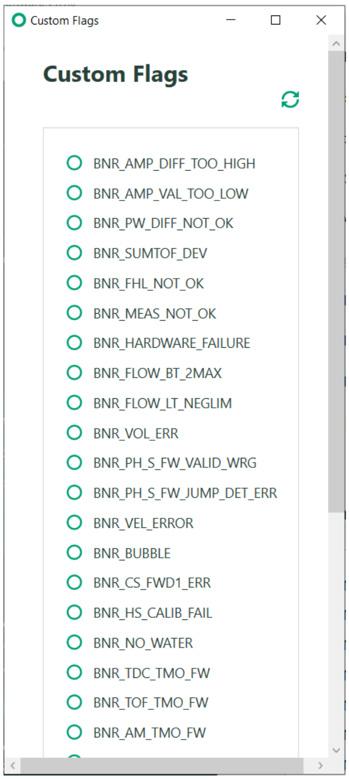


Figure 10: CPU values and flags

7 First Hit Level Setting

7.1 First hit level selection criteria

The receive signal of an ultrasonic flow meter is typically a more or less slowly rising sine wave burst. Time-of-flight (TOF) measurements usually refer to the distance between the start of the fire pulse signal and a number of selected zero crossing points of the receive wave, which are interpreted as hits by the TDC. To achieve comparable TOF values, a decision has to be taken which of the various receive hits are evaluated. This decision deserves careful investigations since it has a strong influence on measurement quality and operation stability. Relevant criteria are:

- Which noise level is related to a hit?
  Zero crossings of the early waves with low amplitude create hits with higher noise.
- How strong is the influence of device tolerances on hits?
  After the last fire pulse is received, the receive signal decays with a frequency which is determined not by the fire frequency, but by the resonance frequency of the particular receiving transducer. These waves and their corresponding hits are not suitable for device-independent measurements (decaying waves in picture below).
- How stably can a particular hit be identified?
  For operation stability, and in particular for a correct calculation of sound velocity and temperature, a reliable identification of always the same receive hit as reference for the measurement is crucial. This so-called first hit must remain the same over the whole operation time. Flow is calculated from differences of TOF_UP and TOF_DOWN (DIFTOF), which - in well-designed systems - does not depend on the first hit, as long as the first hits for both TOFs are the same. Otherwise, the measurement is wrong and may be widely misleading. So, a stable and unambiguous first hit detection is also important for the flow measurement itself.

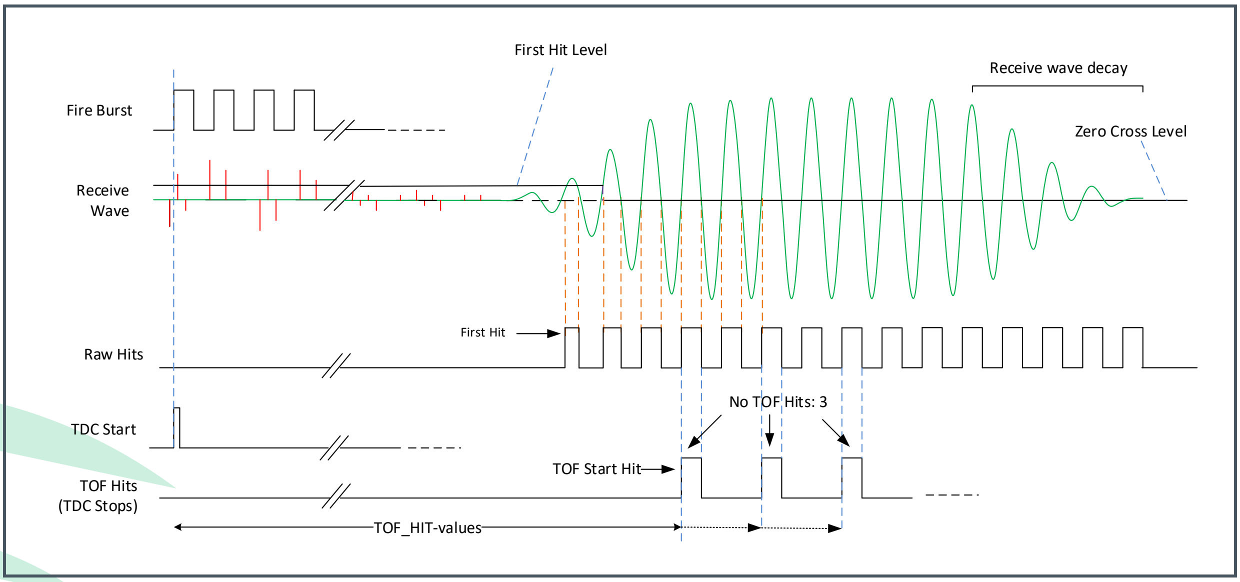


Figure 11: First hit detection

The selection of hits for evaluation depends on all three factors and should be optimized for a given spool piece, current consumption specification and accuracy demand. With a given first hit, AS6031 permits the selection of a sum of hits with configurable start hit and length. It should be noted that in general the first hit itself and the second hit are not suitable for evaluation since they are involved in the first hit detection process and are not exactly corresponding to a well-defined zero crossing point of the receive signal.

The remaining part of this section concentrates on the first hit detection process and options and also on the regulation algorithms offered by the ScioSense firmware.

AS6031 offers a number of auxiliary measurements and settings to support stable first hit detection:

- Peak amplitude measurement of selected waves of the receive signal.
- First hit pulse width ratio (PWR) measurement to optimize the quality of the first hit detection.
- Automated zero crossing calibration.
- First hit level setting to detect the chosen wave at some particular amplitude.
- Alternatively, a start hit delay to determine the start hit not by detecting some particular first hit, but after a user-defined delay time.

The last point, using a start hit delay, is a well-known method, not depending on any amplitude measurements. It usually requires accurate regulation of delay times since the TOF values, which relate to the delays, change strongly with flow and temperature. The ScioSense firmware does not support this method since it depends highly on the device. Customers who apply the start hit delay need to implement their own regulation schemes in their user firmware or their external controller.

By setting a fixed first hit level (FHL), a particular first hit can be detected as reference for the measurement. This detection is stable as long as the chosen FHL is sufficiently different from the peak wave amplitudes around the first hit. When it is lower than the peak amplitude of the preceding wave, one hit before the desired one will be detected, and when it is higher than the next peak wave amplitude, one hit after the desired one is detected. Thus, the FHL value has a usable voltage range between these two wave amplitudes. For maximum clearance, which means optimal EMI immunity, FHL should be chosen in the middle between these two amplitudes. The pulse width ratio (PWR) measurement is an indicator for the right selection of FHL. It approaches zero for too high FHL (measured pulse width gets small close to the wave peak), and for too low FHL some maximal PWR value is reached just before the first hit detection jumps to the preceding wave. The maximum PWR is determined by the rise time of the receive signal, which means by the quality factor of the employed transducers. The optimal FHL is found where a PWR between this maximum pulse width and zero is measured. In summary, a first selection of FHL level can be done the following way:

1. Measure the first few receive signal wave amplitudes using AS6031 (You can do that using an oscilloscope, but the probe head’s impedance will influence the measured peak values, often too strongly).
2. For maximum clearance, select the wave at the highest amplitude difference as position for the chosen first hit (this selection may be revised for more sophisticated reasons, see point 4. below).

Set FHL to the middle value between these two wave amplitudes and measure the pulse width ratio. This would be the nominal pulse width ratio for the optimal FHL. This value gives some first information about the suitability of the selection, and of the transducers as well. Values of 0.6 – 0.7 would be quite good for this nominal pulse width ratio. If the result is significantly smaller, the quality factors of the employed transducers may be too high. This causes long rise and decay times, with small differences between subsequent wave amplitudes, as well as a high sensitivity against production tolerances. Both is undesired and should be avoided.

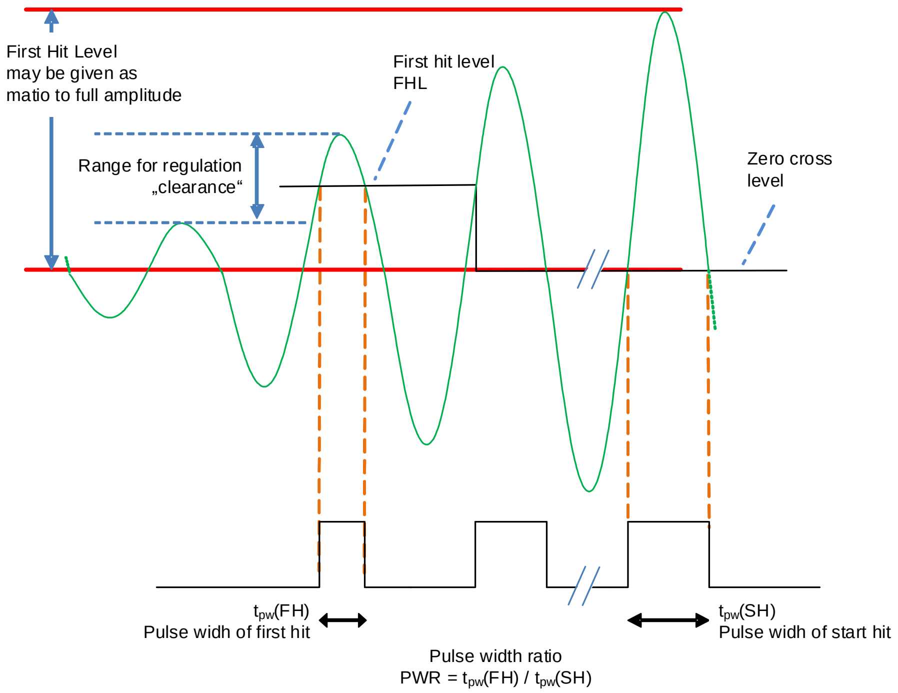


Figure 12: First hit detection

3. This is the first important step of the FHL selection. Until here, it is not related to ScioSense firmware and can be used in remote control operation as well. But now it should be noted that the result of the selection process described above can change over temperature, production tolerances and aging; for some devices and transducers it may even depend on flow and pressure. The problem is to obtain reliable and stable first hit detection under the influence of all these different factors. So, a fourth step should follow:
4. Check how stable the chosen selection is against temperature, tolerances, aging, flow and pressure. It can happen that, regarding temperature changes, the selection of a different FHL with less clearance, but better temperature stability may be preferable (see picture below). If strong dependences of optimal FHL on actual flow or pressure are observed, the performance of the transducers should be checked critically. Usually, transducers which respond to pressure changes with strong signal changes are not optimal for flow meters.

Regarding the other influence factors tolerances and aging, only an active reaction on parameter changes can improve the situation. This may be done on customer side, for example by defining an individual FHL in relation to the receive signal amplitude, or even actively by a user firmware code or an external controller. The ScioSense firmware has several regulation methods implemented which gives the user some choice for an optimized operation stability. The next section describes these methods.

The following figure demonstrates how different receive wave amplitudes - and with them the corresponding FHL levels – can vary over temperature for different spool pieces (#1 to #4). Accordingly, different spool pieces require different FHL regulation methods for stable first hit detection.

