



Application Note

AN000586

TDC-GP30

How to Write Custom Firmware

v1-00 • 2019-Jan-10

Content Guide

1	Introduction.....	3	3.5	Compile	11
2	Preparation.....	4	3.6	Download Code to the Target	12
2.1	Project Files.....	4	4	Summary / Results	13
2.2	Open the .asm Example.....	4	4.1	Verify Code Executing Properly	13
2.3	Assembler File Description	5	5	Revision Information.....	15
3	Assembler Programming	6	6	Legal Information	16
3.1	Declaration	6			
3.2	Initialization of TDC-GP30.....	8			
3.3	Error Handling	9			
3.4	Jump to Toggling Subroutine	10			

1 Introduction

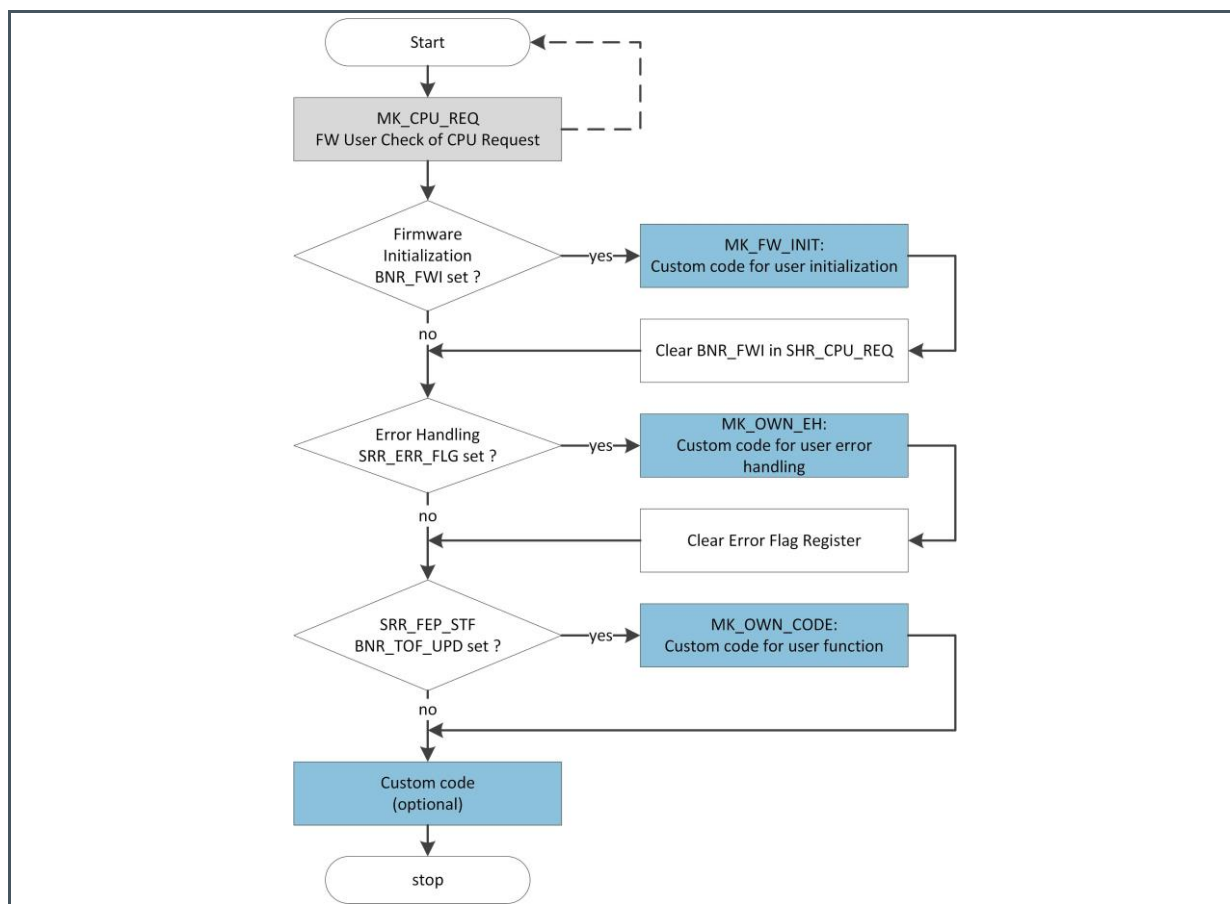
TDC-GP30 is a system-on-chip solution for ultrasonic flow metering. Using its integrated CPU and code memory, TDC-GP30 can be operated with a dedicated firmware for evaluation of results and operational control.

