This product, formerly sold by ams AG, and before that optionally by either Applied Sensors GmbH, acam-messelectronic GmbH or Cambridge CMOS Sensors, is now owned and sold by

ScioSense

The technical content of this document under ams / Applied Sensors / acam-messelectronic / Cambridge CMOS Sensors is still valid.

Contact information

Headquarters:

Sciosense B.V.

High Tech Campus 10

5656 AE Eindhoven

The Netherlands

info@sciosense.com

www.sciosense.com

Application Note

AN000586

TDC-GP30

How to Write Custom Firmware

v1-00 • 2019-Jan-10

Content Guide

1 Introduction

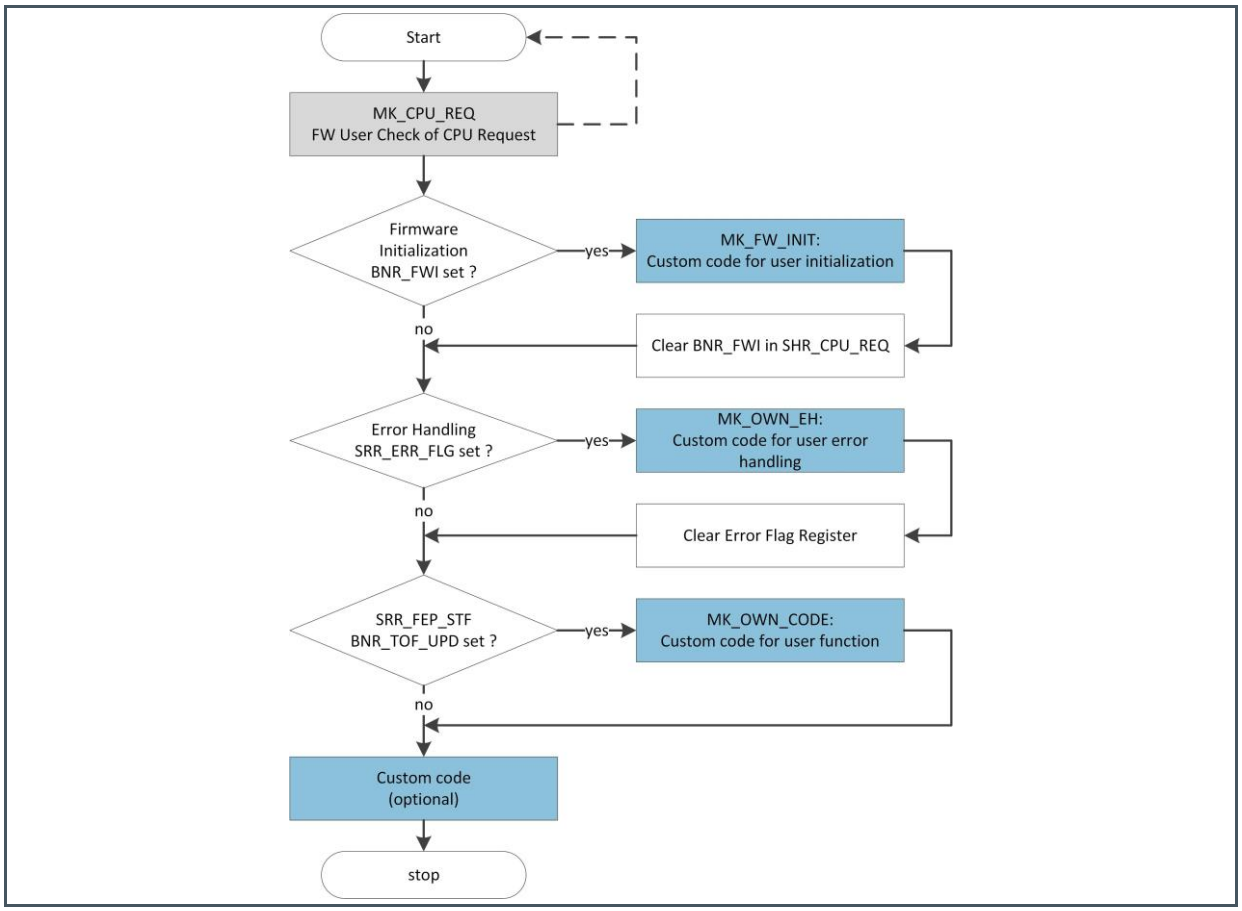
TDC-GP30 is a system-on-chip solution for ultrasonic flow metering. Using its integrated CPU and code memory, TDC-GP30 can be operated with a dedicated firmware for evaluation of results and operational control.

