



Application Note

AN000536

TDC-GP30-F01

How to Add Custom Code

v1-01 • 2018-Dec-10

Content Guide

1	Introduction.....	3	4	Summary / Results	11
2	Preparation.....	5	4.1	Verify Code Executing Properly.....	11
2.1	Project Files.....	5	5	Revision Information.....	12
2.2	Open the .asm Example.....	5	6	Legal Information	13
2.3	Assembler File Description	6			
3	Assembler Programming	7			
3.1	Declarations	7			
3.2	Compile	9			
3.3	Download Code to the Target	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 result evaluation and operation control.

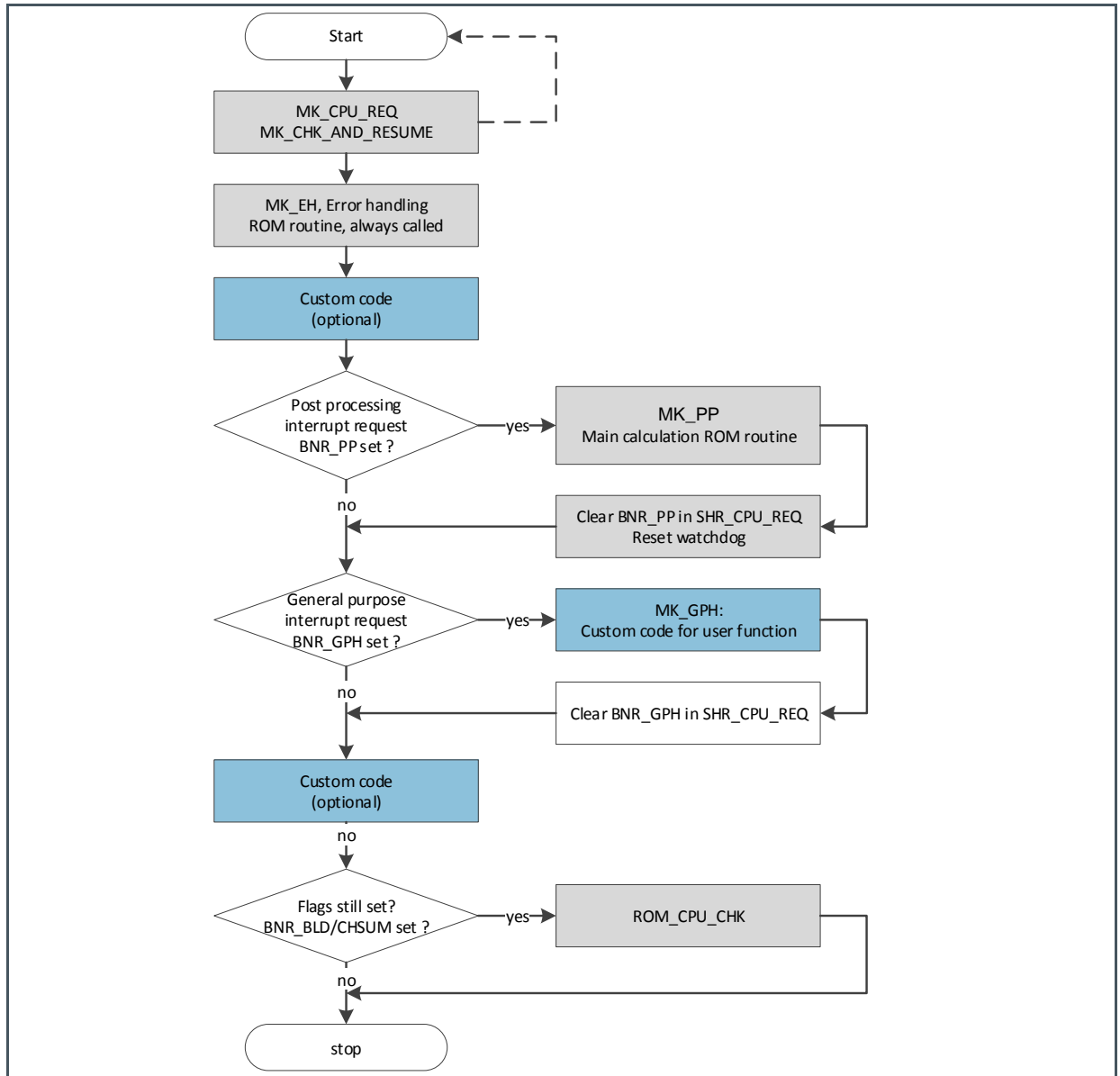
The TDC-GP30-F01 variant comes already with a preprogrammed **ams** firmware for flow and temperature calculation. The main part of this firmware is protected, but there is also an open memory space that allows a user to add some custom code.

