



# AS6031/40

## How to Write Custom Firmware

**AS6031/40 application note**

Revision: 1

Release Date: 2021-11-26

Document Status: Production

## Content Guide

<b>Content Guide</b> .....	<b>2</b>
<b>1 Introduction</b> .....	<b>3</b>
<b>2 Preparation</b> .....	<b>4</b>
2.1 Project Files .....	4
2.2 Open the .asm Example .....	4
2.3 Assembler File Description .....	5
<b>3 Assembler Programming</b> .....	<b>6</b>
3.1 Declaration .....	6
3.2 Initialization of AS6031/40 .....	7
3.3 Jump to Subroutine .....	8
3.4 Compile .....	9
3.5 Download to the Target .....	9
<b>4 Summary / Result</b> .....	<b>12</b>
4.1 Verify Code Executing Properly .....	12
<b>5 Copyrights &amp; Disclaimer</b> .....	<b>14</b>
<b>6 Revision information</b> .....	<b>14</b>

# 1 Introduction

AS6031/40 is a system-on-chip solution for ultrasonic flow metering. Using its integrated CPU and code memory, AS6031/40 can be operated with a dedicated firmware for evaluation of results and operational control.

This application note describes how to write a customized firmware, using AS6031/40 without flow meter firmware.

Following the naming convention, the modified file should be saved with a different name such as A1.C1.00.YY, where C indicates it is a custom code.

Figure 1 shows the basic flow diagram of the main program.

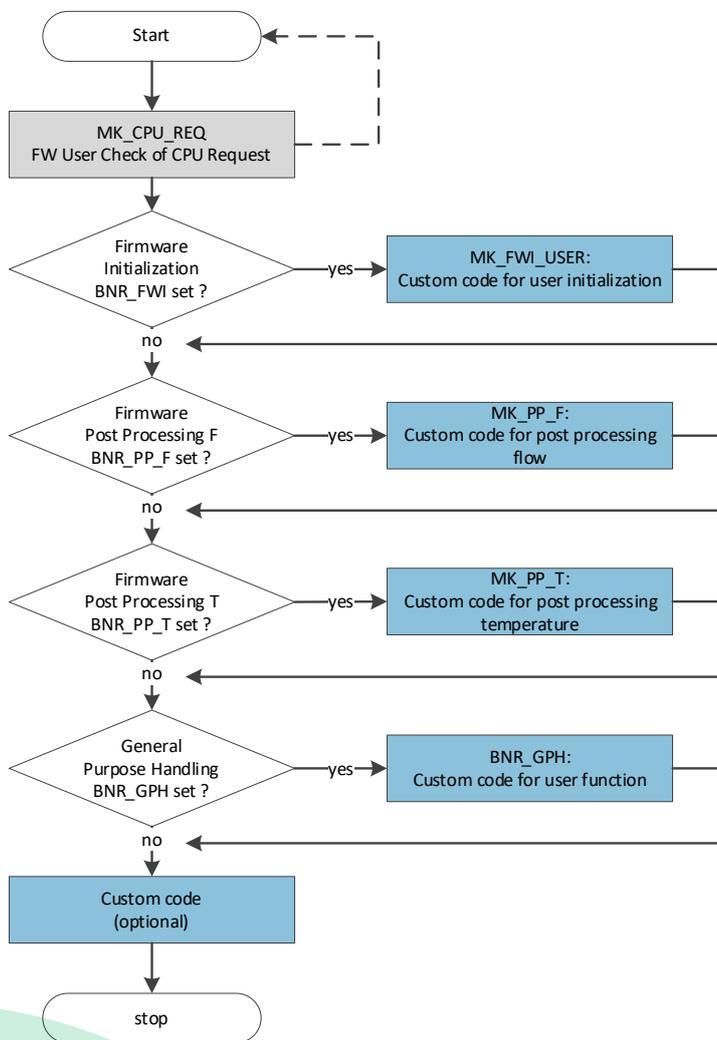


Figure 1: Firmware Custom Code

For illustrative purposes, a very simple example is used:

It depends on AS6031/40 configuration (Autoconfig Release Code, Post Processing F, Post Processing T and CPU Request General Purpose Handling) whether the jump into subroutines will be executed and, for debugging purposes, different numbers of pulses at GPIO3 are sent.