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

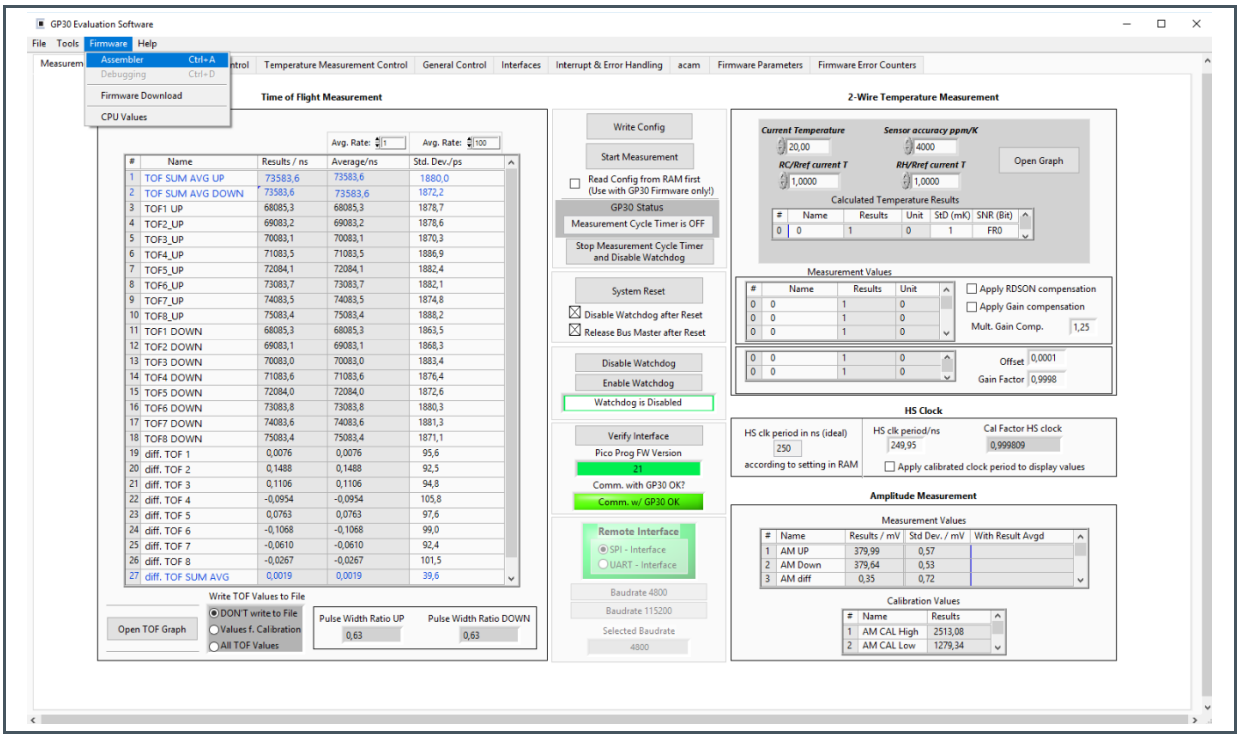
Please do not make any changes in the system folder. Copy all the files into your private folder for making changes if needed.

- 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.