This application note describes how to write a customized firmware.

Following the naming convention, the modified file should be saved with a different name such as A1.F1.11.YY, where F indicates it is an **ams** example 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:

With each TOF measurement a counter (NUMBER_OF_RUNS) is increased. Also, a pulse is generated on GPIO4 that lasts until the next CPU call.

2 Preparation

2.1 Project Files

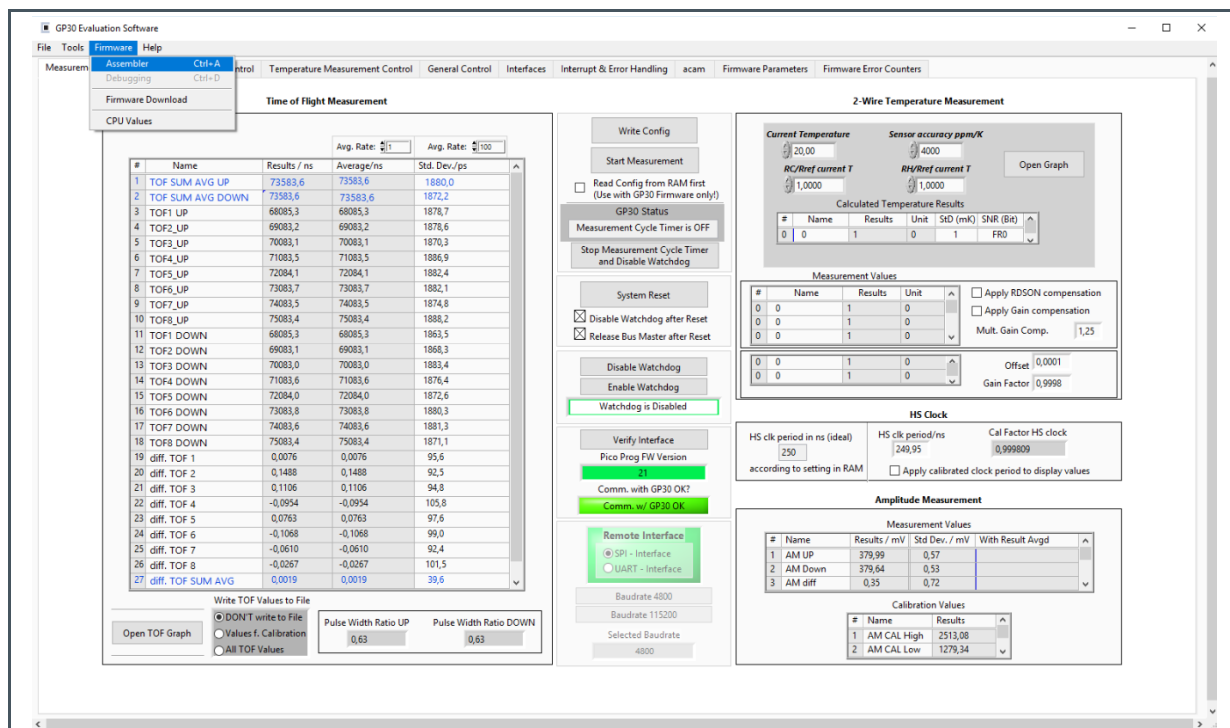
Please do not make any changes in the system folder. Copy all the files into your private folder for making changes if needed.