This application note describes how a customized code is added to the open part of the **ams** flow firmware. The firmware user code is referred to as FWU. The relevant file is named GP30Y_A1.D2.11.xx.

Following the naming convention, the modified file should be saved with a different name such as A1.C2.11.YY, where C indicates it is a custom version.

Figure 1 shows the basic flow diagram of the main program and the various points where custom code can be added.

Figure 1:
ams Firmware Open Part - FWU



For illustrative purposes, a very simple example is used:

With each TOF measurement a counter (NUMBER_OF_RUNS) is increased. The counter is reset every time it reaches a specified level (MAX_NUMBER) and a pulse is generated on GPIO0 that lasts until the next CPU call.



Information

The pulse interface needs to be turned off, as the GPIO0 is used.

2 Preparation

2.1 Project Files

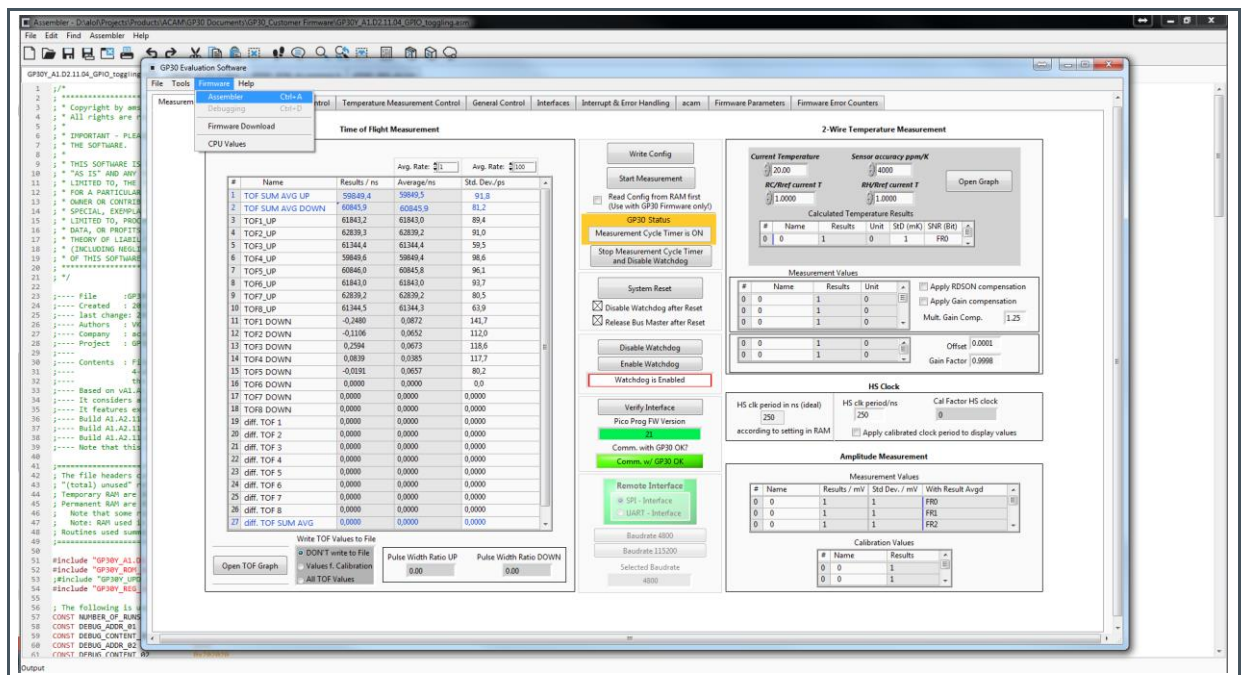
Please do not make any changes in the system folder. Copy all the files to your private folder if you are planning to make changes to the code. The relevant files are:

- The is the modified assembler source file, based on the default file GP30Y_A1.D2.11.04.asm. (in our example: GP30Y_A1.C2.11.01.asm)
- The compiled .hex-file, that is downloaded into the chip. (in our example: GP30Y_A1.C2.11.01.hex)
- The firmware data file, including configuration and other data. It is also downloaded into the chip. (in our example: GP30Y_A1.C2.11.01.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
- Add your code. Small snippets can be inserted directly. Larger code sections should be accessed by jumping to subroutines.

Figure 3 :
Spots for Custom Code

```

94 ; ----- Always call error handling
95 ;ramadr SHR_CPU_REQ ;-- Set RAM Address to SHR_CPU_REQ
96 ;skipBitC r, BNR_EH, 1 ;-- Check Error Handling Flag: this flag shall not be used since it triggers no
97 jsub MK_EH ;-- Always jump to User Error Handling Subroutine; now modified ROM routine; cl
98
99 ;***** Cusustom code ***** ①
100 ;***** End cusustom code *****
101
102 ; -----Here comes the main call for post processing
103 ramadr SHR_CPU_REQ ;-- Set RAM Address to SHR_CPU_REQ
104 skipBitC r, BNR_PP, 3 ;-- Check User Memory Post Processing Flag
105 jsub MK_PP ;-- Jump to User Post Processing
106 ; in case a communication request was set, the FW has requested an interrupt at next IDLE state
107 ; clear the communication request in any case to prevent more subsequent interrupts
108 ramadr SHR_EXC
109 bitset r, BNR_COM_REQ_CLR ; change of A1.A2.11.04
110
111 ; ----- General purpose routine: is only called after a general purpose request.
112 ; replace register SHR_CPU_REQ and bit BNR_GPH with any other desired flag to have your routine called as desire
113 ; remember to reset the same flag bit inside the routine MK_GPH. Else the routine will be called again after next r
114 ramadr SHR_CPU_REQ ;-- Set RAM Address to SHR_CPU_REQ; update of vA1.A1.12.01
115 skipBitC r, BNR_GPH, 1 ;-- Check General Purpose Flag
116 jsub MK_GPH ;-- Jump to User General Purpose Handling
117
118 ;***** Custom code ***** ②
119 ; jsub MK_CUSTOM_CODE
120 ;***** End cusustom code *****
121
122 ; ----- End of the main program
123 ramadr SHR_CPU_REQ ;-- Set RAM Address to SHR_CPU_REQ
124 skipBitS r, BNR_BLD_EXC, 2 ;-- check if BNR_BLD_EXC or BNR_CHKSUM may still be set; then skip stop; change
125 skipBitS r, BNR_CHKSUM, 1
126

```

- 1 Custom code section 1: Between error handling and post-processing
- 2 Custom code section 2: Between post processing and end of main program

3 Assembler Programming

3.1 Declarations

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

- NUMBER_OF_RUNS. Counts the number of TOF measurements
- MAX_NUMBER. If NUMBER_OF_RUNS exceeds this limit then the GPIO is set and the counter is reset to zero.
- DEBUG_ADDR_xx determines the USER RAM address for debugging. DEBUG_CONTENT_xx uses different values for debugging.
- OWN_FLAG is a debug register used to store the content of the BNR_TOF_UPD flag in register SRR_FEP_STF. This indicates whether a new TOF measurement is available. The mirror is needed because the MK_PP routine resets this flag.
- BNR_GPIO. Defines the bit number of the mirrored flag in the mirror register