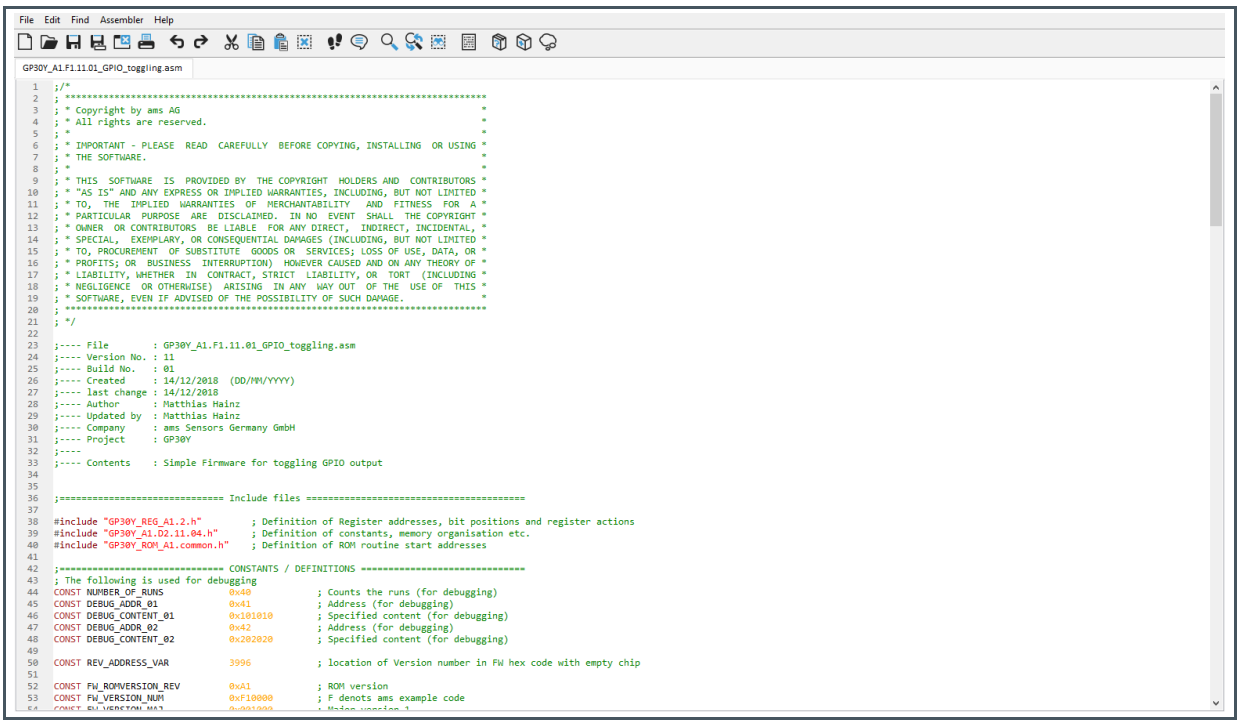
Figure 2 :
Firmware Menu > Assembler



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



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

```
;============================== CONSTANTS / DEFINITIONS ==============================
; The following is used for debugging
CONST NUMBER_OF_RUNS            0x40            ; Counts the runs (for debugging)
CONST DEBUG_ADDR_01             0x41            ; Address (for debugging)
CONST DEBUG_CONTENT_01          0x101010        ; Specified content (for debugging)
CONST DEBUG_ADDR_02             0x42            ; Address (for debugging)
CONST DEBUG_CONTENT_02          0x202020        ; Specified content (for debugging)

CONST REV_ADDRESS_VAR           3996            ; location of Version number in FW hex code with empty chip

CONST FW_ROMVERSION_REV         0xA1            ; ROM version
CONST FW_VERSION_NUM            0xF10000        ; F denots ams example code
CONST FW_VERSION_MAJ            0x001000        ; Major version 1
CONST FW_VERSION_MIN            0x000100        ; Minor version 1
CONST FW_VERSION_BLD            0x000001        ; Current build 01
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

```
;===============================================================================
;================== Check of CPU Requests - Main Firmware routine
;===============================================================================
; This routine checks the status flags for activity requests and calls the according routines
; Inputs        :   Status flags in SRR_FEP_STF, SRR_ERR_FLG and SHR_CPU_REQ
; Output        :   all processing done and status flags cleared
; (total)unused : (all used)
; Temporary RAM : -
; Permanent RAM :
; Routines used :
;===============================================================================