## 2 Preparation

### 2.1 Project Files

Please do not make any changes in the system folder. Copy all the files into your private folder for making changes if needed.

- The assembler source file (in our example: AS6031\_AS6040\_A1.C1.00.01.asm).
- The compiled .hex-file that is downloaded into the chip (in our example: AS6031\_AS6040\_A1.C1.00.01.hex).
- The project file, including configuration, firmware data and other data. It is also downloaded into the chip (in our example: AS6031\_AS6040\_A1.C1.00.01\_Template.ufc).
- .h files are headers containing the register descriptions of the device. They are needed for successful compilation (typically those are AS6031\_AS6040\_User\_FW.h, AS6031\_AS6040\_REG\_A1.h, and AS6031\_AS6040\_ROM\_A1\_common.h).

### 2.2 Open the .asm Example

- Launch UFC evaluation software and select Firmware menu.

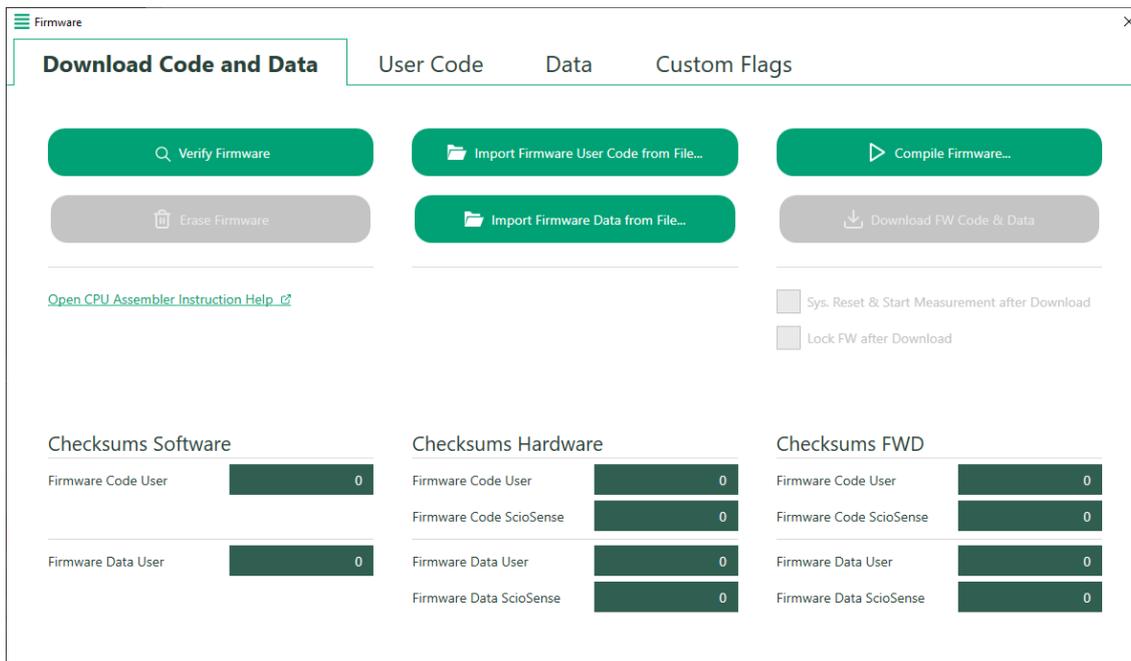


Figure 2: Firmware Menu

## 2.3 Assembler File Description

- Open .asm file with any text editor and adjust date, file name, author and notes on changes
- Search for the section of the source code that is designated to custom code

```

1  ;/*
2  ; *****
3  ; * Copyright by SciSense B.V.
4  ; * All rights are reserved.
5  ; *
6  ; * IMPORTANT - PLEASE READ CAREFULLY BEFORE COPYING, INSTALLING OR USING
7  ; * THE SOFTWARE.
8  ; *
9  ; * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
10 ; * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
11 ; * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
12 ; * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
13 ; * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
14 ; * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
15 ; * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
16 ; * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
17 ; * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
18 ; * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
19 ; * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
20 ; *****
21 ; */
22
23 ;---- File : AS6031_AS6040_A1.C1.00.01.asm
24 ;---- Created : 2021/11/12
25 ;---- last change : 2021/11/12
26 ;---- Authors : SciSense Support Team
27 ;---- Company : SciSense B.V.
28 ;---- Project : Assembler Template File for Customers, including debugging
29 ;----
30 ;---- Contents : Firmware entry when a post processing is requested.
31 ;---- The user can add own assembler code in this file.
32 ;----
33 ;
34
35 ; FOR DEBUGGING, using Measurement Task Request (RC_MT_REQ):
36 ; EC_MT_REQ [Bit 0]: VCC Voltage Measurement