- The assembler source file (in our example: GP30Y_A1.F1.11.01_GPIO_toggling.asm).
- The compiled .hex-file that is downloaded into the chip (in our example: GP30Y_A1.F1.11.01_GPIO_toggling.hex).
- The firmware data file, including configuration and other data. It is also downloaded into the chip (in our example: GP30Y_A1.F1.11.01_GPIO_toggling.dat).
- .h files are headers containing the register descriptions of the device. They are needed for successful compilation (typically those are GP30Y_A1.D2.11.04.h, GP30Y_REG_A1.2.h, and GP30Y_ROM_A1.common.h).

2.2 Open the .asm Example

- Launch GP30 evaluation software and select Assembler in Firmware menu.

Figure 2 :
Firmware Menu > Assembler



2.3 Assembler File Description

- Open .asm file and adjust date, file name, author and notes on changes
- Tip: Double click on the #include files to show them on the menu tab.
- Search for the section of the source code that is designated to custom code

Figure 3 :
Spots for Custom Code

```

1  ;/*
2  ;*****
3  ; * Copyright by ams AG
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 LIMITED *
11 ; * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A *
12 ; * 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 LIMITED *
15 ; * TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR *
16 ; * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF *
17 ; * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING *
18 ; * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS *
19 ; * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
20 ;*****
21 ;*/
22
23 ;---- File : GP30Y_A1.F1.11.01_GPIO_toggling.asm
24 ;---- Version No. : 11
25 ;---- Build No. : 01
26 ;---- Created : 14/12/2018 (DD/MM/YYYY)
27 ;---- last change : 14/12/2018
28 ;---- Author : Matthias Hainz
29 ;---- Updated by : Matthias Hainz
30 ;---- Company : ams Sensors Germany GmbH
31 ;---- Project : GP30Y
32 ;----
33 ;---- Contents : Simple Firmware for toggling GPIO output
34
35
36 ;***** Include files *****
37
38 #include "GP30Y_REG_A1.2.h" ; Definition of Register addresses, bit positions and register actions
39 #include "GP30Y_A1.02.11.04.h" ; Definition of constants, memory organisation etc.
40 #include "GP30Y_ROM_A1.common.h" ; Definition of ROM routine start addresses
41
42 ;***** CONSTANTS / DEFINITIONS *****
43 ; The following is used for debugging
44 CONST NUMBER_OF_RUNS 0x40 ; Counts the runs (for debugging)
45 CONST DEBUG_ADDR_01 0x41 ; Address (for debugging)
46 CONST DEBUG_CONTENT_01 0x101010 ; Specified content (for debugging)
47 CONST DEBUG_ADDR_02 0x42 ; Address (for debugging)
48 CONST DEBUG_CONTENT_02 0x202020 ; Specified content (for debugging)
49
50 CONST REV_ADDRESS_VAR 3996 ; location of Version number in FW hex code with empty chip
51
52 CONST FW_ROMVERSION_REV 0xA1 ; ROM version
53 CONST FW_VERSION_HURT 0xF10000 ; F denotes ams example code
54 ;CONST_ELF_VERSION_HURT 0x001000 ; Modern version 1

```

3 Assembler Programming

3.1 Declaration

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

- `NUMBER_OF_RUNS`. Counts the number of TOF measurements
- `DEBUG_ADDR_xx` determines the USER RAM address for debugging.
- `DEBUG_CONTENT_xx` uses different values for debugging.
- `REV_ADDRESS_VAR` is the location of the version number in the FW hex code.
- `FW_VERSION` stands for the complete version number, including 4 bytes (ROM version, FW type and version number, major and minor release number, build).