MK_CPU_REQ:

; ------------------- Counts the runs (for debugging)
    ramadr NUMBER_OF_RUNS
    incr        r
; --------------------

;*******************************************************************************
ramadr      SHR_CPU_REQ                 ;-- Set RAM Address to SHR_CPU_REQ
skipBitC    r,  BNR_FWI, 1              ;-- Check Firmware Init Flag
    jsub        MK_FW_INIT              ;-- Jump to Firmware Init if it is first start
;*******************************************************************************

;*******************************************************************************
    ramadr  SRR_ERR_FLG                 ;-- Set RAM Address to SRR_ERR_FLG
    getflag r                           ;-- Set the signum and zero flag according to the addressed register
    skipEQ  1                           ;-- Skip one commands if SRR_ERR_FLG is equal to zero.
        jsub    MK_OWN_EH               ;-- Jump to User Error Handling Subroutine
;*******************************************************************************

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

MK_STOP:

    ramadr      SHR_CPU_REQ             ; Set RAM Address to SHR_CPU_REQ
    clear       r                       ; Clear SHR_CPU_REQ

    clrwdt                              ; Clearing watchdog
stop

;###############################################################################################
;       End of Main Program MK_CPU_REQ
;###############################################################################################
```

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

```
;===============================================================================
;==================== place your own code here for initialization
;===============================================================================
; The subroutine is to configure a fixed value for the ZCD_FHL_U and  ZCD_FHL_D bits.
; These First Hit Level (FHL) values are specific to the hardware.
; Inputs        :   -
; Output        :   -
; (total)unused : (all used)
; Temporary RAM : -
; Permanent RAM : -
; Routines used : -
;===============================================================================

MK_FW_INIT:

    jsub        ROM_USER_RAM_INIT       ; Initialising all USER RAM cells to 0

    ramadr      FWD_R1_FHL_VALUE        ;---- Value for the first hit level
    move        x, r

    ramadr      SHR_ZCD_FHL_U           ; Setting the same FHL for Up and Down
    move        r, x
    ramadr      SHR_ZCD_FHL_D
    move        r, x

MK_FW_INIT_END:

    ramadr      SHR_CPU_REQ             ; Set RAM Address to SHR_CPU_REQ
    bitclr      r,  BNR_FWI             ; Clear Firmware Init Flag

    jsubret                             ; Jump Back

;###############################################################################
;      End of Subroutine MK_FW_INIT
;###############################################################################
```

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

```
;=============================================================================================
;===================== place your own code here for error handling
;=============================================================================================
; The subroutine is not active yet and just clears the error flag register
; Inputs        :   -
; Output        :   -
; (total)unused : (all used)
; Temporary RAM : -
; Permanent RAM : -
; Routines used : -
;=============================================================================================

MK_OWN_EH:

; ------------------- Writes into RAM (for debugging)
    move y, DEBUG_CONTENT_01
    ramadr DEBUG_ADDR_01
    move r, y
; --------------------

    ;############ Place your code here for error handling ############
    nop                             ;--
;   nop                             ;--
;   nop                             ;--
;   nop                             ;--
    ;#################################################################

MK_OWN_EH_END:

    ramadr      SHR_EXC                 ; Set RAM Address to SHR_EXC
    bitset      r,  BNR_EF_CLR          ; clear error flag register