37 ; EC_MT_REQ [Bit 1]: not used
38 ; EC_MT_REQ [Bit 2]: Time Of Flight Measurement
39 ; EC_MT_REQ [Bit 3]: Amplitude Measurement
40 ; EC_MT_REQ [Bit 4]: Amplitude Measurement Calibration
41 ; EC_MT_REQ [Bit 5]: Temperature Measurement
42 ; EC_MT_REQ [Bit 6]: High Speed Clock Calibration
43 ; EC_MT_REQ [Bit 7]: Zero Cross Calibration
44
45 ;----- Info about file headers -----
46 ; The file headers contain a short description in and output.
47 ; "(total) unused" refers to registers X,Y,Z and R and includes all internal subroutine calls - meaning the register is unchanged and never used in the whole process until jsubret.
48 ; Temporary RAM are RAM cells which are used and changed, but need no initialisation and need not be kept after the call
49 ; Permanent RAM are RAM and FWD cells which either need to be initialized and/or contain results for further processing after jsubret
50 ; Note that some routines use relative addresses, such that the permanent RAM address depends on the input parameter.

```

Figure 3: Spots for Custom Code

## 3 Assembler Programming

### 3.1 Declaration

First, variables and constants should be declared. In our example, these are:

- `DBG_COUNT_LOOP`. Counts the passing in CPU Request Loop
- `DBG_COUNT_PP_F`. Counts the passing in Post Processing F
- `DBG_COUNT_PP_T`. Counts the passing in Post Processing T
- `DBG_COUNT_GPH`. Counts the passing in General Purpose Handling
- `FW_VERSION` stands for the complete version number, including 4 bytes (ROM version, FW type and version number, major and minor release number, build).

```

61 ; FOR DEBUGGING (RAM REGISTER)
62 #define DEBUGGING                ; Activate Debugging (#ifdef DEBUGGING ... #endif)
63 #ifdef DEBUGGING
64     CONST DBG_COUNT_LOOP          0x30          ; Counts the passing in CPU Request Loop
65     CONST DBG_COUNT_PP_F         0x31          ; Counts the passing in Post Processing F
66     CONST DBG_COUNT_PP_T         0x32          ; Counts the passing in Post Processing T
67     CONST DBG_COUNT_GPH         0x33          ; Counts the passing in General Purpose Handling
68 #endif
69
70 CONST FW_ROMVERSION_REV          0xA1          ; The user can create here his own revision number. It is stored at the beginning of the program memory.
71 CONST FW_VERSION_NUM             0xC10000     ;
72 CONST FW_VERSION_MAJ             0x000000     ;
73 CONST FW_VERSION_MIN             0x000000     ;
74 CONST FW_VERSION_BLD             0x000001     ;
75 CONST FW_VERSION                 FW_VERSION_NUM + FW_VERSION_MAJ + FW_VERSION_MIN + FW_VERSION_BLD ;
  