Figure 4:
Parameter Declaration

```

62 ; The following is used for debugging
63 CONST NUMBER_OF_RUNS      0x38      ; RAM address actual counter value
64 CONST MAX_NUMBER          0x10      ; Maximum counter value
65 CONST DEBUG_ADDR_01       0x40      ; RAM address of a debug value
66 CONST DEBUG_CONTENT_01    0x101010  ; Arbitrary debug value
67 CONST OWN_FLAG            0x42      ; Own Flag Register, mirrors BNR_TOF_UPD flag in register SRR_FEP_STF
68 CONST BNR_GPIO            0          ; Bit Number of OWN_FLAG register
69

```

After the error handling we added a code snippet that:

- Clears the GPIO0, setting it low
- Copies the value of the TOF update flag into our own flag

Figure 5:
Custom Code 1

```

101 ;##### Custom code part 1 #####
102 ramadr SHR_GPO      ; Clear GPIO0 with the (next) CPU call
103 bitclr r, BNR_GPIO_OUT ;
104
105 ramadr SRR_FEP_STF  ;-- Set RAM Address to SRR_FEP_STF
106 skipBitC r, BNR_TOF_UPD, 2 ;-- Check US_TOF_UPD Flag
107 ramadr OWN_FLAG     ;-- Save as OWN_FLAG (SRR_FEP_STF will be reset by MK_PP)
108 bitset r, BNR_GPIO
109 ;##### END Custom code part 1 #####
110

```

After the post-processing we added a jump into the major custom subroutine. The subroutine does the following:

- Check, whether a TOF measurement triggered the CPU
- If yes, increase the counter
- Compare the counter with the maximum. If this is exceeded, jump into the event routine
- The event routine writes the debug value. This is just for demonstration how you can debug
- The counter is reset to zero
- GPIO0 is set HIGH
- The OWN flag is reset

Figure 6:
Main Custom Subroutine

```

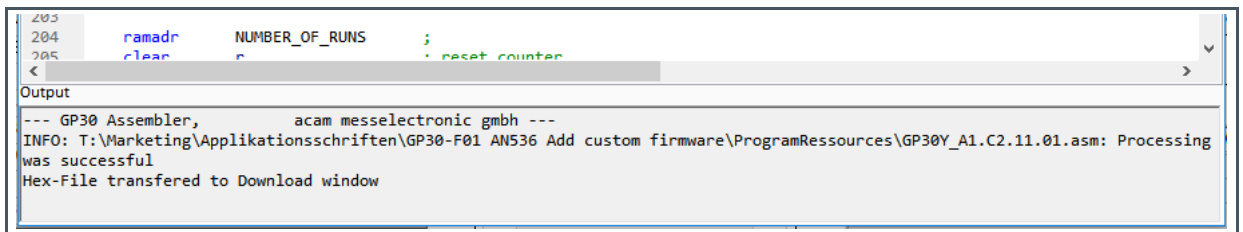
173 ;##### Custom code part 2 #####
174 MK_CUSTOM_CODE:
175 ; ----- Counts the TOF measurements
176   ramadr   OWN_FLAG      ;-- Set RAM Address to SRR_FEP_STF
177   skipBitC r, BNR_GPIO, 1 ;-- Check US_TOF_UPD Flag
178   jsub     MK_CUSTOM_TOF
179
180   jsubret
181 ; ----- END MK_CUSTOM_CODE
182
183 MK_CUSTOM_TOF:
184 ; ----- Counts the TOF measurements
185   ramadr   NUMBER_OF_RUNS ;
186   incr     r               ; incr counter by 1
187
188   move     y,r
189   clear    x
190   add      x, MAX_NUMBER
191   sub      y,x             ; check whether maximum number is reached
192   skipPos  1
193   jsub     MK_CUSTOM_EVENT
194   ;nop
195
196   jsubret
197 ; ----- END MK_CUSTOM_TOF
198
199 MK_CUSTOM_EVENT:
200   move     y, DEBUG_CONTENT_01 ; Write 0x101010 to address 40
201   ramadr   DEBUG_ADDR_01      ; for debugging to prove that this sequence was reached
202   move     r, y
203
204   ramadr   NUMBER_OF_RUNS ;
205   clear    r               ; reset counter
206
207 ; ----- Set BNR_GPIO bit and OWN_FLAG register
208   ramadr   OWN_FLAG          ; clear custom OWN_FLAG
209   bitclr   r, BNR_GPIO
210
211   ramadr   SHR_GPO           ; set GPIO0 and reset takes about ~200ns
212   bitset   r, BNR_GPIO_OUT
213
214   jsubret
215 ; ----- END MK_CUSTOM_EVENT
216
217 ;##### END Custom code part 2 #####

```


3.2 Compile

- 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 is used to configure whether after each compiling the download is executed.