    jsubret

;#############################################################################################
;       End of Subroutine MK_OWN_EH
;#############################################################################################
```

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.

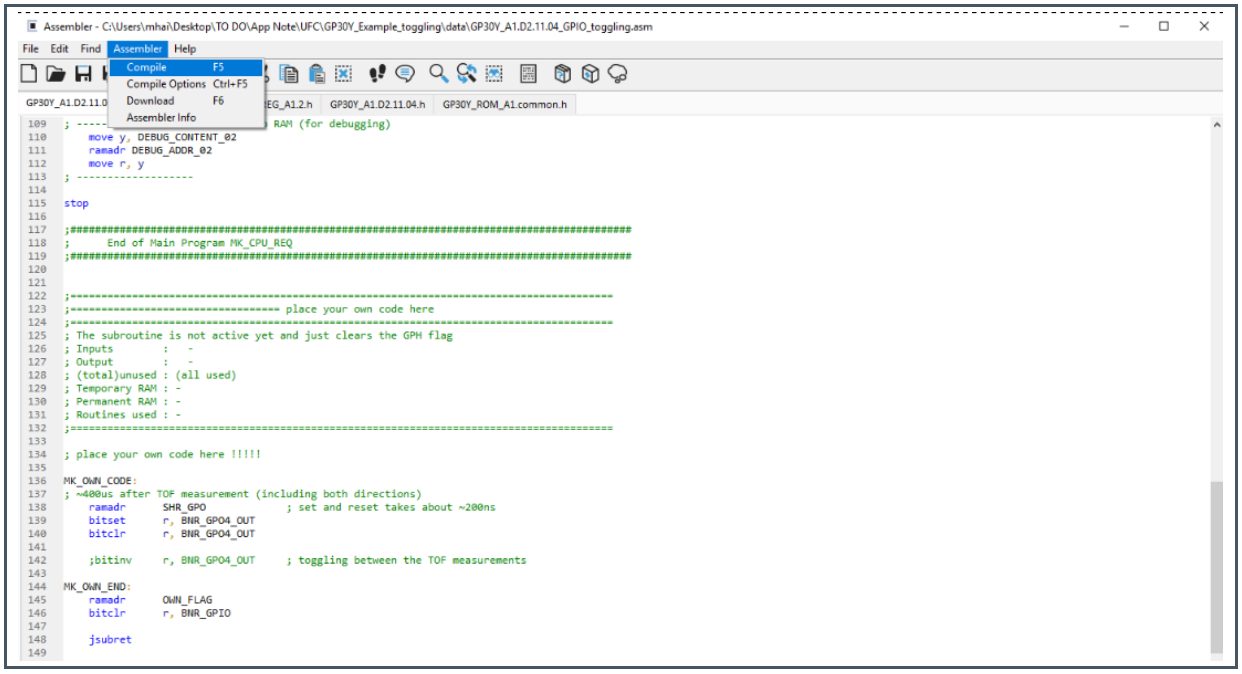
Figure 9 :
Main Custom Subroutine

```
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_GPO             ; set and reset takes about ~200ns
173     bitset      r, BNR_GPO4_OUT
174     bitclr      r, BNR_GPO4_OUT
175
176     ;bitinv     r, BNR_GPO4_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

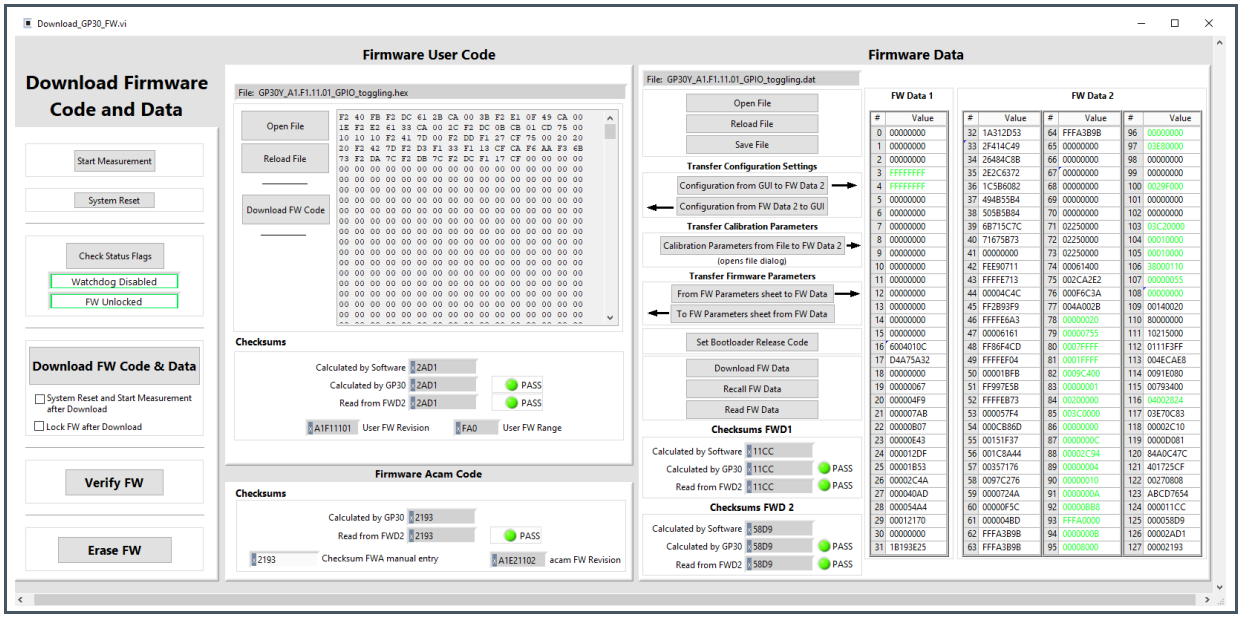
Be sure that the TDC-GP30 is idle.