```

Figure 4: Parameter Declaration

After declaration we add a code snippet for the following actions:

- Increment the register at location (`DBG_COUNT_LOOP`).
- Jump once into `MK_FWI_USER` subroutine after the start to initialize AS6031/40 with needed details.
- Jump to subroutine (`MK_PP_F`, `MK_PP_T`, `MK_GPH`), according to the CPU Request flag.
- It is also possible to use your own flag register and check your own bit number accordingly.

```

88 ;#####
89 ;##### Start of CPU_Request Loop
90 ;#####
91 MK_USER_FW:
92   ramadr SHR_CPU_REQ
93   skipBitC   r, BNR_FWI, 1      ;-- Check Firmware Init Flag
94   jsub      MK_FWI_USER        ;-- Jump to Firmware Init
95
96   ramadr SHR_CPU_REQ
97   skipBitC   r, BNR_PP_F, 1     ;-- Check Firmware PP-F Flag
98   jsub      MK_PP_F            ;-- Jump to Postprocessing F
99
100  ramadr SHR_CPU_REQ
101  skipBitC   r, BNR_PP_T, 1     ;-- Check Firmware PP-T Flag
102  jsub      MK_PP_T            ;-- Jump to Postprocessing T
103
104  ramadr SHR_CPU_REQ
105  skipBitC   r, BNR_GPH, 1      ;-- Check Firmware General Purpose Handling Flag
106  jsub      MK_GPH              ;-- Jump to General Purpose Handling
107
108  #ifdef DEBUGGING
109    ; FOR DEBUGGING, ONLY
110    ramadr DBG_COUNT_LOOP
111    incr r
112  #endif
113
114  goto      MK_STOP

```

Figure 5: Custom Code 1

## 3.2 Initialization of AS6031/40

It is important to call of the subroutine MK\_FWI\_USER for initialization of the AS6031/40. Especially it is important to initialize the USER RAM cells to zero, to load the Ultrasonic Release Delay (USM\_RLS\_x) into the System Handling Register (SHR) and to clear the Firmware Init Flag.

```

232 ;#####
233 ;##### Firmware Init User Routines
234 ;#####
235 MK_FWI_USER:
236 ; Requirements:
237 ; CPU Request Firmware Initialization
238 ; Triggered by bootloader sequence in ROM code,
239 ; automatically cleared when CPU stops
240
241 ; MANDATORY:
242 ; Set Bootloader Release Code in Firmware Data (in FWD)
243 ; 0x16B --> Autoconfig Release Code (in FWD) = 0xABCD7654
244
245 ; FOR EXAMPLE (IF NEEDED)
246 ; Initialization of USM_RLS_DLY_UP and USM_RLS_DLY_DOWN registers
247 ramadr 0x102 ; FWD cell t.b.d. by customer
248 move x, r
249 ramadr SHR_USM_RLS_DLY_U
250 move r, x
251 ramadr SHR_USM_RLS_DLY_D
252 move r, x
253
254 #ifdef DEBUGGING
255 ; FOR DEBUGGING, ONLY
256 ramadr DBG_COUNT_LOOP
257 clear r
258 ramadr DBG_COUNT_PP_F
259 clear r
260 ramadr DBG_COUNT_PP_T
261 clear r
262 ramadr DBG_COUNT_GPH
263 clear r
264 #endif
265
266 ; place your own code here !!!!!
267 ;jsub ROM_USER_RAM_INIT ; Optional: Initialising all USER RAM cells to 0
268
269
270 jsubret
  
```

Figure 6: Initialization of AS6031/40

### 3.3 Jump to Subroutine

The subroutine does the following:

- Checks, whether a Post Processing F triggered the CPU
- If yes, Jump to MK\_PP\_F

```

96 ramadr SHR_CPU_REQ
97 skipBitC r, BNR_PP_F, 1 ;-- Check Firmware PP-F Flag
98 jsub MK_PP_F ;-- Jump to Postprocessing F
  