Figure 7:
Assembler Output



```
203  
204 ramadr NUMBER_OF_RUNS ;  
205 clear r ; reset counter  
< >  
Output  
--- GP30 Assembler, acam messelectronic gmbh ---  
INFO: T:\Marketing\Applikationsschriften\GP30-F01 AN536 Add custom firmware\ProgramRessourcen\GP30Y_A1.C2.11.01.asm: Processing  
was successful  
Hex-File transferred to Download window
```

3.3 Download Code to the Target



Attention

Be sure that the TDC-GP30-F01 is idle.

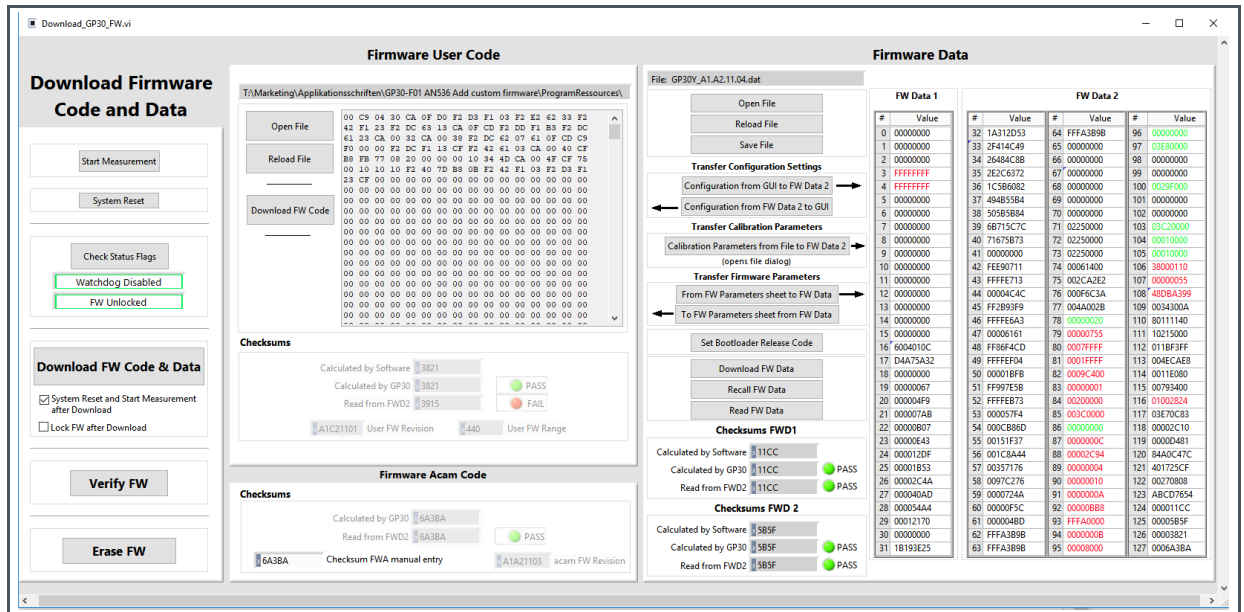
- Open .asm file and .dat file.
- Press button “Verify FW”.
- Copy “Calculated Checksum by GP30” to field “Checksum FWA manual entry”.
- Press button “Download FW Code & Data”.



