STEVAL-MKI109V3 Professional MEMS Tool motherboard for MEMS adapter boards

Introduction

The STEVAL-MKI109V3 motherboard provides users with a complete, ready-to-use platform for the evaluation of STMicroelectronics MEMS products.

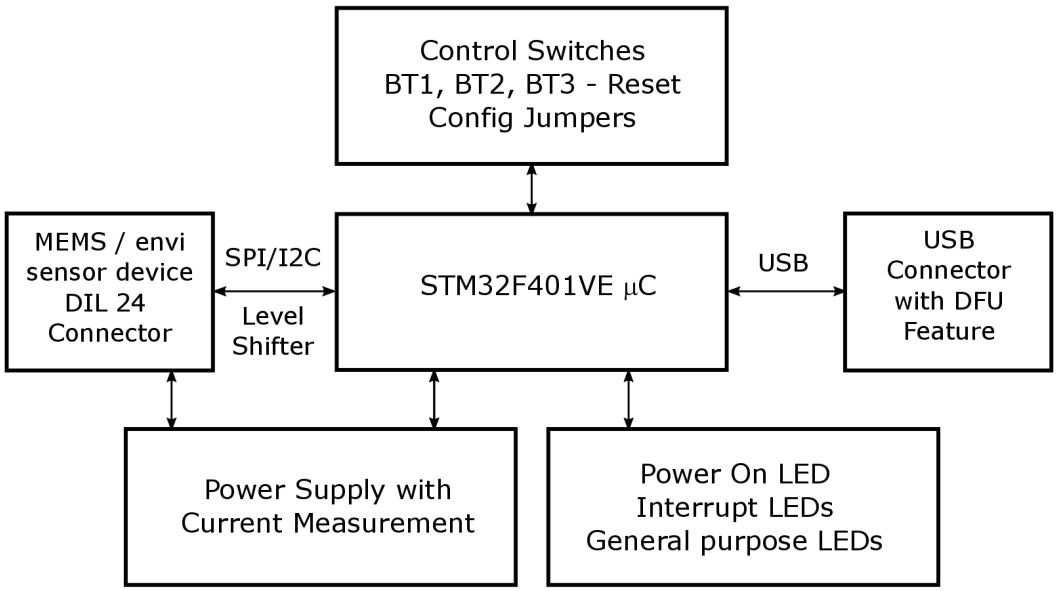
It includes a high-performance 32-bit microcontroller which functions as a bridge between the sensors and a PC, on which you can download and run the graphical user interface (GUI) or dedicated software routines for customized applications.

The board features a DIL24 socket to mount all available adapters for both digital and analog output MEMS devices.

1 Demonstration kit description

The Professional MEMS Tool is a complete demonstration kit for digital and analog MEMS sensors. Thanks to its DIL24 connector, a wide range of MEMS adapter boards can be used.

Figure 1. Demonstration board block diagram



The Professional MEMS Tool demonstration kit is based on the STM32F401VE microcontroller and can be connected to a PC via USB. Data from MEMS sensors connected to the board can be read through the PC GUI provided with the kit.

The Professional MEMS Tool also implements the device firmware upgrade (DFU) feature, so it can be reprogrammed with a new firmware release without the need to use a programmer (see www.st.com/mems).

The Professional MEMS Tool integrates:

- Six LEDs:
  - two LEDs connected via FET buffers to the interrupt pins of digital adapters
  - a power/USB LED
  - three general-purpose LEDs for firmware state indication
- Three buttons:
  - two user buttons on a dedicated GPIO of the microcontroller
  - a microcontroller reset button

All the MEMS adapter pins are available on board connectors J1 and J3.