```

Figure 7: Subroutine

- Increase DBG\_COUNT\_PP\_F what counts the passing in Post Processing F
- Send one pulse at GPIO3, with pulse width (10 ns), which corresponds to the time needed by DSP clock to set, clear the GPIO and jump into debug subroutine.
- Return to previous routine.

```

117 ;#####
118 ;##### Post Processing F
119 ;##### (after USM flow and amplitude measurement task)
120 ;#####
121 MK_PP_F:
122     ; Requirements:
123     ; Enables final post processing F
124     ; 0x0C6 --> TS_PP_F_EN[15] != 1
125
126     ; place your own code here !!!!!
127
128     ; FOR EXAMPLE
129     ramadr SRR_FEP_STF
130     skipBitS r, BNR_TOF_UPD, 1 ; Check if update for Ultrasonic Measurement
131     nop; jsub MK_FLOW_CALCULATION
132
133     skipBitS r, BNR_AM_UPD, 1 ; Check if update for Amplitude Measurement
134     nop; jsub MK_AMPLITUDE_CALCULATION
135
136     skipBitS r, BNR_AMC_UPD, 1 ; Check if update for Amplitude Calibration Measurement
137     nop; jsub MK_UPDATE_CALIBRATION
138
139
140     ; Post Processing F
141
142     #ifdef DEBUGGING
143     ; FOR DEBUGGING, ONLY
144     ramadr DBG_COUNT_PP_F
145     incr r
146     jsub MK_DBG_PULSE
147     #endif
148
149     ; CPU Request Post Processing F
150     ; automatically cleared when CPU stops
151
152
153     jsubret

```

Figure 8: Subroutine

Note:

The usage is similar for both, Post Processing T (for debugging, two pulses) and Post Processing GPH (for debugging, three pulses).

### 3.4 Compile

- Select the Firmware menu, see Figure 2 and press [Compile Firmware] button.
- After pressing [Compile Firmware] button, the output ‘Checksum FWD’ column will be changed, if there is no error.

### 3.5 Download to the Target

Attention

Be sure that the AS6031/40 is idle.

- After compiling or import 'User Code' using .ufc file or .hex file.
- Import 'Data', using .ufc file or .dat file.
- Press [Download FW Code & Data] button.
- Press [Verify Firmware] button.

Figure 9: Import User Code

#	Name	Signed	Value (dec)	Value (hex)	Factor	Calculated
0	Firmware Code User, Checksum	<input type="checkbox"/>	0	00000000	1	0
1	Firmware Data User, Checksum	<input type="checkbox"/>	0	00000000	1	0
2		<input type="checkbox"/>	0	00000000	1	0
3		<input type="checkbox"/>	0	00000000	1	0
4		<input type="checkbox"/>	0	00000000	1	0
5		<input type="checkbox"/>	0	00000000	1	0
6		<input type="checkbox"/>	0	00000000	1	0
7		<input type="checkbox"/>	0	00000000	1	0

Figure 10: Import Data

- Select the Firmware menu, see Figure 2 and press [Download FW Data & Code] button.
- The [Verify Firmware] button updates 'Checksum Hardware' column.
- 'Checksum Hardware' column and 'Checksum FWD' column should show the same values. Especially the row 'Firmware Code User' and Firmware Data User'.

Checksums Software		Checksums Hardware		Checksums FWD	
Firmware Code User	48AD	Firmware Code User	48AD	Firmware Code User	48AD
		Firmware Code SciSense	3F91	Firmware Code SciSense	3F91
Firmware Data User	242	Firmware Data User	242	Firmware Data User	242
		Firmware Data SciSense	470	Firmware Data SciSense	46F

*Figure 11: Verify Firmware*

(Note: Shown data in Figure 11 may vary from your window, especially ‘Firmware Data SciSense’ might be different.)