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 8:
Firmware Download Window

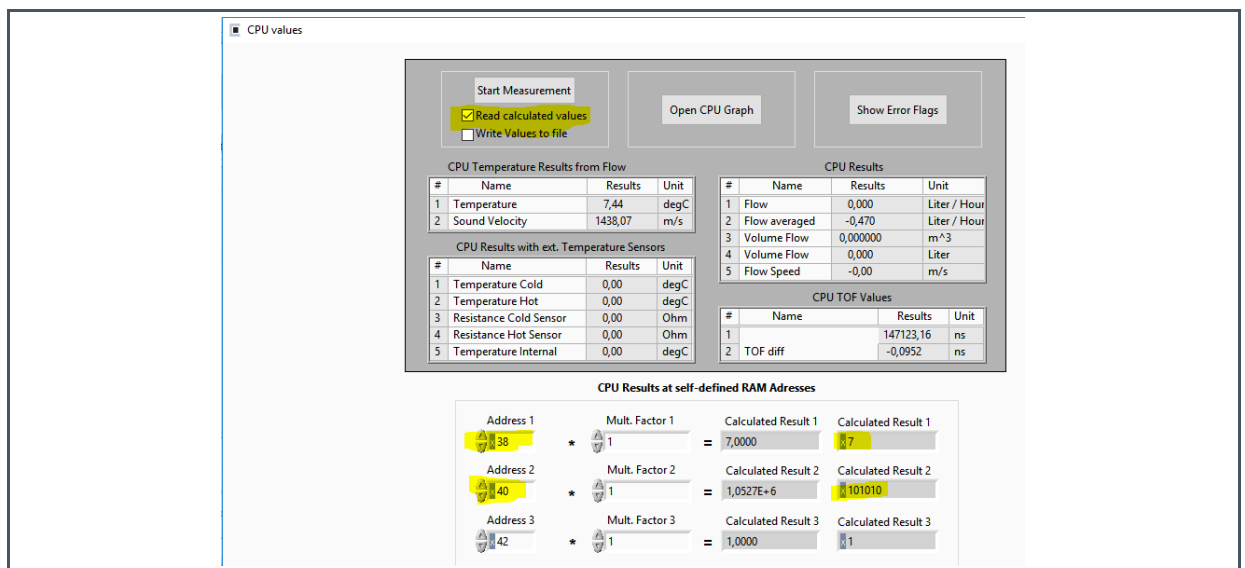


4 Summary / Results

4.1 Verify Code Executing Properly

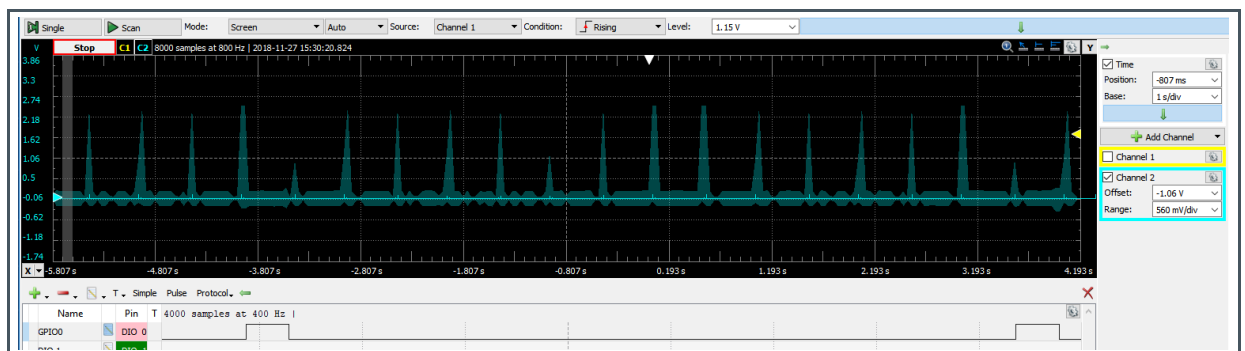
- Read registers 0x38 and 0x40 in the RAM memory, either by menu items Tools-RAM Memory or by the CPU Values window.
- 0x38 will show the current counter status, 0x40 should display the debug value (e.g. 0x101010)

Figure 9:
CPU Values Window



Of course, monitoring the signal at GPIO0 is the final verification. In this example, the GPIO0 is set every 15 measurements.

Figure 10:
Fire Burst and GPIO0



5 Revision Information

Changes from previous version to current revision v1-01	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