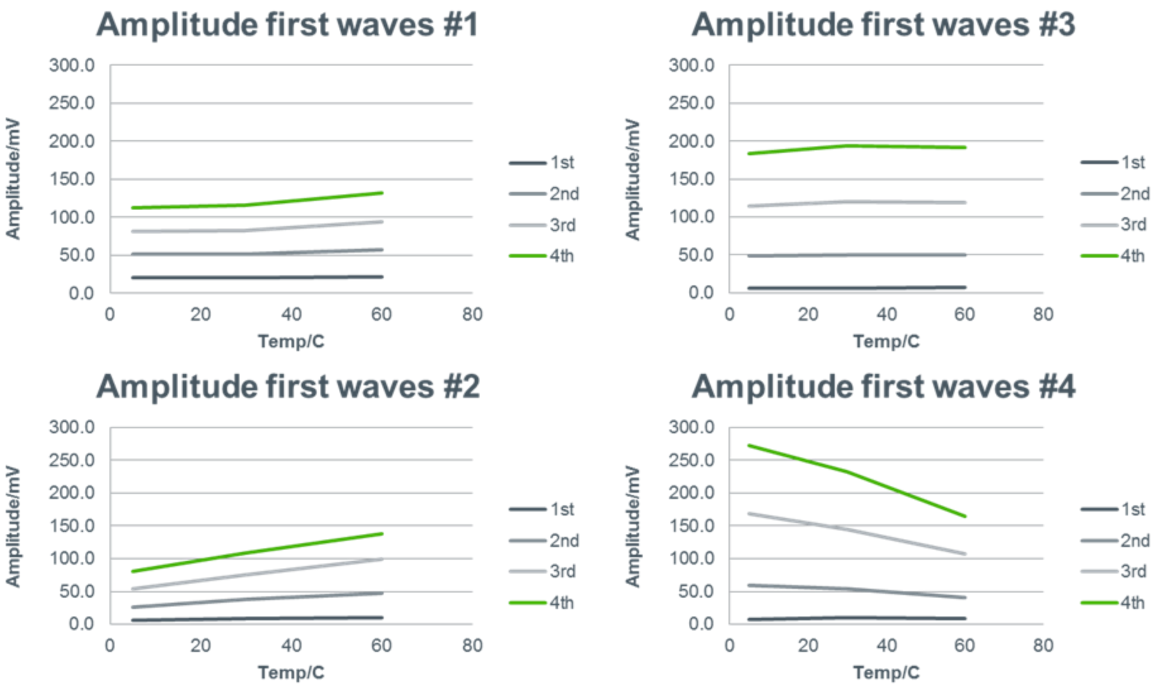


Figure 13: First wave amplitudes

7.2 First Hit Level Regulation Methods

As discussed above, there may be reasons to adapt the first hit level (FHL) according to environment parameters. The ScioSense firmware offers three different regulation methods and three configuration options, to permit an optimal choice for operation stability of an individual device.

In general, all methods aim at controlling and, if necessary, adapting FHL such that the device operates stably at the chosen working point. This is particularly important after situations where no valid measurement is possible – typically after a no-water/hardware error situation, but also after critical error situations like a strong appearance of bubbles.

Most of the time, the operation of an optimized device is error-free. All FHL regulation methods react slowly. This increases the time needed to return to normal operation in failure case but adds a huge amount of operation stability in normal operation. The actual implementation is that FHL regulation is done every 32nd measurement, or even less (see option C below). This way, the regulation is not confused by transients and does not itself confuse measurement operation by sudden changes.

It should be pointed out that any regulation also adds some amount of failure probability. So, the general recommendation is to examine carefully which level of regulation is needed, maybe even redesign the system to add stability, and then decide for a minimum of regulation mechanisms for highest operation reliability.

The available regulation methods in overview:

1. Keep FHL constant: This is of course the most stable method, but it is suitable only if the spool piece behaves stably enough over temperature. Whatever additional option may be chosen; this method enforces a given FHL every 32 measurements.
2. Return to a trusted FHL in case of inconsistency: It may be advantageous to let FHL freely regulate, for example for optimized pulse width (see option A below), but then return to a less optimal, but stably operating FHL value when problems appear.
3. Offset trusted FHL: The method also assumes that one trusted FHL level exists which ensures a desired working point over temperature and other parameter changes. But it may happen that this point has low clearance between the wave amplitudes and thus is not the best choice. Instead, a different FHL is used for operation, at a given time offset, meaning a given number of hits away from the trusted level. So, the method temporarily switches every 32 measurements to the trusted level to check the currently operating FHL for consistency with the user-given time offset condition. In case of inconsistency it starts regulating.

Any of these methods can be combined with the following options:

A. Regulate PWR: FHL is adapted every 32 measurements (16 measurements offset from the FHL regulation methods above) to achieve a user-defined PWR. This option is useful to optimize EMI immunity. Such a regulation must be combined with a method for consistency check (like methods 2 - 3 above), else it may get stuck at undesired FHL levels after, for example, no-water situations. Since this regulation can contradict an active FHL regulation, it is always temporarily switched off when FHL regulation gets active. Option A is activated by setting ***FWD_PW_NOM*** to the desired nominal PWR value, gine in fd7 format. Setting the variable to 0x0 switches off the option.

B. Define FHL as ratio to receive amplitude: Instead of defining FHL levels as fixed voltages, this option interprets FHL values as ratios to the measured amplitude. Option B is useful to automatically compensate production tolerances and aging changes. Of course, this adds some amounts of uncertainty since the currently measured receive amplitude may be corrupted, for example by bubbles. The ScioSense firmware incorporates precautions to identify corrupted measurements, it will never modify any FHL level based on a measurement which is considered questionable.
   Option B is activated by setting bit **BNR_FWCONF_FHL_RATIO** in ***FWD_FW_CONFIG***.
C. Activate FHL regulation modes only in failure case: This option disables the regulations of method 1 - 3 as long as operation is considered correct and free of errors. It does not disable PWR regulation. Only in case of recognized measurement errors, it activates FHL regulation.

   The advantage of this option is that it avoids frequent intentional measurement interruptions caused by the regulation methods. The disadvantage is that FHL is not checked unless an actual error appeared. Switching on this option also prevents FHL regulation from correcting errors which are only recognized by FHL consistency checks. Thus, it may happen that the device operates for long times in an undesired working point without being noticed. Decide to activate this option only after careful examination.
   Option C is activated by setting bit **BNR_FWCONF_TESTMODE** in ***FWD_FW_CONFIG***.

See section 5.9 Parameters for FHL Regulation for details about the individual parameters.

Formally, all options can be combined with any regulation method. It should be understood that a few combinations are not reasonable. For example, activating PWR-regulation (option A) combined with fixed FHL (method 1, and without option C) will always cancel the steps of PWR regulation.

In zero flow case, the measurement rate is usually slowed down. In standard configuration (bit 17 of ***FWD_FW_CONFIG*** = 0), this reduction of measurement frequency is disabled whenever FHL regulation is active, to avoid extremely slow regulation.

Finally, a different FHL value is defined in FWD cell 107: ***FWD_R1_FHL_VALUE*** (always in mV, never interpreted as amplitude ratio) as fallback option. This is the start value when the chip is switched on. In case of TOF timeout (typically no-water, but may also be some error case), the firmware switches temporarily (every 32 measurements) to this user-defined FHL just to check if at this FHL value normal operation may resume. This is just a safety measure against unexpected cases of corrupted operation.

The parameter settings for configuration of the different FHL regulation methods and options are described in section 5.9 in overview, and in more detail in the following functional descriptions.

7.2.1 Method 1: Keep FHL constant

This method is simple and straightforward: It enforces the user-defined FHL value in ***FWD_FHL_USER*** every 32 measurements (or, if option C is activated, in error case only). It offers a maximum of operation stability on firmware side, but of course it is suitable only if the spool piece behaves stably enough over temperature and other parameter changes.

Option A (Regulate PWR) can be activated, but its regulations get cancelled after another 16 measurements unless option C (Activate FHL regulation modes only in failure case) is active, too.

Option B (Define FHL as ratio to receive amplitude) can be activated in addition to compensate production tolerances and aging; see comments above for general pros and cons of this option.

FHL regulation method 1 is activated by setting **BNR_FWCONF_FHL** to 0x00. It only uses ***FWD_FHL_USER*** in addition. See sections 4.9 for details.

7.2.2 Method 2: Return to Trusted FHL

This method is similar to method 1 above, but it does not generally enforce the user-given FHL value. It rather regularly checks the currently active FHL against a trusted FHL value given in ***FWD_FHL_USER***. It does that by switching to a test mode every 32 measurements or, with option C active, only in error case. The criterion for consistency is the deviation between the SUMTOF measured in test mode and the average of the last eight SUMTOF measurements (with error values and outliers excluded). If the deviation is larger than ***FWD_TOFSUM_VAR_LIM***, the currently active FHL is considered inconsistent, and gets replaced by the trusted FHL value.

Option A (regulate PWR) can be activated and has enduring influence on the active FHL, in contrast to method 1. Thus method 2 is suitable for optimized EMI immunity, but of course only when the spool piece features at least one trusted FHL, sufficiently stable with temperature, with acceptable EMI clearance (see picture above).

Option B (define FHL as ratio to receive amplitude) can be activated in addition to compensate production tolerances and aging; see comments above for general pros and cons of this option.

FHL regulation method 2 is activated by setting **BNR_FWCONF_FHL** to 0x01. It uses in addition ***FWD_FHL_USER*** and ***FWD_TOFSUM_VAR_LIM***. See section 5.9 for details.

7.2.3 Method 3: Offset Trusted FHL

This method adds one more feature to method 2: Instead of checking directly against a trusted FHL, it checks for a user-defined SUMTOF-offset in ***FWD_TOF_SUM_DELTA*** (two times the nominal time difference in the plot below). As always, it does that by switching to a test mode every 32 measurements or, with option C active, only in error case. The check for consistency is now that the deviation of the SUMTOF measured in test mode and the average of the last eight SUMTOF measurements (with error values and outliers excluded) is closer than ***FWD_TOFSUM_VAR_LIM*** to ***FWD_TOF_SUM_DELTA***. If the deviation to ***FWD_TOF_SUM_DELTA*** is larger than ***FWD_TOFSUM_VAR_LIM***, the currently active FHL is considered inconsistent, and FHL regulation gets active: Depending on the sign of the deviation, the active FHL is increased or reduced until the consistency check is successful again.

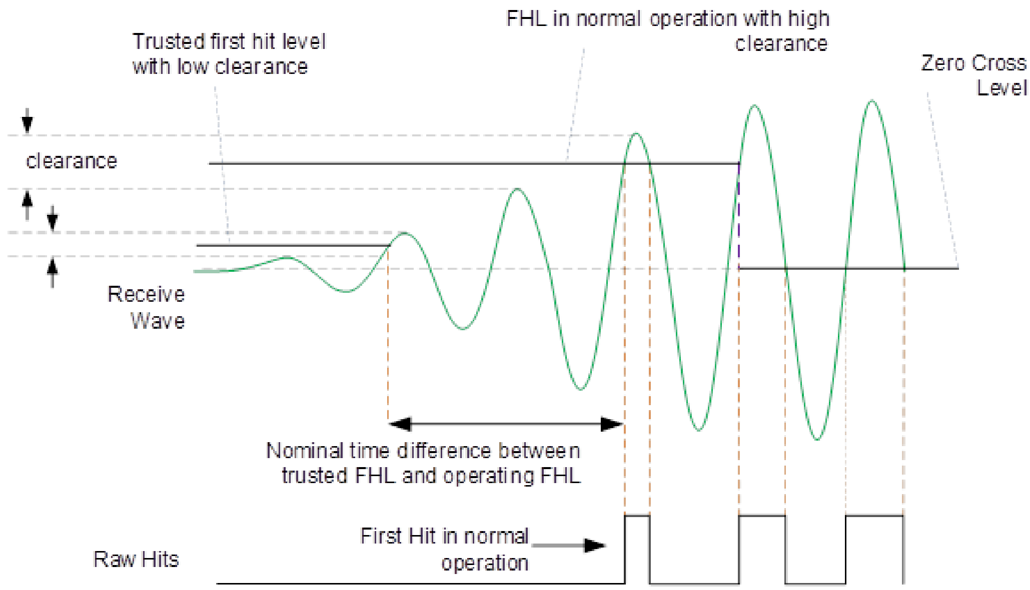


Figure 14: Offset-trusted FHL

The essential difference of this method compared to methods 1 and 2 is that the active FHL can be arbitrarily different from the trusted FHL. For example, the trusted FHL may be suitable for detecting the second wave as first hit, over all temperatures and tolerances. But it may feature low EMI clearance (small differences between neighboring wave amplitudes), such that a later wave with higher amplitude and higher clearance may be preferable, for example the fourth wave. Now the corresponding FHL for the later wave may not be stable over temperature and not suitable as trusted FHL. Then this FHL regulation method 3 uses the “small” trusted FHL as reference only and regulates the active FHL such that its SUMTOF deviates by a user-defined value ***FWD_TOF_SUM_DELTA*** from the trusted FHL, for example by 4 µs. If, for example, the measurement frequency is 1 MHz, a SUMTOF difference of 4 µs corresponds to a first hit which appears two periods (2 x 1 µs) later than the SUMTOF measured with the trusted FHL. This way, a first hit with high EMI immunity can be chosen even though it may not be suitable as trusted FHL. Still at least another trusted FHL must exist to use this method.

It should be noted that this method typically causes temporary SUMTOF and other errors when regulation is active. This happens every 32 measurements, such that there is no actual influence on measurement results.

Option A and Option B can be activated with the same implications as under method 2, see above.