## 4 Summary / Result

### 4.1 Verify Code Executing Properly

Read registers (0x30..0x33) in the RAM memory, either by menu item ,RAM Memory‘ or by the ‘CPU Values‘ window.

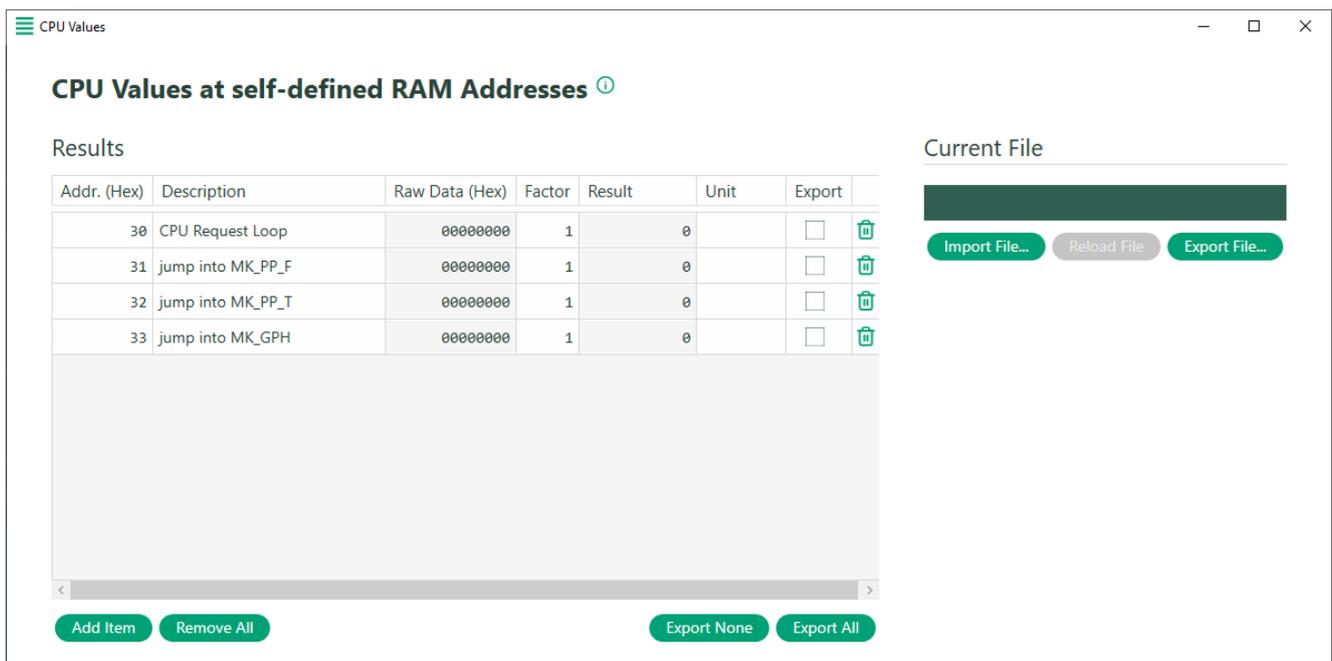


Figure 12: CPU Values window

Of course, monitoring the signal at GPIO3 is the final verification. In this example, the GPIO3 shows the jump into ‘Post Processing F’ subroutine and ‘Post Processing General Purpose Handling’ subroutine after TOF Cycle, then synchronous firmware interrupt (INTN).

See Figure 13 below, digital inputs are INTN and GPIO3, analog inputs show waveforms of TOF Cycle.