Figure 4 :
Parameter Declaration

```

42 ;===== CONSTANTS / DEFINITIONS =====
43 ; The following is used for debugging
44 CONST NUMBER_OF_RUNS      0x40          ; Counts the runs (for debugging)
45 CONST DEBUG_ADDR_01      0x41          ; Address (for debugging)
46 CONST DEBUG_CONTENT_01   0x101010      ; Specified content (for debugging)
47 CONST DEBUG_ADDR_02      0x42          ; Address (for debugging)
48 CONST DEBUG_CONTENT_02   0x202020      ; Specified content (for debugging)
49
50 CONST REV_ADDRESS_VAR     3996          ; location of Version number in FW hex code with empty chip
51
52 CONST FW_ROMVERSION_REV   0xA1          ; ROM version
53 CONST FW_VERSION_NUM      0xF10000      ; F denots ams example code
54 CONST FW_VERSION_MAJ      0x001000      ; Major version 1
55 CONST FW_VERSION_MIN      0x000100      ; Minor version 1
56 CONST FW_VERSION_BLD      0x000001      ; Current build 01
57 CONST FW_VERSION          FW_VERSION_NUM + FW_VERSION_MAJ + FW_VERSION_MIN + FW_VERSION_BLD ;

```

After declaration we add a code snippet that:

- Increment the register at location (`NUMBER_OF_RUNS`).
- Jump once into `MK_FW_INIT` subroutine after the start to initialize TDC-GP30 with needed details.
- Jump at the error in the `MK_OWN_EH` subroutine for error handling.
- Read the `BNR_TOF_UPD` flag (TOF Update) in register `SRR_FEP_STF` (status register). If this bit is set, jump into `MK_OWN_CODE` subroutine.

Figure 5 :
Custom Code 1

```

63 ;=====
64 ;===== Check of CPU Requests - Main Firmware routine
65 ;=====
66 ; This routine checks the status flags for activity requests and calls the according routines
67 ; Inputs      :   Status flags in SRR_FEP_STF, SRR_ERR_FLG and SHR_CPU_REQ
68 ; Output      :   all processing done and status flags cleared
69 ; (total)unused : (all used)
70 ; Temporary RAM : -
71 ; Permanent RAM :
72 ; Routines used :
73 ;=====
74
75 MK_CPU_REQ:
76
77 ; ----- Counts the runs (for debugging)
78     ramadr  NUMBER_OF_RUNS
79     incr    r
80 ; -----
81
82 ;=====
83 ramadr    SHR_CPU_REQ          ;-- Set RAM Address to SHR_CPU_REQ
84 skipBitC  r, BNR_FWI, 1        ;-- Check Firmware Init Flag
85 jsub      MK_FW_INIT          ;-- Jump to Firmware Init if it is first start
86 ;=====
87
88 ;=====
89 ramadr    SRR_ERR_FLG          ;-- Set RAM Address to SRR_ERR_FLG
90 getflag r                        ;-- Set the signum and zero flag according to the addressed register
91 skipEQ 1                        ;-- Skip one commands if SRR_ERR_FLG is equal to zero.
92 jsub      MK_OWN_EH            ;-- Jump to User Error Handling Subroutine
93 ;=====
94
95 ;=====
96 ramadr    SRR_FEP_STF          ;-- Set RAM Address to SRR_FEP_STF
97 skipBitC  r, BNR_TOF_UPD, 1    ;-- Check US_TOF_UPD Flag
98 jsub      MK_OWN_CODE          ;-- Jump to User Code
99 ;=====
100
101 MK_STOP:
102
103     ramadr  SHR_CPU_REQ          ; Set RAM Address to SHR_CPU_REQ
104     clear   r                    ; Clear SHR_CPU_REQ
105
106     clrwdt                        ; Clearing watchdog
107     stop
108
109 ;=====
110 ;           End of Main Program MK_CPU_REQ
111 ;=====

```

3.2 Initialization of TDC-GP30

The call of the subroutine MK_FW_INIT is important to initialize TDC-GP30. Especially to initialize the USER RAM cells to zero, to load the First Hit Level (FHL) into the System Handling Register (SHR) and to clear the Firmware Init Flag.