FHL regulation method 3 is activated by setting **BNR_FWCONF_FHL** to 0x10. It uses in addition ***FWD_FHL_USER***, ***FWD_TOFSUM_VAR_LIM*** and ***FWD_TOF_SUM_DELTA***. See section 5.9 for details.

8 Error Handling and Operation Safety

Flow meter systems have to face two typical problems: They have to handle one special operation case when there is no medium in the spool piece (no-water case), and they have to operate stably and reliably over long times without any maintenance. In addition, ultrasonic spool pieces suffer from bubbles which may also appear from time to time. Thus, a well-designed ultrasonic flow meter device should basically handle three different types of events:

- Usual long-term interruptions of normal flow measurement, like no-water: Such events are expected to happen. They should not modify the measured results, and normal operation should be resumed quickly and reliably after the event.
- Usual short-term interruptions, like bubbles: Such events will influence the measurement results, since at some level of distortion the flow measurement will be wrong or impossible. But their influence should be kept under control: The quantity of events should be known, and they should not interrupt normal operation. As far as possible, corrections should be done.
- Unusual, even very rare failure and distortion events: Considering even very rare and improbable events is appropriate when reliable long-term operation of huge amounts of devices should be achieved without the need or possibility of regular maintenance.

This chapter discusses the tools and processes the ScioSense firmware offers to handle the majority of such special or failure events. Regarding rare failure events, like a possible corruption of stored data or configuration, ScioSense firmware development was done under the assumption that the chip should resume normal operation even in case any arbitrary volatile memory cell of the chip may have lost its content. It should be noted that this is not an expected event. Actually, data corruption of volatile cells was only observed at permanent supply voltage drops below 2 V. However, the firmware is designed to handle even extremely improbable cases, for example caused by some temporary strong electromagnetic interference, to achieve highest reliability for a mass production device.

In addition, error handling is strongly related to first hit regulation, since a wrong first hit selection will cause errors on the one hand; on the other hand, corrupted measurement data should never be used for first hit regulation.

8.1 Error Handling

The following list of events and operation modes ranges from normal operation over typical special conditions like bubbles or no-water to unusual and unexpected severe error events. It discusses the tools and processes the ScioSense firmware offers to handle the majority of these events, as well as proposals for additional precautions on side of the external user. The setting of corresponding parameters is explained in section 5.7 Parameters for Error Handling.

Table 19: Error cases

Description	Behavior of AS6031	Occurence	External User Tasks
Normal operation	Flow and temperature are calculated according to calibration, Flow volume is stored in volatile memory.	Normal operation	Further evaluation, storage and communication of results
Singular error events, caused e.g. by bubbles or EMI	Singular events are always ignored by the implemented outlier-filter (no matter if they are identified as error or not). They do not influence any measurement result. They may still be temporarily signalized by an error flag.	Frequently, depending on overall device design and application situation. With recommended circuitry on test stand, EMI events are typically never observed.	-
Short sequences of errors e.g. by bubbles	Sequences of not more than four errors are ignored not by filtering, but by replacing them with former valid measurement values	Frequently, depending on overall device design and application situation. Typically caused by bubbles.	Sequences of errors must be identified by their measurement results. User should define error limits.
No-water or hardware error	Recognize and indicate no-water situation or hardware error. It also stops adding volume and does regular checks to see if condition persist	Normal operation situation	No-water is mainly indicated by a low amplitude. User has to set the low-amplitude limit suitably.
Long sequences of errors, e.g. by impurities or large amounts of air in the water	Recognize and indicate error situation; stop adding volume	Should be avoided by construction – no reliable measurement possible	Sequences of errors must be identified by their measurement results. User should define error limits
FHL regulation errors	Should be resolved after some regulation time, according to the chosen regulation mode.	Typically after an error situation or after no-water	User has to carefully evaluate the behaviour of his spool piece and configure the suitable FHL regulation method.
Corruption of volatile data in AS6031	Several possibilities: Can range from non-recognized small changes to corrupted operation, ending in watchdog reset; ScioSense firmware issues a data recall every hour, such that corrupted volatile data is overwritten and thus revised regularly. The stored flow volume	Never observed even in extensive tests. Still the firmware has procedures implemented to resolve even such very improbable situations. Note that corrupted data can of course be generated by intentional	User should in any case control the real time clock to identify resets – AS6031 will not remember any volatile data after a reset, so it is important to recognize such events. It is proposed in addition to set a maximum volume/h limit on controller

	has an optional additional security storage.	or unintentional wrong input from outside.	side to limit the influence of unrecognized errors of that type.
Corruption of permanently stored data in AS6031	AS6031F1 can't correct that, but the ScioSense firmware regularly controls the checksums of stored data to recognize and signal such errors.	Should not be observed and has to be considered a hardware defect. May also be caused by wrong storage processes over the interface.	The checksums initially stored in firmware data must be correct to enable regular checks. User must evaluate the error flags and may try refreshing the stored data in error case.
Low voltage	AS6031F1 indicates low voltage, if configured, by an error flag and by a low signal on GPIO6. It takes no further action and keeps operating as long as possible.	Always at the end of battery lifetime	User has to evaluate the low voltage flag or, preferably, the signal on GPIO6, and take external precautions.

In summary, the following precautions are proposed on user side to support optimal reliability and operation stability:

- In development and evaluation:
  - Set FHL regulation parameters and error limits carefully for stable regulation and reliable detection of bubbles and other errors.
  - Make sure to store the right checksums in FWD cells 124 to 127 (see section 5.7)
  - Select which error flags should be observed by the controller and configure error counters and interrupts accordingly. It is the free decision of the user which corrective actions, in addition to the built-in processes, may be taken. Note that ScioSense firmware already takes precautions through its built-in processes to resolve any temporary problem.
- In operation (tasks for the controller of the flow meter system):
  - Watch the relevant error flags which indicate problems during running operation and take appropriate corrective actions (as freely defined by user as result of the evaluation).
  - Monitor the real time clock of AS6031F1 in **SRR_TS_MIN_SEC** and **SRR_TS_HOUR**, to recognize resets (for example caused by watchdog events). After a reset, AS6031 has lost all volatile data and restarts with the stored configuration. Resets may also be recognized by storing some value in an unused RAM cell and checking if the cell returned to zero.
  - The following errors can't be resolved by AS6031 chip or ScioSense firmware and must be treated (by corrective action or at least recognition) on side of the external controller:
    - Low voltage, checksum errors, uncorrectable volume storage error.
    - Low voltage check is supported by AS6031 through an adjustable low voltage limit and an error flag. In addition, low operation voltage is signaled through the firmware by a "low" level on GPIO6. If voltage increases again, this signal is revised every full hour.

– In addition, it is proposed to set an upper limit to total flow volume/h to limit the influence of any unrecognized data corruption. Such events are highly unexpected, and this would just be a precaution which acknowledges the fact that even very improbable events may still happen sometimes.

The following precautions are automatically enforced by the ScioSense firmware:

- Watchdog is active – in case of fatal errors a reset happens to resume operation
- Hourly recall is configured (with TM_RATE=1, recall with every measurement)
- TOF rate is read at every measurement from *FWD_USM_PRC*/B3

8.2 Error Handling Flow Chart

The flow chart below sketches the sequence of error handling in the ScioSense firmware.

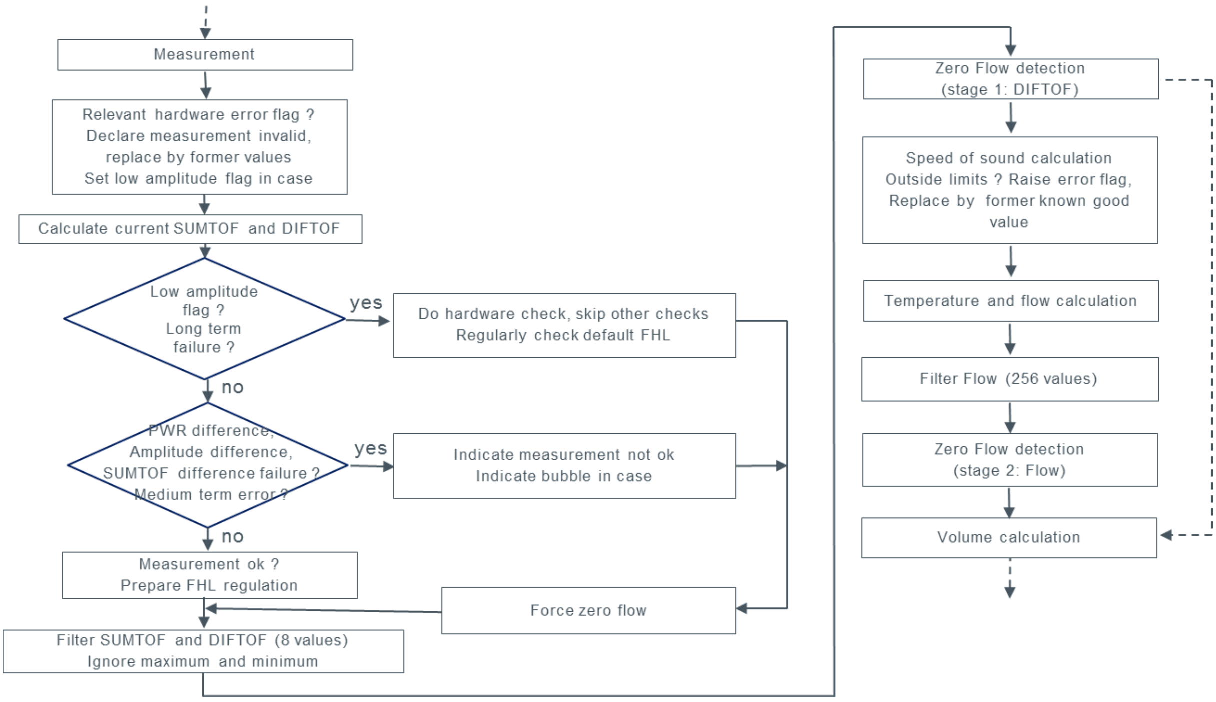


Figure 15: Error handling

A number of decisions depends strongly on the frequency of events and the duration of errors. Information on error counts can be retrieved from the error counters described in the next section. The following numbers are important for error handling and flow calculation as well as for first hit level regulation and zero flow detection:

- 1: Single measurements which deviate from the last seven measurements are always ignored by the averaging filter, no matter if an error flag was raised or not.
  Independent from that, single measurement which come with relevant error messages are in general ignored and, to some extent, replaced by former results. Such measurements are marked not o.k. At the first error-free measurement the "measurement not o.k." flag is cleared.

- 4: After four measurement errors in a sequence, the measurement process is considered corrupted and an internal flag BNR_MEAS_FAILURE_ALERT (bit 10 of *RAM_R_FW_STATUS*) is set. Then flow is forced to zero until the next valid measurement.
- Bubble detection flag is raised when four or more consecutive measurements appeared which violated the limits given in *FWD_TOFSUM_VAR_LIM*, *FWD_AM_DIFF_LIM* and *FWD_PW_DEV* (see section 4.7)
- 8: Fixed length of the input DIFTOF and SUMTOF filter.
- 32: FHL and PWR regulations take place each 32nd measurement, with an offset of 16 measurements between FHL and PWR. This way, the regulations don't interfere mutually and with the input filter length. In consequence, all regulations are comparably slow, which increases operation stability, but also increases the time before inconsistencies are resolved by regulation.
- FHL and PWR regulation is never done based on measurement results which are considered corrupted. As far as applicable, former results are utilized instead.
- *FWD_LONG_TERM_ERROR* (FWD cell 78): After this number of low-amplitude errors, the firmware switches into special configurations to distinguish no-water from no-water/hardware error.
- 16* *FWD_FLOW_AVG_FACTOR* (FWD cell 90): This number defines the length of the flow averaging filter, and thus the noise level of the averaged flow result, in particular of zero flow. Longer filters permit lower zero flow levels (limited by the quality and repeatability of the zero flow offset calibration). Of course, longer filters also mean longer settling times for averaged flow, and consequently longer times before the decision for zero flow is taken. Note that flow volume is calculated from unfiltered flow values. Note also that switching back into full flow mode is done not only when averaged flow exceeds the zero flow limit *FWD_ZERO_FLOW_LIMIT*, but also when the unfiltered flow result exceeds this limit by a factor of 8.
- *FWD_TOF_RATE_FACTOR* (FWD cell 89): This number determines the reduction of measurements in zero flow case. With BNR_FWCONF_FHL_ZEROFLOW (bit 17 of *FWD_FW_CONFIG*) set to 0, the measurement rate is reset to normal operation when FHL regulation gets active, to avoid too long regulation times.
- BNR_FWCONF_VLIM (bit 27 of *FWD_FW_CONFIG*) set to 1 limits calculated results for speed of sound to physically reasonable values (see section 4.7, FWD cells 74 and 77). Results which violate these limits are ignored and the last valid result is taken instead. In consequence, the calculated water temperature also remains unchanged until a new valid measurement appears.
- BNR_FWCONF_2MAX_NOZERO (bit 16 of *FWD_FW_CONFIG*): When this bit is set to 0, the calculated flow gets limited to 2* *FWD_R_PULSE_MAX_FLOW* (FWD cell 92). Higher results are ignored, and current flow is set to zero.
- *FWD_NEG_FLOW_LIMIT* (FWD cell 93) defines a negative flow limit. When exceeded, the current flow is set to zero. For details, see section 5.6.