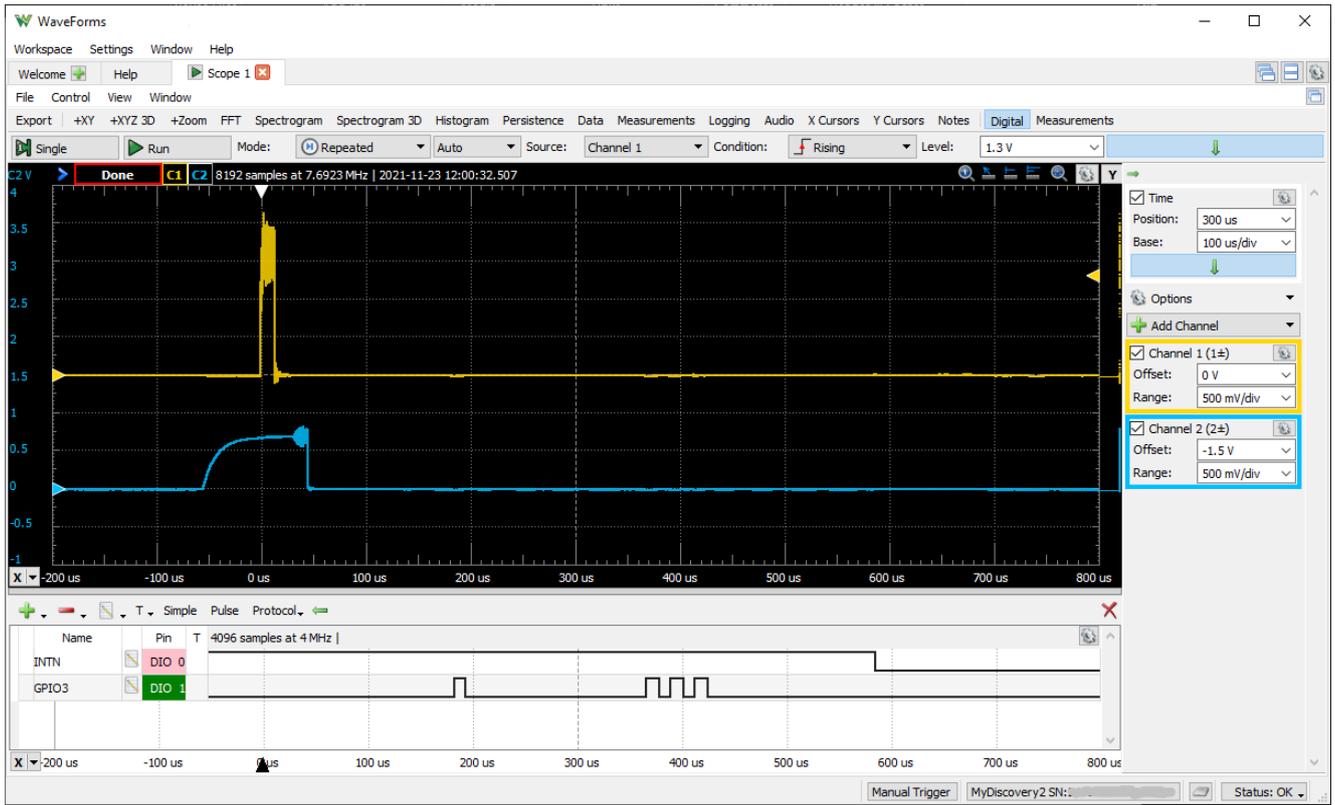


Figure 13: Pulse at GPIO3 after TOF Measurement Sequence

## 5 Copyrights & Disclaimer

Copyright SciSense 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 SciSense B.V. are covered by the warranty and patent indemnification provisions appearing in its General Terms of Trade. SciSense B.V. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein. SciSense 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 SciSense 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 SciSense B.V. for each application. This product is provided by SciSense 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.

SciSense 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 SciSense B.V. rendering of technical or other services.

## 6 Revision information

*Table 1: Revision history*

Revision	Date	Comment	Page
1	26.11.2021	First edition	All

### 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](http://www.sciosense.com)  
[info@sciosense.com](mailto:info@sciosense.com)

A decorative graphic at the bottom of the page consists of several overlapping, rounded shapes in various shades of green, creating a layered, landscape-like effect.