Figure 6 :
Initialization of TDC-GP30

```

186 ;=====
187 ;===== place your own code here for initialization
188 ;=====
189 ; The subroutine is to configure a fixed value for the ZCD_FHL_U and ZCD_FHL_D bits.
190 ; These First Hit Level (FHL) values are specific to the hardware.
191 ; Inputs      : -
192 ; Output     : -
193 ; (total)unused : (all used)
194 ; Temporary RAM : -
195 ; Permanent RAM : -
196 ; Routines used : -
197 ;=====
198
199 MK_FW_INIT:
200
201     jsub      ROM_USER_RAM_INIT      ; Initialising all USER RAM cells to 0
202
203     ramadr    FWD_R1_FHL_VALUE      ;---- Value for the first hit level
204     move      x, r
205
206     ramadr    SHR_ZCD_FHL_U          ; Setting the same FHL for Up and Down
207     move      r, x
208     ramadr    SHR_ZCD_FHL_D
209     move      r, x
210
211 MK_FW_INIT_END:
212
213     ramadr    SHR_CPU_REQ            ; Set RAM Address to SHR_CPU_REQ
214     bitclr    r, BNR_FWI            ; Clear Firmware Init Flag
215
216     jsubret                                ; Jump Back
217
218 ;=====
219 ;      End of Subroutine MK_FW_INIT
220 ;=====

```


3.3 Error Handling

The subroutine is not active yet as it is not written for now. This routine only clears the error flag register.

Figure 7 :
Error Handling Subroutine

```

113 ;=====
114 ;===== place your own code here for error handling
115 ;=====
116 ; The subroutine is not active yet and just clears the error flag register
117 ; Inputs      : -
118 ; Output      : -
119 ; (total)unused : (all used)
120 ; Temporary RAM : -
121 ; Permanent RAM : -
122 ; Routines used : -
123 ;=====
124
125 MK_OWN_EH:
126
127 ; ----- Writes into RAM (for debugging)
128     move y, DEBUG_CONTENT_01
129     ramadr DEBUG_ADDR_01
130     move r, y
131 ; -----
132
133     ;##### Place your code here for error handling #####
134     nop                                ;--
135 ;   nop                                ;--
136 ;   nop                                ;--
137 ;   nop                                ;--
138     ;#####
139
140 MK_OWN_EH_END:
141
142     ramadr     SHR_EXC                ; Set RAM Address to SHR_EXC
143     bitset     r,  BNR_EF_CLR        ; clear error flag register
144
145     jsubret
146
147 ;#####
148 ;       End of Subroutine MK_OWN_EH
149 ;#####

```

3.4 Jump to Toggling Subroutine

The subroutine does the following:

- Checks, whether a TOF measurement triggered the CPU
- If yes, Jump to MK_OWN_CODE

Figure 8 :
Main Custom Subroutine

```

95 ;*****
96 ramadr    SRR_FEP_STF          ;-- Set RAM Address to SRR_FEP_STF
97 skipBitC  r, BNR_TOF_UPD, 1    ;-- Check US_TOF_UPD Flag
98 jsub      MK_OWN_CODE          ;-- Jump to User Code
99 ;*****

```

- Set the pointer address to the SHR_GP0 register (GPO control register).
- Set and clear bit BNR_GP04_OUT (GPIO4). This timing takes about 200 ns, which corresponds to the time needed by DSP clock to set and clear the GPIO.
- Return to previous routine.