- 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

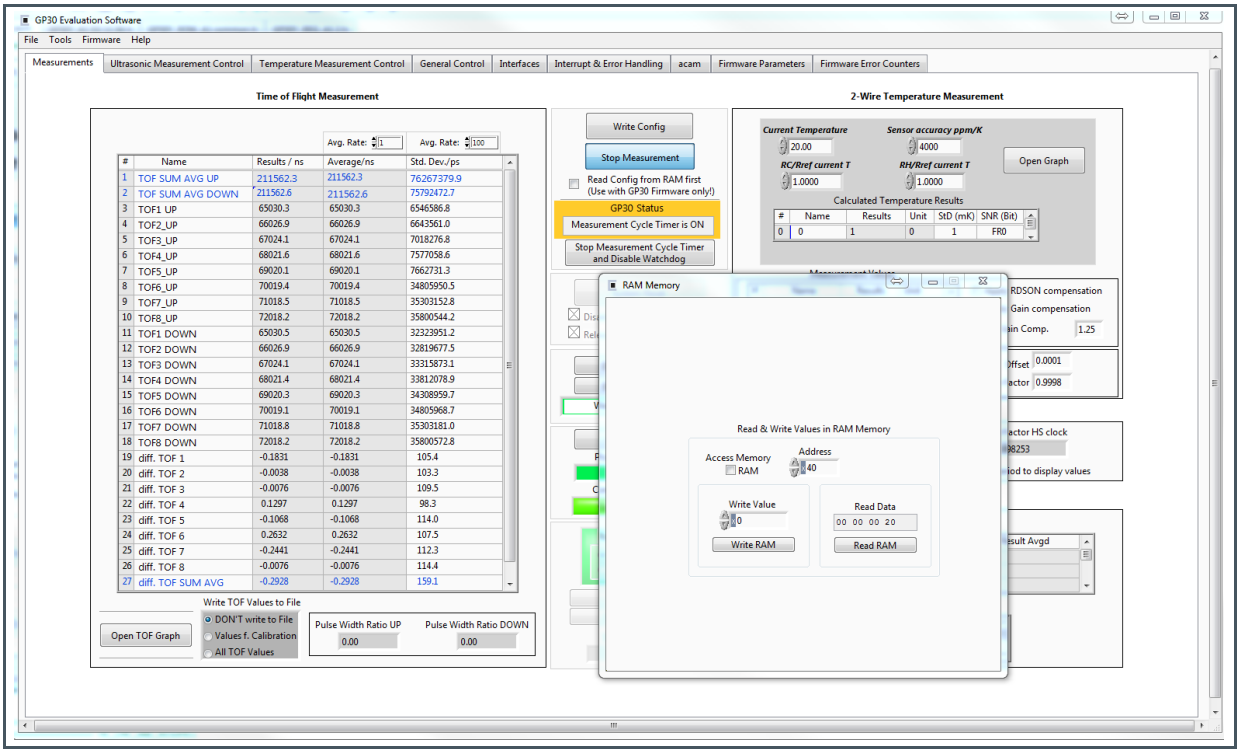


4 Summary / Results

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.