An overview of all available error flags is given in section 6.1.3

8.3 Error Counters

To keep track of error events, the ScioSense firmware offers functions to count errors and store peak values of errors per hour. There are two types of counters: one adds up the number of consecutive measurement error events, separated into

- Ultrasonic measurement errors (*RAM_R_USM_ERR_CTR*),
- Amplitude measurement errors (*RAM_R_AM_ERR_CTR*),
- Low amplitude events (*RAM_LOW_AM_ERR_CTR*),
- Sensor temperature measurement errors (*RAM_R_TM_ERR_CTR*),
- Task sequencer timeouts (*RAM_TS_ERR_CTR*) and
- Ultrasonic measurement error events which prevent new flow calculation (*RAM_R_FHL_ERR_CTR*)

These counters are all reset to zero at the next correct measurement. They are used for error handling control and can be helpful for configuration check during development.

The other type of counter is optional and user configurable and adds up all configured error events within one hour. These counters are reset for each new hour, and their peak value is stored in ***RAM_ERROR_COUNT_21*** and ***RAM_ERROR_COUNT_43***. They can be used for statistics, long term evaluation and operation diagnostics. To switch on these counters, set bit **BNR_FWCONF_ERR** in ***FWD_FW_CONFIG*** (see section 8.1). Counter 4 or 3 (Bytes B3/B2 or B1/B0 in ***RAM_ERROR_COUNT_43***) are configured to count all errors or all hardware defined error flags, respectively. Counters 2 or 1 can be configured by setting the bits corresponding to the error flags of ***RAM_R_FW_ERR_FLAGS*** (see section 0) in their configuration registers ***FWD_ERROR_COUNT_CONF2*** or ***FWD_ERROR_COUNT_CONF1***, respectively.

For example, if counter 1 should measure how often the **BNR_BUBBLE** error flag of ***RAM_R_FW_ERR_FLAGS*** is raised within one hour, set ***FWD_ERROR_COUNT_CONF1*** to 0x00004000 (only bit 14 = position of **BNR_BUBBLE** set). Then ***RAM_ERROR_COUNT_21*** will count all bubble error flag events in bytes B1/B0 during the first hour and will be increased if in a subsequent hour the number of error events is higher. This way, the peak value of average error events can be measured. Counter 1 and 2 can be configured for any desired combination of error flags.

If the average error counters should be reset to zero by the external controller, it is necessary also to reset FWD cells ***FWD_ERROR_COUNT_21*** and ***FWD_ERROR_COUNT_43***, since in the RAM part of these NVRAM cells contains the current error count.

8.4 Error Interrupt

A user-defined combination of errors can be configured to issue a synchronous firmware interrupt in a similar way as the error counter configurations above: Set the desired bits according to error flag positions in ***RAM_R_FW_ERR_FLAGS*** (see section 7) and switch on the synchronous firmware interrupt in **CR_IEH** and FWD cell 112, respectively. This should be used to issue an irregular interrupt at a user-defined error (combination). A typical application of such an interrupt would be if AS6031F1 communicates its results only rarely to the system's microcontroller, but special events should be recognized immediately. For example, writing 0x00008000 in cell 83 ***FWD_ERR_INTERRUPT*** will issue an irregular interrupt on the interface as soon as no-water is detected.

8.5 Error Signals through Pulse Interface

ScioSense firmware can be configured to signal error conditions over the pulse interface, too. In case of error, the pulse output goes permanently high, and the direction output toggles with the TOF measurement frequency. This creates no additional pulse count and can be easily identified by some external readout device or master controller. With bit 22 of the firmware configuration register ***FWD_FW_CONFIG*** it can in addition be chosen if a no-water situation should also be signaled as error or not.

This way of signaling errors is not suitable if flow in both directions should be measured and signaled over two pins for different directions (this is the alternative configuration for the pulse interface). In this case error signaling through the pulse interface should be switched off. In case no reverse flow is counted, error signaling can be used as described even using only the positive pulse pin.

9 Guide to Operation - Step by Step

The subsequent sections describe in detail how to configure and operate AS6031F1. Every single step is supported by software and template files, but all these steps are necessary to customize AS6031F1 and its firmware. In overview, the necessary development steps are:

1. Define the suitable chip configuration for your particular flow meter system.
   Copy this configuration into the firmware data and download the firmware data into the chip. With the bootloader release code being set the configuration will be copied into the configuration registers after a power on reset.
   Also, when the firmware it is disturbed or interrupted, it will resume nominal operation by a watchdog reset, and when its configuration is modified, it will restore it latest after any full hour of the built-in real time clock **SRR_TS_HOUR** and **SRR_TS_MIN_SEC** (0x0E6 and 0x0E7)..

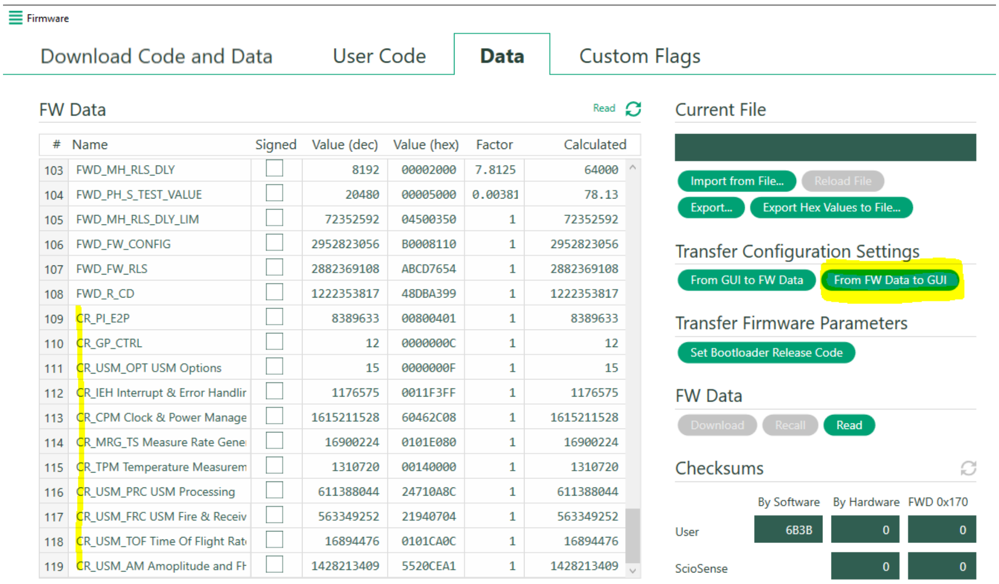


Figure 16: Configuration data in firmware data

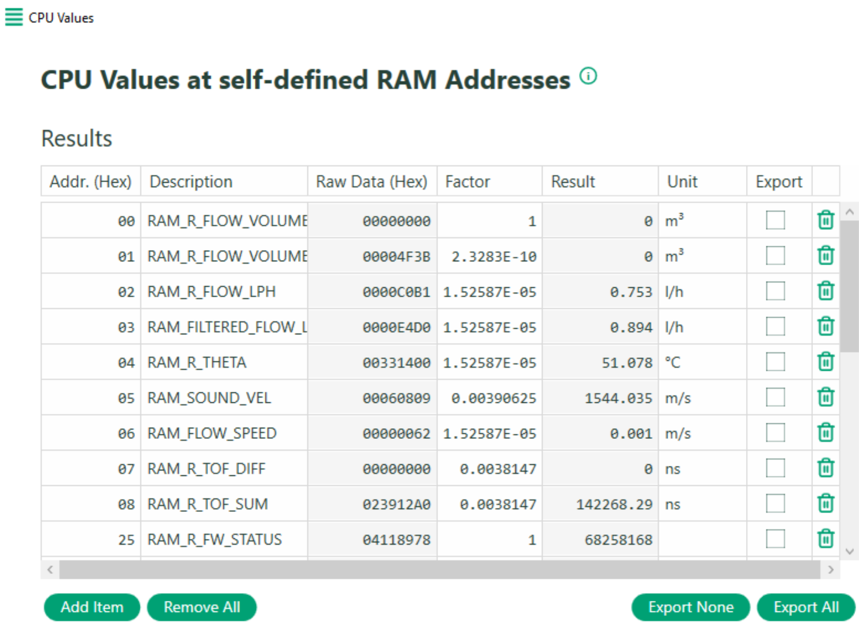
2. Characterize your system and do a master calibration for the particular type of spool piece. Therefore, collect pure TOF, AMP and PWR data over various flow rates and temperatures. See an overview on section 9.1 and for details user guide SC-001279-UG.
3. Set the appropriate limits and operating mode settings.
4. Scale the master calibration according to calibration measurements of each spool piece. See an overview on section 9.4 and for details user guide SC-001279-UG
5. Write both chip and firmware configuration as well as the individual calibration into a firmware data file and store this file into the particular chip which operates this spool piece. For details see section 9.3.

Of course, these steps will be repeated recursively during development. In production, only one individual calibration step is needed to define and store an individual calibration into each spool piece. Then AS6031 will be controlled typically by the microcontroller of the particular flow meter system, and all chip communication as well as the calibration scaling procedure will be done over that system.

The fastest way to proceed for an experienced user is surely to use the ScioSense evaluation software, to start with the provided template files and to modify them.

9.1 A First Impression

When starting to work with ScioSense firmware on AS6031F1, it is proposed to get a first impression from our demo kits with operating firmware. In the ScioSense UFC evaluation software, under the menu point CPU values the following window can be found:



Addr. (Hex)	Description	Raw Data (Hex)	Factor	Result	Unit	Export
00	RAM_R_FLOW_VOLUME	00000000	1	0	m³	
01	RAM_R_FLOW_VOLUME	00004F3B	2.3283E-10	0	m³	
02	RAM_R_FLOW_LPH	0000C0B1	1.52587E-05	0.753	l/h	
03	RAM_FILTERED_FLOW_L	0000E4D0	1.52587E-05	0.894	l/h	
04	RAM_R_THETA	00331400	1.52587E-05	51.078	°C	
05	RAM_SOUND_VEL	00060809	0.00390625	1544.035	m/s	
06	RAM_FLOW_SPEED	00000062	1.52587E-05	0.001	m/s	
07	RAM_R_TOF_DIFF	00000000	0.0038147	0	ns	
08	RAM_R_TOF_SUM	023912A0	0.0038147	142268.29	ns	
25	RAM_R_FW_STATUS	04118978	1	68258168		

Figure 17: CPU values in evaluation software

This example screenshot gives on overview of the ScioSense firmware capabilities: Calculation of actual and accumulated flow as well as water temperature and sensor temperature measurement on the one hand (middle panel); operation control and error signal and evaluation on the other hand (error flag panels at each side). This window also demonstrates how results are read out: The lower section contains three rows where the contents of any RAM cell can be read and displayed after multiplication by a suitable factor. For example, under address 0x004 in the first row the current water temperature is stored in some hexadecimal format. The numeric value in $^\circ$C, displayed under “Calculated result 1”, is achieved by multiplication with the suitable factor $1/2^{16}$. Information about location of results, format and suitable factors is given in chapter 0.