Figure 9 :
Main Custom Subroutine

```

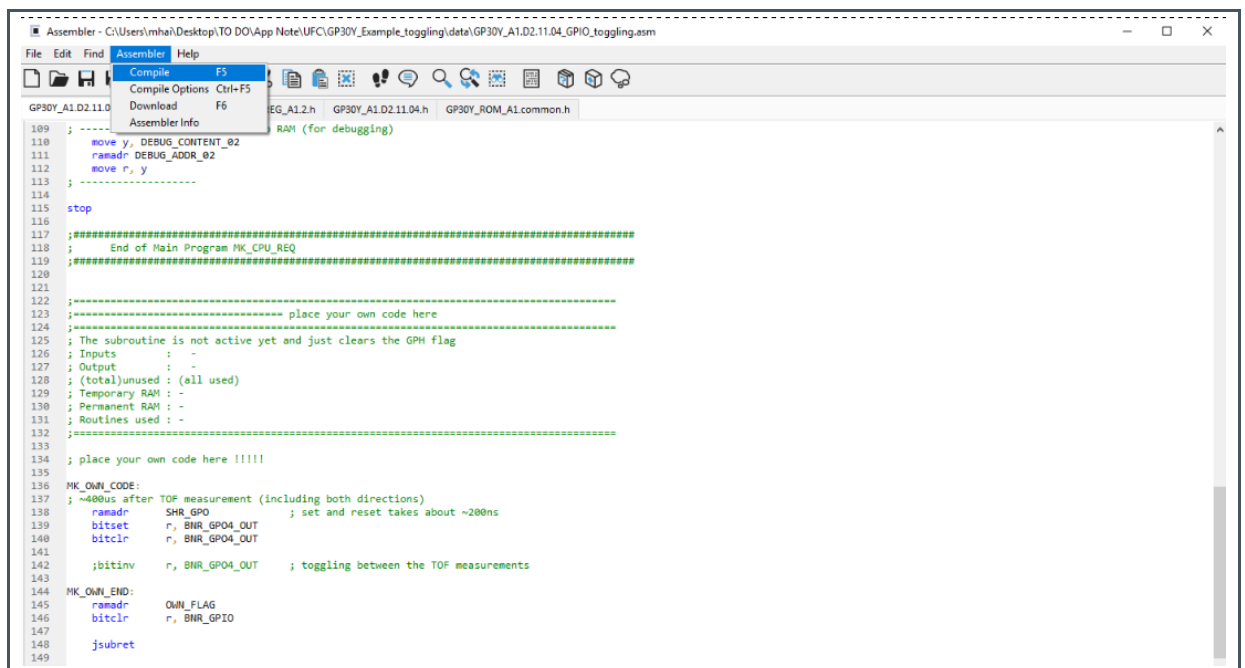
151 ;=====
152 ;===== place your own code here for post processing
153 ;=====
154 ; The subroutine is simple firmware for toggling GPIO output
155 ; Inputs      : -
156 ; Output      : -
157 ; (total)unused : (all used)
158 ; Temporary RAM : -
159 ; Permanent RAM : -
160 ; Routines used : -
161 ;=====
162
163 MK_OWN_CODE:
164
165 ; ----- Writes into RAM (for debugging)
166 move y, DEBUG_CONTENT_02
167 ramadr DEBUG_ADDR_02
168 move r, y
169 ; -----
170
171 ; ~400us after TOF measurement (including both directions)
172 ramadr    SHR_GP0              ; set and reset takes about ~200ns
173 bitset    r, BNR_GP04_OUT
174 bitclr    r, BNR_GP04_OUT
175
176 ;bitinv    r, BNR_GP04_OUT      ; toggling between the TOF measurements
177
178 MK_OWN_END:
179
180 jsubret
181
182 ;=====
183 ; End of Subroutine MK_OWN_CODE
184 ;=====

```

3.5 Compile

- Select the Assembler menu, Compile (or press F5)
- After pressing Compile and Download, the output window should show “Processing was successful” and “Hex-File transferred to Download window”. The Compile Options are used to configure whether the download is executed after each compile.

Figure 10 :
Assembler Output



3.6 Download Code to the Target



Attention

Be sure that the TDC-GP30 is idle.