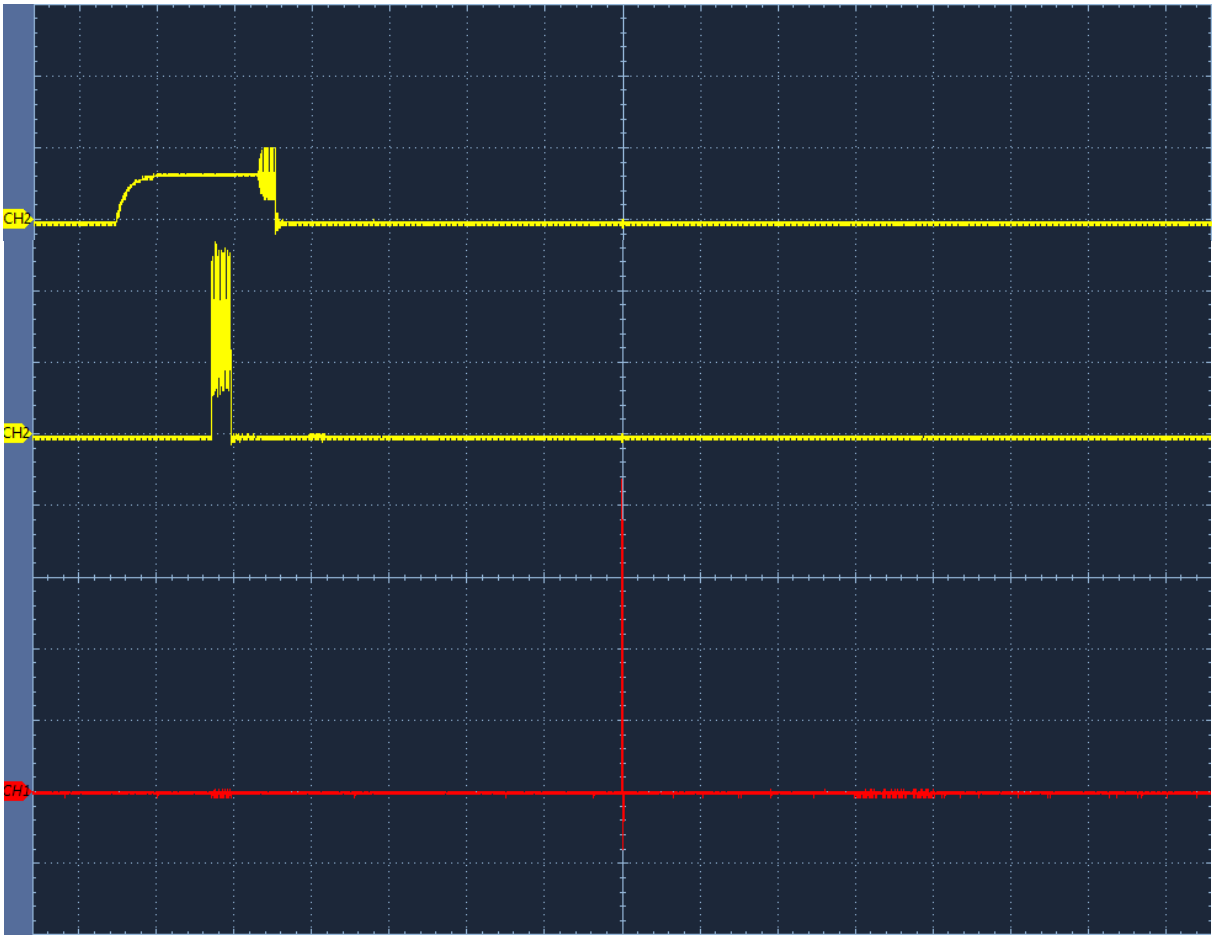
Figure 12 :
RAM Window



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 µs/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

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