Window “Firmware” gives some insight about the code and how parameters are stored:

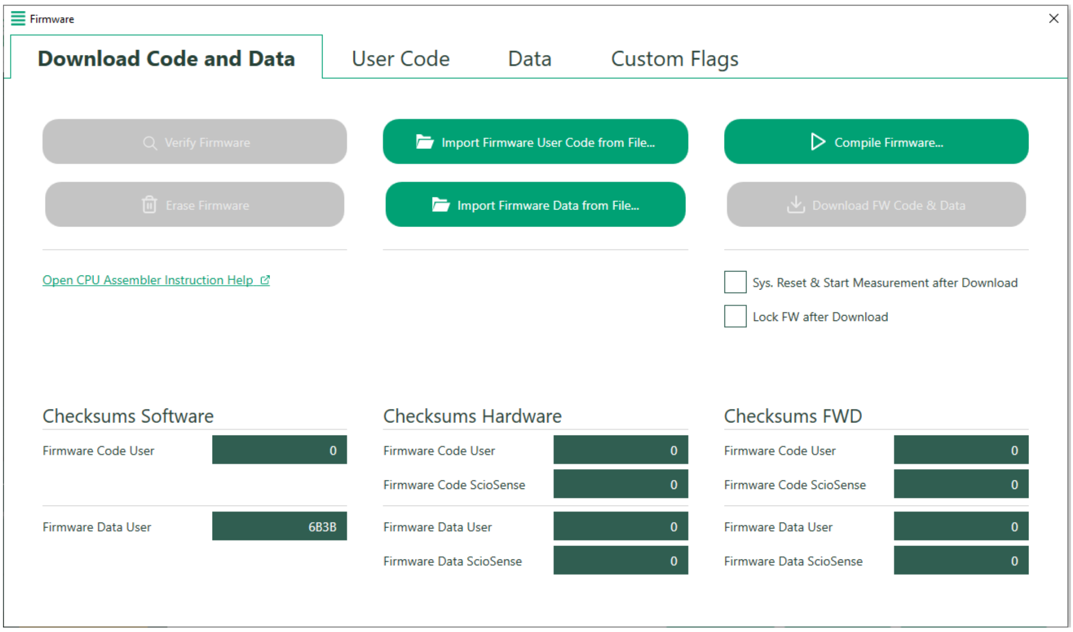


Figure 18: Firmware window

The firmware data are displayed with labels and scaling factors (being stored in the project file).

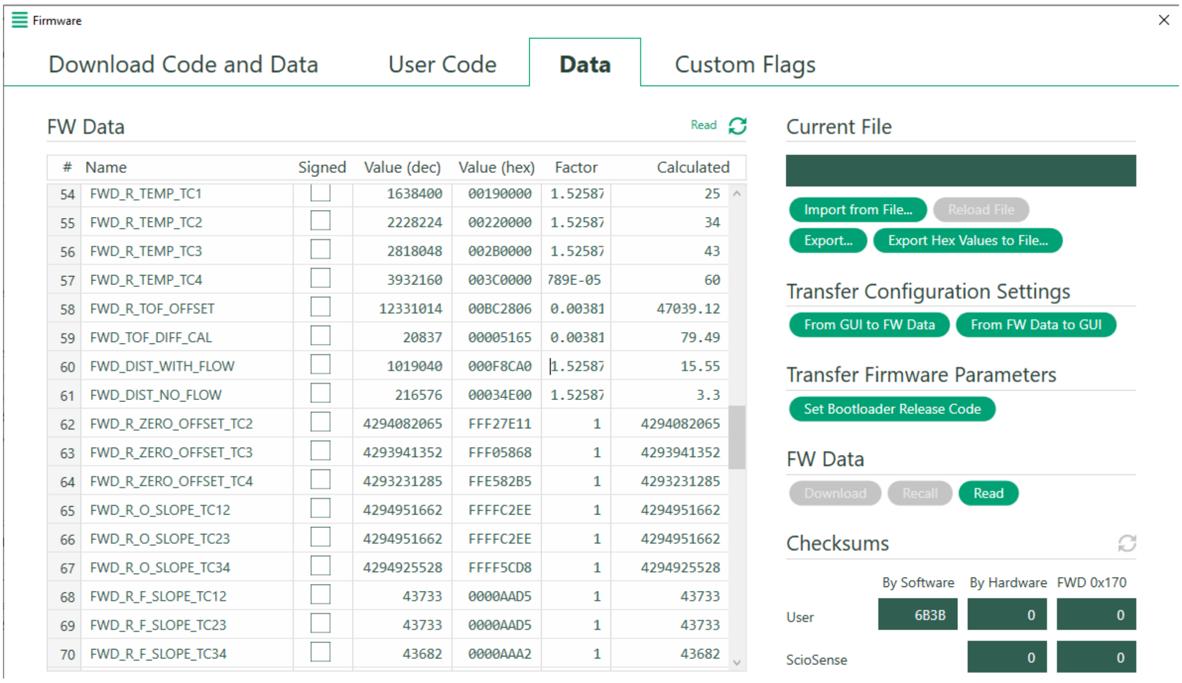


Figure 19: Evaluation software Download window

This window is the interface of the PC software for downloading and checking firmware code and firmware data. Firmware code separates into user code and ScioSense code. The ScioSense code part is pre-programmed at delivery and can't be changed. Both code parts have a version number which is displayed after "Verify FW". For details on version numbers see section **Error! Reference source not found.**. The firmware data hold memory space where permanent configurations, parameters and calibration coefficients are stored. This data can also be opened, displayed and stored to file, and even be modified in this window. In contrast to firmware code, firmware data can also be read from the chip. For details on how to modify this data see section 9.3. The functional meaning of the parameters is discussed throughout chapters 4 to 7, and the overview of all parameters is given in chapter 4.

For both code and firmware data memory, AS6031F1 calculates a checksum, and the PC software compares it to the values stored in firmware data cells 100 and 101 as well as to the displayed data. This permits a check on data integrity. Note that checksums may be updated by the PC software.

9.2 Setup and Customization

When a flow meter system is designed around AS6031F1, it is usual to operate in the beginning remotely in time conversion mode. This is done to define the basic setup and configuration for the measurement operation, like TOF rate, number of hits, first hit level and so on.

This preparation phase leads to a customized chip and measurement configuration, which is adapted to the particular flow meter setup and application. With such a customized configuration at hand, operation of the firmware can be started easily.

When the firmware is purchased by the customer, AS6031F1 chips will be delivered pre-programmed with some standard-configuration. The measurement starts running automatically when the chip is powered on. The chip can be operated with the customized configuration in two ways:

1. Customize configuration temporarily (for quick tests in development): In principle, the customized configuration can be loaded by PC-software or microcontroller as usual. But the running firmware will restore its stored configuration automatically – this will happen at latest after one hour of operation, in some cases even immediately. Thus, to test (temporarily) with a customized configuration, it is safer to switch off firmware operation by disabling post processing. The watchdog must also be disabled to prevent watchdog resets after typically 13 s, which would again restore the stored configuration.
2. Store customized configuration permanently to the chip: If the chip should keep the customized configuration permanently, this configuration has to be stored in the firmware data of the chip. The following section describes how this is done in detail. With the desired configuration stored, AS6031F1 will load this configuration automatically at power-on and after system reset, and the ScioSense firmware will restore it latest every hour.

After first tests, it is in any case necessary to create a firmware data file which contains chip configuration, firmware configuration and individual calibration data for the particular spool piece. The next section presents in detail how customized firmware data files can be created.

Note: that some configuration settings will be modified or enforced by the ScioSense firmware. See sections 0 and 5.13 for details.

Note: GPIO5 and GPIO6 are blocked by the firmware and can't be used.

9.3 Creating and downloading a firmware data file

The evaluation package for the AS6031F1 firmware contains a template firmware data file named AS6031F1_A1C20001.dat (or higher version numbers in later releases). It is initialized with calibration data and configuration for a typical DN20 spool piece with an ultrasonic flow measurement length of

0.06 m. This file should be used as template for a customized firmware data file. The meaning of the file entries is explained throughout this and the next chapter.

Three steps are necessary to customize this file:

1. Store the custom chip configuration in the file (see 9.3.1).
2. Customize the firmware configuration parameters (see 9.3.2).
3. Also store the individual calibration data for the particular flow meter in the firmware data file. In production, each flow meter will have its own firmware data file, containing the individual calibration data of this flow meter (see 9.3.3 and calibration engine user guide SC-001279-UG).

Finally, with all custom parameters written to the firmware data file, this file is downloaded to the particular AS6031 chip of the flow meter. After this last step, configuration and calibration data is permanently stored on the chip and automatically loaded or restored.

The subsequent sections describe the single steps in more detail.

9.3.1 Storing the Chip Configuration in the File

With the description of the parameters in the file at hand (see chapter 7), this step could be done manually, by modifying the memory cells which correspond to configuration registers (cells 108 to 123, cell addresses 0x16C to 0x17B). The AS6031 PC software makes this step easier:

- Set the PC software to the desired configuration. You can easily test the performance by downloading this configuration directly to the chip (switch off post processing and the watchdog to keep the downloaded configuration, else an operating firmware may overwrite it).
- Go to Firmware-> Firmware Download. Stop the measurement. You may open a template firmware data file or read the FW data stored on the chip as template. Now you can click “From GUI to FWD2”. This transfers the current configuration settings of the software into cells 108 to 122. A few things have to be considered:
  - Post processing must be switched on to run the firmware. If you have switched it off for testing the configuration, switch it on again before clicking “From GUI to FWD2”.
  - TOF rate is not stored in the configuration. If the chosen TOF rate is 1, nothing needs to be done. Else, write your chosen TOF rate as hex number into B3 of ***FWD_USM_PRC*** (cell 116). Example: TOF rate is 8=0x8, or TOF rate is 20=0x14; ***FWD_USM_PRC*** is 0x00002824. Change it to 0x08002824, or to 0x14002824.
  - Change cell 108 to a number not equal zero (usually 0x95659C6A – this is an arbitrary test value). This enables the hourly configuration refresh.
- Click “Set bootloader release code”. This sets cell 123 to the value 0xABCD7654. This is the bootloader release code. With that value stored in cell 123, the chip will transfer the configuration stored in cells 108 to 122 to the actual configuration registers of AS6031F1 at power-on and after resets. This needs to be set when working with the ScioSense F01 firmware.
- The first hit levels are also not stored in the configuration registers. Define the first hit levels through the firmware configuration. If you click “1st Hit Level to FWD2”, your current

value will be transferred to cells 93 and 107, but this will not be the final definition. See chapter 7 for details on first hit level definitions.

A firmware data file with the modifications described above can be downloaded to the chip. Then AS6031F1 will automatically move the stored configuration into its configuration registers at startup or at reset, and the firmware will refresh the configuration settings every hour. Download to the chip can be done via remote commands or using the download window of the PC software.

Note that some configuration settings will be modified or enforced by the ScioSense firmware.

9.3.2 Customizing Firmware Configuration Parameters

Major firmware configuration parameters can be manipulated directly in the UFC Evaluation Software. Others need to be changed manually by changing entries in the firmware file. The meanings of the individual parameters are explained in detail in the subsequent sections – see chapter 5.1 for a complete overview of parameters. Firmware operation configuration is mainly done in register ***FWD_FW_CONFIG*** (cell 106, address 0x16A).

A convenient way to create a customized firmware data file is to use the template firmware data file AS6031F1_A1C20001.dat (or higher version later) and to modify the firmware parameters. It is recommended to check all parameters described in chapter 5; usually many settings can remain unchanged, but others like e.g. the low amplitude limit (***FWD_R_AM_MIN***, cell 85, address 0x155) should be adapted to the particular spool piece.

With parameters and configuration register ***FWD_FW_CONFIG*** checked and, where desired, adapted, the customized firmware data file should be stored and can be downloaded to the chip. In combination with an updated chip configuration (see preceding section), this file will set the chip into the desired measurement and firmware operation. The last missing adaption for a fully customized firmware data file is then a suitable calibration, as discussed in the next section.

9.3.3 Creating and Storing Individual Calibration Data (Overview)

Calibration data for operation with ScioSense firmware can only be generated through the calibration engine, as described in detail in user guide SC-001279-UG Calibration Engine (former UG403 or Volume 5). The calibration engine uses an initial firmware data file, ideally a customized file as described in the two preceding sections. For the next actions, development phase and production phase must be distinguished:

- In development phase, a sufficiently high number of measurements is needed to characterize the flow meter design and to generate a master calibration. This should be done with the identical configuration as defined in section 9.3.1.
- The calibration engine stores this master calibration into the initial firmware data file. This is the first (nearly) complete firmware data file to totally configure the system as a calibrated flow meter.
- The following points are the only actions that need to be done in production phase:
- Do calibration measurements for each particular flow meter. This will be typically one zero flow measurement and one high flow measurement.
- Scale the master calibration from production phase, using these measurements, according to the scaling rules in Manual SC-001279-UG Calibration Engine (former UG403 or Volume 5).