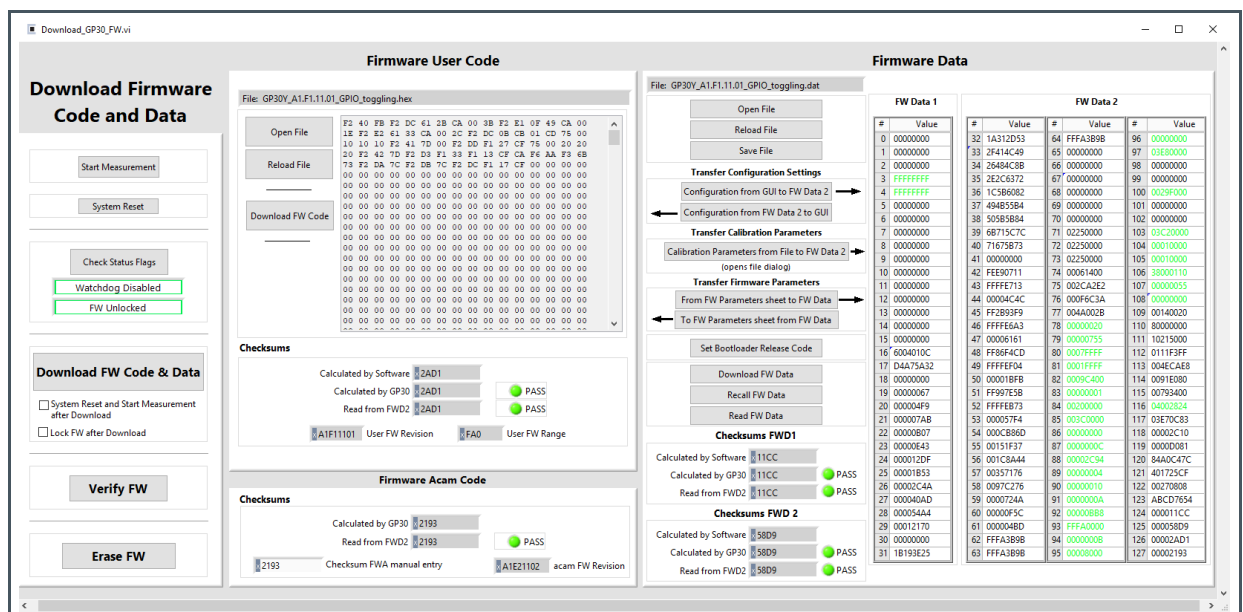
- Open .asm file and .dat file.
- Press button “Verify FW”.
- Copy “Calculated Checksum by GP30” to field “Checksum FWA manual entry”.
- Press button “Download FW Code & Data”.



Information

The assembler transfers the compiled code directly into the download window. So the new code can be downloaded directly by pressing “Download FW Code”.