This requires as input the customized firmware data file, with the master calibration for this flow meter system stored.

- Store the modified calibration values in a copy of the firmware data file. This file is then the individual calibration file for this particular flow meter. So this process creates one file for each flow meter which is being calibrated.
- Finally, the checksums should be calculated and stored to the files (cells 124 to 127, addresses 0x17C to 0x17F, see section 5.7). Storing the right checksums permits regular integrity checks of the stored data.
- The individual firmware data files created in this process should then be downloaded to the corresponding flow meter chips. This is the final step to get a calibrated flow meter with AS6031F1 in operation.

The five actions in production phase listed above may of course be done in one process by an external controller, without even storing a separate individual firmware data file.

9.4 Calibration process in development and production

The customer has to do the calibration and store calibration and configuration data in the firmware data memory. ScioSense supports the procedure of generating this data through its calibration engine. The procedure is sketched in the following flow charts:

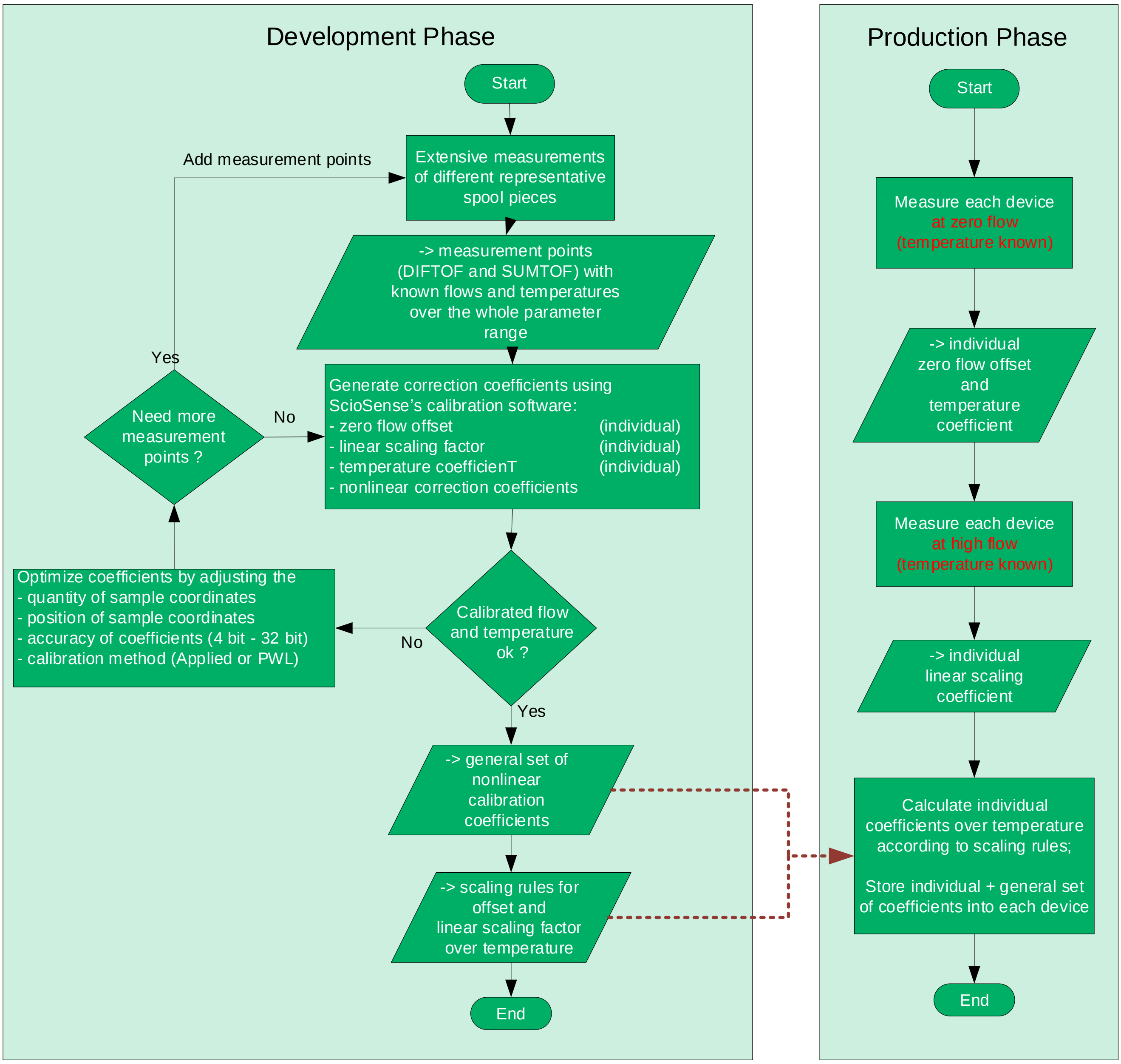


Figure 20: Calibration process

In practice, this means that the customer has to do a thorough calibration, actually a characterization, in development phase on some representative devices. These initial calibrations require a number of measurements at flows and temperatures, sufficiently large to characterize the devices and covering the complete range of application. The calibration coefficients gained by this preparation (the master calibration) are adapted in mass production phase using dedicated scaling rules and based on only two measurements at zero flow and high flow and at an arbitrary (but known) temperature.

The accuracy of the applied calibration depends on the particular spool piece and its production stability. To achieve good results with this (or any other) calibration, the following should be considered:

- The production of the spool piece must be sufficiently repeatable:
  - No measurement deviations due to device tolerances beyond accuracy limits,
  - Linear scaling over temperature is sufficiently comparable among devices,
  - Non-linear behavior is sufficiently comparable among devices.
- Tolerances and parameters change due to aging effects must be limited as well.
- Influence of uncontrolled parameters must be insignificant (e.g. housing and environment temperature).

If above criteria are not fulfilled, more individual calibration in series production and/or additional measurements of uncontrolled parameters are needed.

Note that such enhanced methods are currently not implemented in ScioSense firmware. Enhanced calibration methods thus require a dedicated firmware, typically developed by the customer himself.

Please also consider possible influences of the water quality on the measurement results.

ScioSense firmware supports a complete linear calibration and offset correction, and two different calibration schemes for nonlinear calibration. One is based on the well-known piecewise linear (PWL), the other incorporates a proprietary method which needs less coefficients and is inherently smooth. It depends on the actual application which one is preferable, customers who order ScioSense firmware can select any of them.

The calibration engine supports the linear calibration as well as both nonlinear methods. It provides automatic calibration coefficients generation and optimization, and it is able to store the results directly into firmware data. An overview description of linear and nonlinear calibration parameters, of their determination and optimization, and of their usage in mass production, can be found in section 3.

10 Adding Custom Code to ScioSense FW

A customer may add his own code to the one provided by ScioSense. Actually, the AS6031F1 chips will be delivered pre-programmed with a part of the code memory that is read- and write-protected (FWA), and with a small main routine as an open-source interface (FWU).

The structure of this default open part firmware is very simple, with only minimized code in the user part (line 37):

```
1   Org 0
2   nop
3
4   org 4
5   MK_CPU_REQ_INT:
6         ramadr     SHR_CPU_REQ
7         skipBitC   r,BNR_SPH,2
8          skipBitC  r,BNR_GPH,1
9           Skip 3        ; both bits are set: run special actions for storing on-the-fly
10          ;...else check where to go          ;skip range [
11        ramadr     FWD_JUMP_FLAG
12        gotoBitC   r,0,MK_USER_FW  ; Bit 0 clear: go directly to user FW
13        goto  MK_ACAM_FW           ;...else go directly to F01 start point
14        clear      r
15        ramadr          SHR_RC
16        skipBitC   r,1,2 ; if bit is set...1 is the upper bit of the double bit SR_CFG_DONE
17          jsub          ROM_CSM_FWDU               ;...run FWDU checksum
18          stop                                     ;...and stop
19        jsub       ROM_BLD_CFG                     ;...else run config boot loader
20        org 31                                ; put stop into last byte of upper protected code block
21        stop
22
23  org 32                             ;
24        equal1     FW_ROMVERSION_REV     ;
25        equal FW_VERSION           ; Defined at the beginning of this file
26
27  org 36
28  PH_S_DETECT:
29  #ifdef PH_S_LIBRARY
30        jsub  MK_PH_S_ROUTINES      ; jump to call every phase-shift routine
31        ;jsub MK_PH_S_MOVE_RESULTS  ; used only for debugging. Not necessary to include.
32        jsubret
33  #else
34        jsubret
35  #endif
36
37  MK_USER_FW:
38        ; *** Enter your code here ***********
39        gotoBitS   x,1,MK_TEST_FIRMWARE
40        skipNE     1
41          jsubret
42        stop
```

The customer has the free choice to add any code parts, for example enhanced calibration or error handling, data storage or different communication setups. But of course it is limited by the free space available. Four different indirect jumps from the applied firmware to the user firmware are implemented. The open code therefore needs jsubret commands to jump back to the applied code. See 2.2

The interfaces to ScioSense firmware are given on the one hand by the results in RAM cells, on the other hand by freely available subroutines.

The following chip resources are available for user code:

- 602 bytes are available for the user
  - ScioSense applied firmware starts at address 1216 (version A1A1C314)
  - Phase shift library needs 546 bytes in advance to FWA.
  - User code starts at address 68, so that.
- RAM usage (of 176 x 32 words):
  - ~ 21 words free / unused
  - ~ 19 words available for temporary storage
- Firmware data usage (of 119 x 32 words):
  - 11 words configuration (always)
  - 7 words always free, additional 27 words if PWL is not used

The sample code defines some constants definitions for firmware revision placement: FW_VERSION_NUM, FW_VERSION_MAJ, FW_VERSION_MIN, FW_VERSION_BLD. Together with FW_ROMVERSION_REV (=A1; leave unchanged) they make up the version number. It is good practice to keep track of firmware versions by changing the version numbers accordingly. If the values of these constants are changed, and the version statement at the end of the code remains as it is, the version number will also appear in the register variable **SRR_FWU_REV**, and will be displayed in the download window of the PC software. For version number definition see section 11.2.

10.1 Editing and Compiling

The easiest way to edit the assembler file is by means of a standard text editor like Notepad++ that offers comfortable editor functionality and text coloring and high lighting. An appropriate language file for a nice text coloring can be provided.

The opcodes are described in the AS6031 datasheet, but they can be shown in the evaluation software, too. The link is given in the Firmware window on the “Download Code and Data” tab.

```
; Included libraries
#include "Header/AS6031_Flow_FW_A1A1.h"                  ; Header file containing memory address and constant
#include "Header/AS6031_AS6040_REG_A1.h"                  ; Definition of Register addresses, bit positions an
#include "Header/AS6031_AS6040_ROM_A1.h"          ; Header file containing ROM routine start addresses

CONST FW_ROMVERSION_REV          0xA1             ; The user can create here his own revision number. It is st
CONST FW_VERSION_NUM             0xC10000         ;
CONST FW_VERSION_MAJ             0x000000         ;
CONST FW_VERSION_MIN             0x000000         ;
CONST FW_VERSION_BLD             0x000003         ;
CONST FW_VERSION                 FW_VERSION_NUM + FW_VERSION_MAJ + FW_VERSION_MIN + FW_VERSION_BLD  ;

CONST CORR_TRIM2                 0x401100C4

;##############################################################################
;####### Start of user code with firmware code user (FWCU) revision number
;##############################################################################
; Important: Leave the following 5 lines unchanged !
    org 0                       ; only as information for ScioSense compiler
    nop                         ; only as information for ScioSense compiler

#include "Libraries/UFC_PHASE_SHIFT.lib"

    org 4
    ramadr  FWD_JUMP_FLAG
    gotoBitC    r,0,MK_USER_FW  ; Bit 0 clear: go directly to user FW and from there to STOP
        goto    MK_APPL_FW          ;...else go directly to F01 start point

    clear       r
    ramadr      SHR_RC
    skipBitC    r,1,2               ; if bit is set...1 is the upper bit of the double bit SR_CFG_DONE
      jsub      ROM_CSM_FWDU        ;...run FWDU checksum
      stop                          ;...and stop
    jsub        ROM_BLD_CFG         ;...else run config boot loader

    ramadr      CR_TRIM2            ; temporary, correction of TRIM 2 value, critical at this point due
    move        r,CORR_TRIM2        ; to be removed with corrected F1 chips

    org 31
    stop
```

Figure 21: Notepad++ for editing

The compiler is part of the UFC Evaluation Software. It is integrated in the Firmware window.

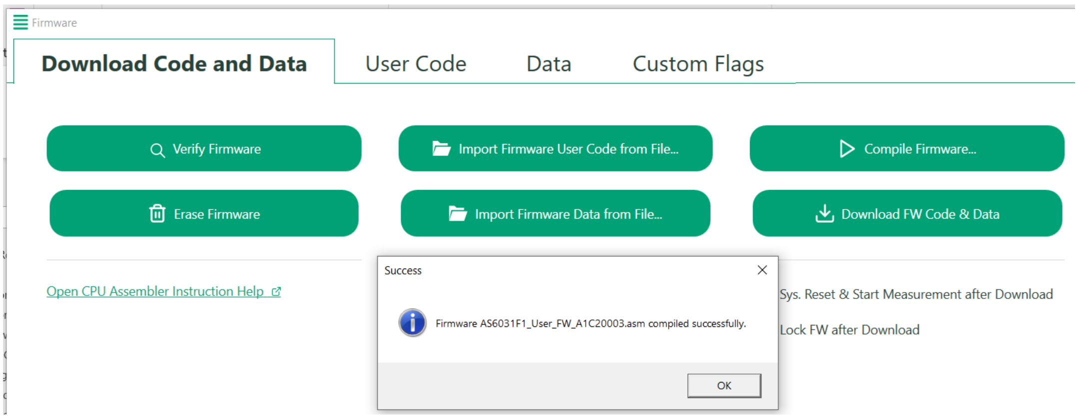


Figure 22: Compiler

When you click Compile then you are asked to open the assembler source file. Compilation is then automatically started. Success or error are shown in a pop-up window and download is possible. The .hex file wile will be stored. This is the one that will be uploaded to the chip.

Note: Before you press Download make sure that the firmware data file is also loaded. The software downloads both together! Under User Code the version can be checked.

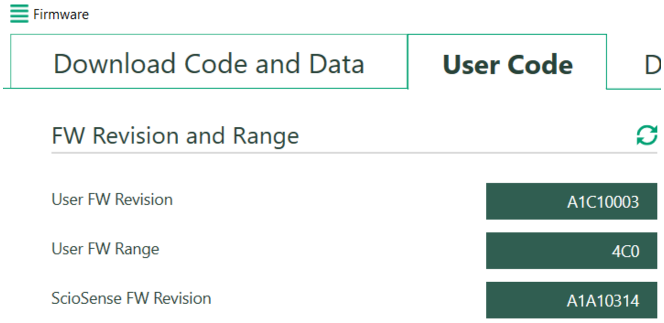


Figure 23: Checksums and version

10.2 Sample Code

The following sample code, file AS6031F1_User_FW_A1C20003.asm, demonstrates the basic structure of user code. The original file with much more comments can be downloaded from our website.

org 4 Here is the short main routine that typically jumps into the ScioSense applied code

org 31 Here is already the mandatory stop of the man routine

org 32 Here you find the firmware revision

org 36 Here is the base routine for the phase jump detection and correction

org 68 Here starts the user code with mainly the on the check for the indirect jump trigger and the according subroutines. A major role play FWD register 2. The lowest bit 0 specifies whether only user code or also the ScioSense applied code is processed. Bits 1 to 4 then specify which of the four indirect jump options are implemented.

```
; Included libraries
#include "Header/AS6031_Flow_FW_A1A1.h"                    ; Header file containing memory address and constants c
#include "Header/AS6031_AS6040_REG_A1.h"                   ; Definition of Register addresses, bit positions and r
#include "Header/AS6031_AS6040_ROM_A1.h"           ; Header file containing ROM routine start addresses

CONST  FW_ROMVERSION_REV          0xA1            ; The user can create here his own revision number. It is store
CONST  FW_VERSION_NUM             0xC10000        ;
CONST  FW_VERSION_MAJ             0x000000        ;
CONST  FW_VERSION_MIN             0x000000        ;
CONST  FW_VERSION_BLD             0x000003        ;
CONST  FW_VERSION                 FW_VERSION_NUM + FW_VERSION_MAJ + FW_VERSION_MIN + FW_VERSION_BLD  ;

CONST  CORR_TRIM2                 0x401100C4

; Important: Leave the following 5 lines unchanged !
    org 0                        ; only as information for ScioSense compiler
    nop                          ; only as information for ScioSense compiler

#include "Libraries/UFC_PHASE_SHIFT.lib"
```

```
org 4
ramadr  FWD_JUMP_FLAG
gotoBitC     r,0,MK_USER_FW   ; Bit 0 clear: go directly to user FW and from there to STOP
    goto     MK_APPL_FW         ;...else go directly to F01 start point

clear        r
ramadr       SHR_RC
skipBitC     r,1,2              ; if bit is set...1 is the upper bit of the double bit SR_CFG_DONE
  jsub       ROM_CSM_FWDU       ;...run FWDU checksum
  stop                          ;...and stop
jsub         ROM_BLD_CFG        ;...else run config boot loader

ramadr       CR_TRIM2           ; temporary, correction of TRIM 2 value, critical at this point due to
move         r,CORR_TRIM2       ; to be removed with corrected F1 chips

org 31
stop

org 32                 ; Address for User Code Version number
equal1  FW_ROMVERSION_REV  ;
equal   FW_VERSION     ; Defined at the beginning of this file

org 36
; ************************ Phase Shift Detection and Correction ************************
PH_S_DETECT:
#ifdef PH_S_LIBRARY
    jsub     MK_PH_S_ROUTINES        ; jump to call every phase-shift routine
    jsub     MK_PH_S_MOVE_RESULTS
    ;ramadr 0x39
    ;add     r, 0x01                 ; incrementing the counter
    jsubret
#else
    jsubret
#endif

org 68
; ************************* USER CODE ****************************************************
MK_USER_FW:
;test
ramadr       CR_TRIM2
move         r,CORR_TRIM2

ramadr SHR_GPO                  ; SHR_GPO Register 0xD3, toggle GP0 for test purposes
bitset r, BNR_GPO0_OUT          ; Set BNR_GPOx_OUT

; Check for jump condition
ramadr       FWD_JUMP_FLAG
gotoBitC     r,0,MK_ONLY_USER_FW  ; Bit 0 clear: go directly to user FW

skipBitC     x,1,1              ; x holds bit number of indirect jump. if bit 1 is set jump to conditic
    jsub     MK_JUMP_COND_1     ; indirect jump at the very beginning
;   jsubret
skipBitC     x,2,1              ; if bit 2 is set jump (before PP) to condition 2 subroutine
    jsub     MK_JUMP_COND_2     ; indirect jump after error handling
;   jsubret
skipBitC     x,3,1              ; if bit 3 is set jump (after PP) to condition 3 subroutine
    jsub     MK_JUMP_COND_3     ; indirect jump after post processing
;   jsubret
skipBitC     x,4,1              ; if bit 4 is set jump (GPH) to condition 4 subroutine
    jsub     MK_JUMP_COND_4     ; indirect jump at the end, general purpose handling (GPH) replacement
;   jsubret

bitclr r, BNR_GPO0_OUT          ; clear BNR_GPOx_OUT
jsubret
goto         MK_STOP
```

```
MK_JUMP_COND_1:
    ; ************
    ; Enter code  for the case of FWD bit 1 = 1, at the veryy beginning, before MK_CHK_AND_RESUME_INT
    ; ************

    ramadr SHR_GPO                  ; SHR_GPO Register 0xD3, toggle GP1 for test purposes
    bitset r, BNR_GPO1_OUT          ; Set BNR_GPOx_OUT
    nop
    nop
    bitclr r, BNR_GPO1_OUT          ; clear BNR_GPOx_OUT

    jsubret
```

…

```
MK_ONLY_USER_FW:
    ; ************
    ; Enter code  for the case of FWD bit 0 = 0
    ; ************
    ramadr SHR_GPO                  ; SHR_GPO Register 0xD3     Additional pulse in case of User FW only for te
    bitclr r, BNR_GPO0_OUT          ; Clear BNR_GPOx_OUT        Propoer function confirmed
    ramadr SHR_GPO                  ; SHR_GPO Register 0xD3
    bitset r, BNR_GPO0_OUT          ; Set BNR_GPOx_OUT

    goto    MK_STOP

MK_STOP:
    ; place additional code if needed

    ; Common Firmware Interrupt
    ramadr SHR_EXC                  ; System Handling Register Executables 0x0DD
    bitset r, BNR_FW_IRQ_S          ; Set Flag IRQ_EN_FW_S

    ; All CPU Requests are automatically cleared when CPU stops
    ; NO need to clear manually
    ;ramadr SHR_CPU_REQ             ; CPU Requests 0x0DC
    ;clear  r                       ; clear all CPU requests

    ramadr SHR_GPO                  ; SHR_GPO Register 0xD3, set back GPO0 for test purposes
    bitclr r, BNR_GPO0_OUT          ; Clear BNR_GPOx_OUT

    clrwdt                          ; Clearing watchdog

stop                            ; stop CPU
```

All CPU requests are automatically cleared when the CPU stops, and no further code is needed for that.

11 Appendix

11.1 Firmware Input Data Overview (FWD)

Table 20: Firmware input data overview - for details see 4

Cell	Variable Name	Description	
0	*FWD_FWU_CS*	Checksums	User code
1	*FWD_FWDU_CS*		User data
2	*FWD_JUMP_FLAG*	Flag to trigger jump from ScioSense code to user code	
3	*FWD_ERROR_COUNT_CONF1*	Error counter 1	
4	*FWD_ERROR_COUNT_CONF2*	Error counter 2	
5	*FWD_ERROR_COUNT_21*	Temporary storage of error counts 2 and 1	
6	*FWD_ERROR_COUNT_43*	Temporary storage of error counts 4 and 3	
7	*FWD_ERROR_COUNT_INV21*	Inverse of 0x105 for constant checksum	
8	*FWD_ERROR_COUNT_INV43*	Inverse of 0x106 for constant checksum	
9 to 15	*NOT USED*	Not used	
16 to 41	*PWL COEFFICIENT TABLE*	Typical PWL calibration coefficients table	
42 to 53	*SCIOSENSE COEFFICIENT TABLE*	ScioSense calibration coefficients table	
54	*FWD_R_TEMP_TC1*	Temperatures for linear calibration in °C Format fd16	1st
55	*FWD_R_TEMP_TC2*		2nd
56	*FWD_R_TEMP_TC3*		3rd
57	*FWD_R_TEMP_TC4*		4th
58	*FWD_R_TOF_OFFSET*	Offset time for SUMTOF in raw TDC units	
59	*FWD_TOF_DIFF_CAL*	DIFTOF at high flow calibration in raw TDC units	
60	*FWD_DIST_WITH_FLOW*	Ultrasonic sound path length along flow in m	
61	*FWD_DIST_NO_FLOW*	Ultrasonic sound path length w/o flow in m	
62	*FWD_R_ZERO_OFFSET_TC2*	Zero flow DIFTOF O(TC)	at TC2
63	*FWD_R_ZERO_OFFSET_TC3*		at TC3

64	*FWD_R_ZERO_OFFSET_TC4*		at TC4
65	*FWD_R_O_SLOPE_TC12*	Zero flow slope S_O(TC) Format fd16	between TC1 and TC2
66	*FWD_R_O_SLOPE_TC23*		between TC2 and TC3
67	*FWD_R_O_SLOPE_TC34*		between TC3 and TC4
68	*FWD_R_F_SLOPE_TC12*	Proportionality factor slope S_F(TC) Format fd16	between TC1 and TC2
69	*FWD_R_F_SLOPE_TC23*		between TC2 and TC3
70	*FWD_R_F_SLOPE_TC34*		between TC3 and TC4
71	*FWD_R_F_OFFSET_TC2*	Proportionality factor F(TC) Format fd16	at TC2
72	*FWD_R_F_OFFSET_TC3*		at TC3
73	*FWD_R_F_OFFSET_TC4*		at TC4
74	*FWD_SOUND_VEL_MAX*	Maximum of speed of sound in m/s	
75	*FWD_1_BY_A*	Medium constant	
76	*FWD_CONST_C*	Medium constant	
77	*FWD_THETA_MAX*	Temperature at maximum speed of sound in °C	
78	*FWD_LONG_TERM_ERROR*	# of low AM measurements before failure	
79	*FWD_FHL_USER*	trusted FHL ratio / absolute trusted FHL	
80	*FWD_TOF_SUM_DELTA*	FHL method 3: SUMTOF operating - trusted FHL	
81	*FWD_TOFSUM_VAR_LIM*	Error limit for deviation of SUMTOF	
82	*FWD_HSC_DEV*	Error limit for HSC calibration	
83	*FWD_ERR_INTERRUPT*	Error flag positions that issue an interrupt	
84	*FWD_AM_DIFF_LIM*	Error limit for amplitude UP - DOWN in mV	
85	*FWD_R_AM_MIN*	Minimum allowed amplitude in mV	
86	*FWD_PW_NOM*	Firm Nominal value of PWR	
87	*FWD_PW_DEV*	Error limit PW UP - DOWN	
88	*FWD_CORR_OFFSET*	Offset correction in case of negative PWL coefficients	
89	*FWD_TOF_RATE_FACTOR*	Factor for TOF rate scaling in zero flow case	
90	*FWD_FLOW_AVG_FACTOR*	2^N number of flow values for averaging	