Figure 11 :
Firmware Download Window

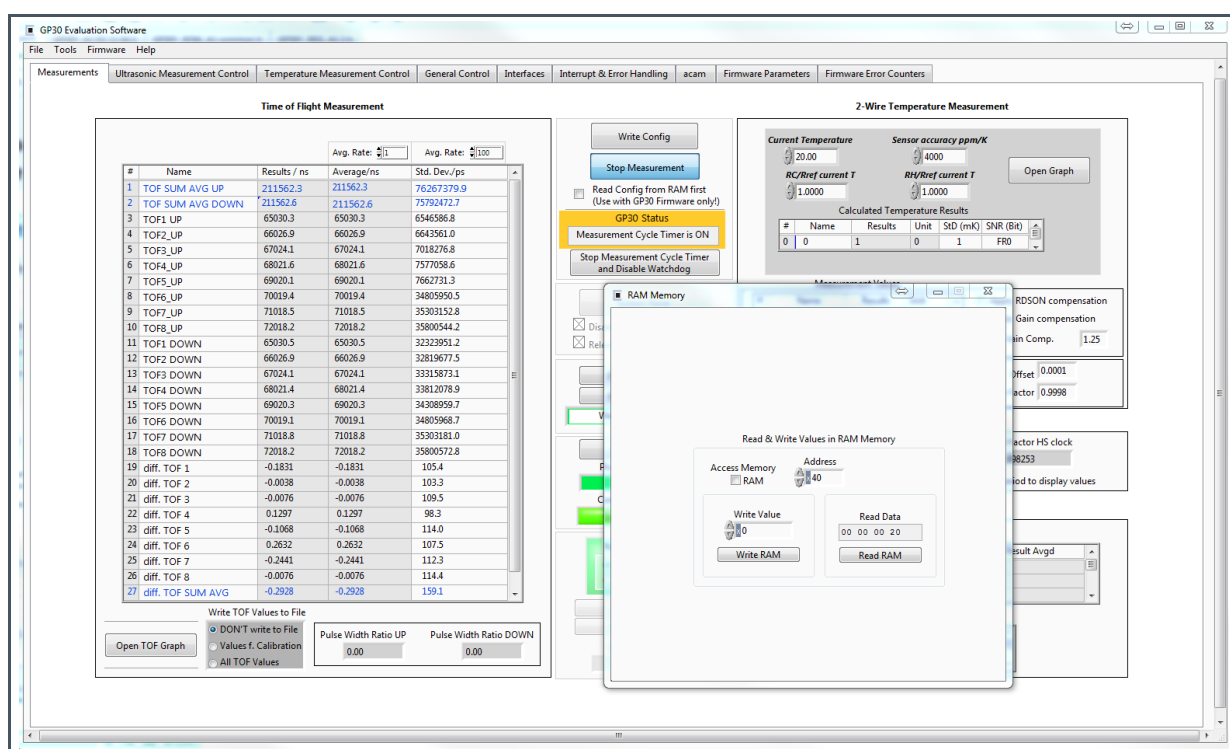


4 Summary / Results

4.1 Verify Code Executing Properly

- Read registers 0x40 in the RAM memory, either by menu items Tools-RAM Memory or by the CPU Values window.

Figure 12 :
RAM Window



Of course, monitoring the signal at GPIO4 is the final verification. In this example, the GPIO4 is toggling after each TOF Cycle.

See Figure 13 below, red waveform refers to GPIO4 and yellow waveforms refer to TOF Cycle.

Figure 13:
Pulse at GPIO4 after TOF Measurement Sequence



(time base = 100 μ s/division; voltage = 1 V/division)

5 Revision Information

Changes from previous version to current revision v1-00	Page
Initial version for release	

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

6 Legal Information

Copyrights & Disclaimer

Copyright ams AG, Tobelbader Strasse 30, 8141 Premstaetten, Austria-Europe. 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.

Information in this document is believed to be accurate and reliable. However, ams AG does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Applications that are described herein are for illustrative purposes only. ams AG makes no representation or warranty that such applications will be appropriate for the specified use without further testing or modification. ams AG takes no responsibility for the design, operation and testing of the applications and end-products as well as assistance with the applications or end-product designs when using ams AG products. ams AG is not liable for the suitability and fit of ams AG products in applications and end-products planned.

ams AG 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 or applications described herein. No obligation or liability to recipient or any third party shall arise or flow out of ams AG rendering of technical or other services.

ams AG reserves the right to change information in this document at any time and without notice.

RoHS Compliant & ams Green Statement

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

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

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

Headquarters

ams AG
Tobelbader Strasse 30
8141 Premstaetten
Austria, Europe
Tel: +43 (0) 3136 500 0

Please visit our website at www.ams.com

Buy our products or get free samples online at www.ams.com/Products

Technical Support is available at www.ams.com/Technical-Support

Provide feedback about this document at www.ams.com/Document-Feedback

For sales offices, distributors and representatives go to www.ams.com/Contact

For further information and requests, e-mail us at ams_sales@ams.com