91	*FWD_R_PULSE_PER_LITER*	Pulse interface: Number of pulses per liter
92	*FWD_R_PULSE_MAX_FLOW*	Pulse interface & maxflow error limit
93	*FWD_NEG_FLOW_LIMIT*	Cut-off limit for negative flow in l/h
94	*FWD_R_TOF_DIFF_LIMIT*	Minimum limit for DIFTOF in raw TDC units
95	*FWD_ZERO_FLOW_LIMIT*	Zero flow limit in l/h
96	*FWD_CAL_PTR_OFFSETR*	Reference branch offset resistance
97	*FWD_EXT_REF_VAL*	Value of reference resistor R_{ref} in Ω
98	*FWD_PT_INT_SLOPE*	Internal sensor resistance slope in (K/Ω)*Rref
99	*FWD_PT_INT_NOM*	Internal sensor nominal resistance
100	*FWD_PTC_RATIO_INV*	Nominal ratio of reference resistor to PT cold sensor resistance at 0°C
101	*FWD_PTH_RATIO_INV*	Nominal ratio of reference resistor to PT hot sensor resistance at 0°C
102	*FWD_HSC_CLOCK*	HSC clock in Hz, used as reference in the clock calibration
103	*FWD_MH_RLS_DLY*	Nominal value for release delay
104	*FWD_PH_S_THRESHOLD*	Absolute value of phase-shift threshold value.
105	*FWD_MH_RLS_DLY*	Multihit release delay limits
106	*FWD_FW_CONFIG*	ScioSense firmware configuration register
107	*FWD_FW_RLS*	Boot loader release. 0xABCD7654 activates the bootloading process after startup
108	*FWD_R_CD*	Watchdog disable code
109	*FWD_IFC_CTRL*	Configuration data for **CR_IFC_CTRL**
110	*FWD_GP_CTRL*	Configuration data for **CR_GP_CTRL**
111	*FWD_USM_OPT*	Configuration data for **CR_USM_OPT**
112	*FWD_IEH*	Configuration data for **CR_IEH**
113	*FWD_CPM*	Configuration data for **CR_CPM**
114	*FWD_MRG_TS*	Configuration data for **CR_MRG_TS**

115	*FWD_TPM*	Configuration data for **CR_TM**	
116	*FWD_USM_PRC*	Configuration data for **CR_USM_PRC**	
117	*FWD_USM_FRC*	Configuration data for **CR_USM_FRC**	
118	*FWD_USM_TOF*	Configuration data for **CR_USM_TOF**	
119	*FWD_USM_AM*	Configuration data for **CR_USM_AM**	
120	*FWD_TRIM1*	**CR_TRIM1**; set to 0x84A0C47C	
121	*FWD_TRIM2*	**CR_TRIM2**; Set to 0x401700CF	
122	*FWD_TRIM3*	**CR_TRIM3**; Set to 0x00270808	
123	*NOT USED*	-	
124	*FWD_R_FWD1_CS*	Firmware data 1	Checksums
125	*FWD_R_FWD2_CS*	Firmware data 2	
126	*FWD_R_FWU_CS*	Firmware code user	
127	*FWD_R_FWA_CS*	Firmware code ScioSense	

11.2 Firmware Version Numbers

AS6031F1 stores two 4-byte firmware version numbers, one for the (changeable) customer firmware code and one for the (fixed at delivery) ScioSense firmware code. The two numbers are available after the chip's bootloader has stored them into registers **SRR_FWU_REV** (0x0ED) and **SRR_FWA_REV** (0x0EE), typically when the chip has bootloader release code set after power-on. They can also be read in the download window of the PC software after verification.

The source code of the open part is available to customers and may be modified. When the open firmware part is modified, its version number should also be changed to indicate the modification.

Table 21: Firmware numbering

Byte	B3	B2		B1	B0
Description	Ax: ROM version	Ax Cx Ex Fx	ScioSense applied FW (protected) Open custom firmware part Empty firmware (protected) ScioSense examples x = main version	xx Version number Major/minor	xx
Examples					
A1E10004	A1	E1	"Empty" firmware (protected)	Version 0.0	Build 4
A1A10314	A1	A1	ScioSense flow firmware (protected)	Version 0.3	Build 14
A1F11101	A1	F1	ScioSense example 1, DIFTOF_over_PI	Version 1.1	Build 1

11.3 Firmware Related Files

The following list contains available files types related to ScioSense firmware:

Table 22: Firmware A1.A1.01.xx Related Files

File Name	File Type	Desription
AS6031F1_A1C20001.dat	Firmware data	This file is the major template for firmware data, containing configurations and calibrations. In production, each individual flow meter has its own firmware data.
AS6031F1_A1C20001_prj.ufc	UFC	Project file for UFC Evaluation Software, including firmware data, CPU, configuration and flags information
AS6031F1_User_FW_A1C20001.asm	Assembler code	Open part of the ScioSense firmware assembler code. This file is needed for restoring the original delivery state after modifications. It should be used as template for customizations.

AS6031F1_User_FW_A1C20001.hex	Hex	This file is generated by the UFC Evaluation Software compiler. It contains downloadable HEX code and corresponding assembler commands as comment.
AS6031F1_User_FW_A1C20001_sim.hex	Hex	This file is generated by the UFC Evaluation Software compiler. It contains only HEX code and can used for download in production.
AS6031F1_User_FW_A1C20001.obj	Object	Firmware label and address list
AS6031F1_User_FW_A1C20001.dbg	Debug	HEX to assembler file line reference
AS6031_Flow_FW_A1A1.h	Header	This general header file contains all variable definitions related to the applied flow firmware. It can also be used as quick reference.
AS6031_AS6040_User_FW.h	Header	Optional header file template for user firmware
AS6031_AS6040_REG_A1.h	Header	General definitions for AS6031 and AS6040 hardware registers
AS6031_AS6040_ROM_A1.h	Header	Label definition for commonly usable ROM routines
UFC_PHASE_SHIFT.lib	Library	Phase shift library, to be added to the user firmware

11.4 Notational Conventions

Throughout the AS6031F1 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 32-bit word are denoted B0, B1, B2 and B3 where B0 is the lowest and B3 the highest byte.
- Negative binary or hexadecimal numbers are given as two's complement. The two's complement of an N-bit number is defined as its complement with respect to 2N.
- For instance, for the three-bit number 010, the two's complement is 110, because 2N = 1000 and 1000 - 010 = 110.
- Abbreviations and expressions which have a special or uncommon meaning within the context of AS6031F1 application are listed and shortly explained in the list of abbreviations, see **Error! Reference source not found.**. 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 *bold italic*. 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).

11.5 Abbreviations

Table 23: Abbreviations

Short	Description
AM	Amplitude measurement
CD	Configuration Data
CPU	Central Processing Unit
CR	Configuration Register
CRC	Cyclic Redundancy Check
DIFTOF, DIFTOF_ALL	Difference of up and down → TOF
DR	Debug Register
FEP	Frontend Processing
FDB	Frontend data buffer
FHL	First hit level (physical value V_{FHL})
FW	Firmware, software stored on the chip
FWC	Firmware Code
FWD	Firmware Data
FWD-RAM	Firmware Data memory
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	System 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
UART	Universal Asynchronous Receiver & Transmitter
USM	Ultrasonic measurement
V_{ref}	Reference voltage
X, Y, Z	Internal registers of the CPU
ZCD	Zero cross detection, physical level V_{ZCD}

11.6 Glossary

Table 24: Glossary

Term	Meaning	AS6031F1 Interpretation
AM	Amplitude measurement	This is a peak measurement of the received signal amplitude. It can be configured to be executed in different time frames, which allows to pick the overall signal maximum (to control the signal level), or to measure only the peak of a selected number of → wave periods. The latter allows for a more detailed receive signal analysis.
Backup	Permanent storage of a data copy	AS6031F1 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
Calibration	Parameter adjustment to compensate variations	In AS6031F1, 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) 32-bit 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
CPU	Central Processing Unit	32-bit 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, VFHL	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	Program code (in a file) for chip operation	The program code can be provided by ScioSense or by the customer, or a combination of both. The complete program code becomes the → FWC (firmware code) when stored in the → NVRAM. The term firmware is in general used for all firmware programs, no matter if they make up the complete FWC or not.
Flow meter mode	Operation mode of AS6031F1 as full flow meter system	In flow meter mode, the AS6031F1 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 AS6031F1 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 means the program in the file. On the other hand, a particular firmware 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.
FWD	Firmware Data	The firmware configuration and calibration data, to be stored in the → FWD-RAM
FWD-RAM	Firmware Data memory	120 x 32 bit words of → NVRAM (built as volatile → SRAM and non-volatile flash memory). The FWD-RAM is organized in two address ranges, FWD1 (→ RAM addresses 0x100 - 0x11F) and FWD2 (RAM addresses 0x120 - 0x17F). Main purpose is calibration and configuration. Due to its structure, FWD-RAM can be used like usual → RAM by the firmware. But note that with every data recall from flash memory the contents of the SRAM cells get overwritten.
GPIO	General purpose input/output	AS6031F1 has up to 6 GPIO pins which can be configured by the user. Some of them can be configured as → PI or → I2C-interface.
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 first 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. - Stop hit is the 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 AS6031F1. 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 AS6031F1 terminology, INIT processes do not 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 AS6031F1. 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 AS6031F1 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	AS6031F1 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 32 bit 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

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. If any kind of firmware code execution is requested, the program counter is continued at 0x0000 (for FW initialization, post processing or general purpose handling).
PWR	Pulse width ratio	Width of the pulse following 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 and → DR).
RAM	Random Access Memory	176 x 32 bit 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	AS6031F1 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 → firmware (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	System Handling Register	Registers that directly control chip operation. The data & flags of system handling registers have a dynamic character. They are typically updated by post processing, but have to be initially configured before measurement starts.
SPI	Serial Peripheral Interface	Standard interface for communication of the AS6031F1 with an external master controller
SRAM	Static RAM	AS6031F1 does not use any dynamic RAM, in fact all RAM in AS6031F1 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.
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 AS6031F1 that controls voltage and timing	The supervisor of AS6031F1 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 AS6031F1 are → TOF measurement, temperature measurement (→ TM), post processing (→ PP), remote communication and voltage measurement.
Time conversion mode	Remotely controlled operation of AS6031F1	In time conversion mode, the AS6031F1 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 AS6031F1. 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). AS6031F1 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). AS6031F1 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.
Vref	Reference voltage	The analog interface of AS6031F1 refers to Vref, 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 Vref, negative swings of the receive signal are avoided.
VZCD	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 AS6031F1 → resets the chip (including → CR refresh) if no watchdog clear (→ firmware command clrwdt) within 13.2s (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.
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 AS6031F1 detects → hits in the received → burst signal by comparing the received signal level to a given reference voltage (see also → FHL, → ZCD and → hit).

12 Copyrights & Disclaimer

Copyright Sciosense B.V. High Tech Campus 10, 5656 AE Eindhoven, The Netherlands. Trademarks Registered. All rights reserved. The material herein may not be reproduced, adapted, merged, translated, stored, or used without the prior written consent of the copyright owner.

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.

13 Revision information

Table 25: Revision history

Revision	Date	Comment	Page
1	13 Mar 2022	First release	All
2	2 Dec 2022	Update Hardware/No Water Error Update TRIM1, TRIM2, Trim 3 Default values	19,21-23,29-32,37,38,43,48-52,61,69

Note(s) and/or Footnote(s):

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

Address: Sciosense B.V.
High Tech Campus 10
5656 AE Eindhoven
The Netherlands

Contact: www.sciosense.com
info@sciosense.com