Figure 2. **Top silkscreen of the Professional MEMS Tool kit**

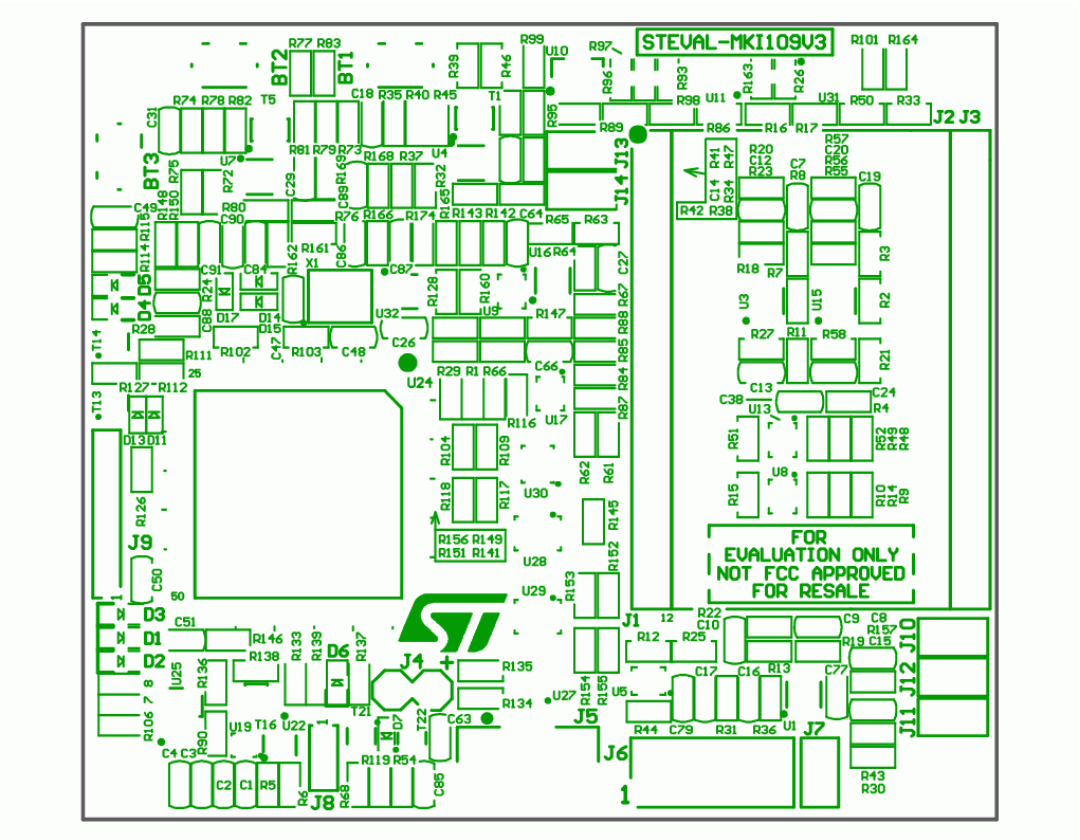


Figure 3. Top view of Professional MEMS Tool kit

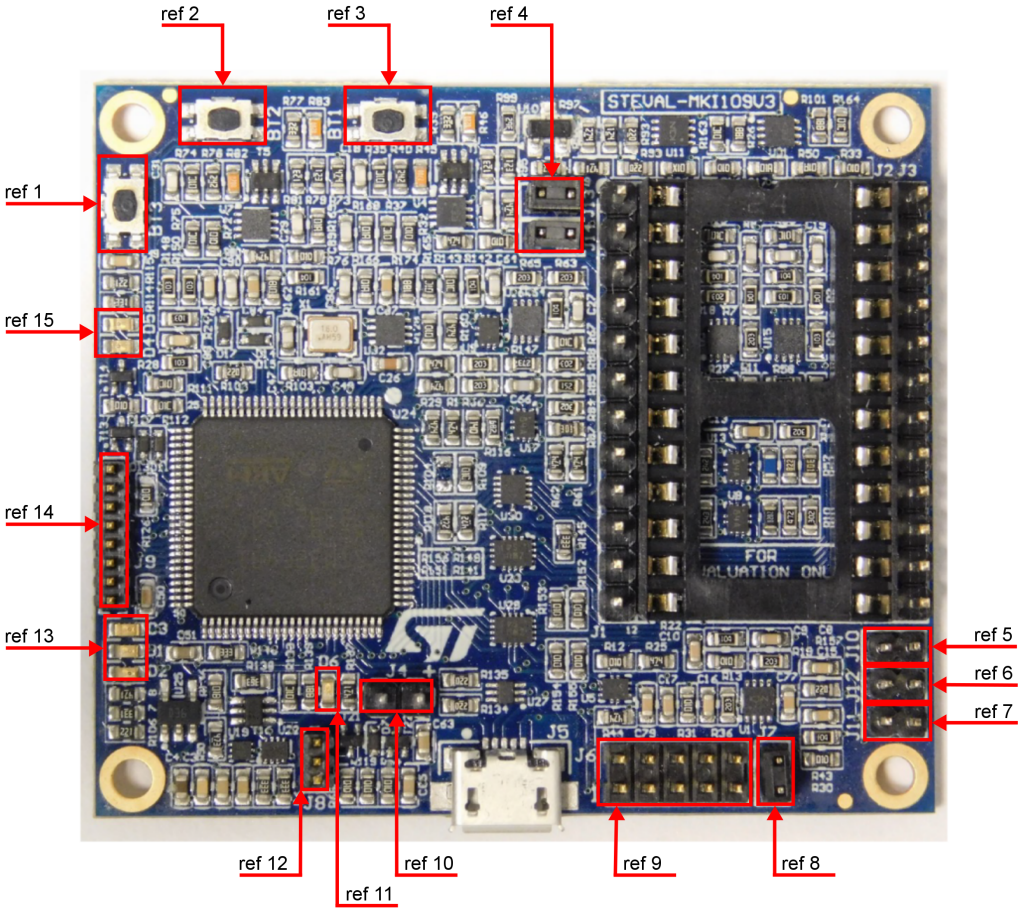


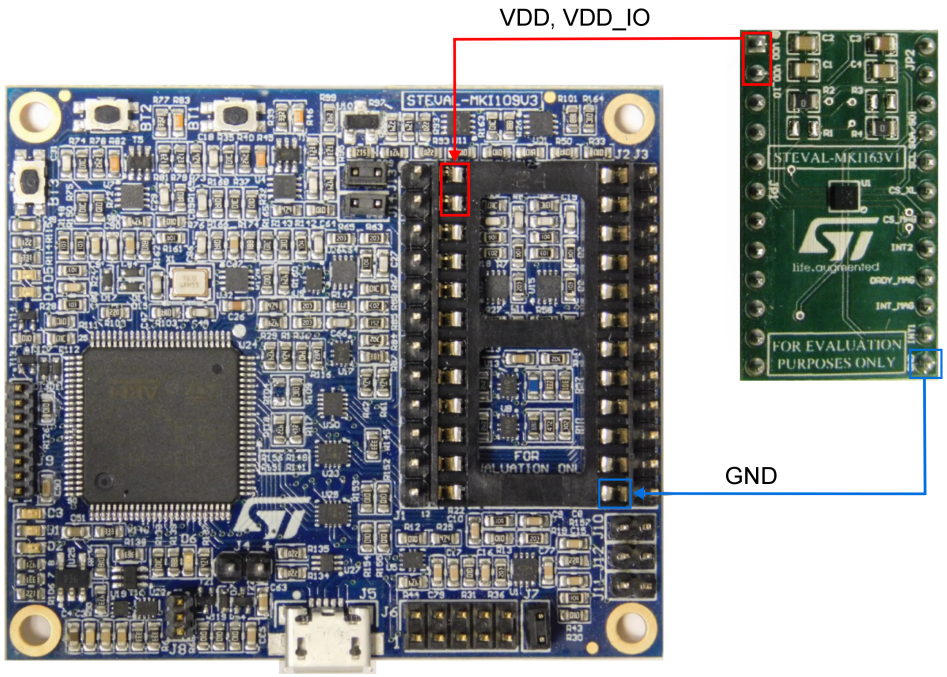
Figure 3 highlights some of the main components on the top layer of the Professional MEMS Tool kit.

1. Button BT3 is used to reset the STM32.
2. Button BT2 connected to STM32 GPIOs and available to the user. To enter DFU mode:
   a. press buttons BT3 (Reset) and BT2 together
   b. first release BT3 and then release BT2
3. BT1 connected to STM32 GPIOs and available to the user.
4. Jumpers J13 (VDD) and J14 (VDDIO) allow the user to measure the sensor current consumption by connecting a multimeter in series with their terminals.
5. Jumper J10 is used as a general purpose input to manually set certain features for several MEMS adapters.
6. Jumper J12 is used as a general purpose input to manually set certain features for several MEMS adapters.
7. Jumper J11 is used to set the self-test feature during testing of Professional MEMS Tool PCB.
8. Jumper J7 is used to select either JTAG (JP7 open – NRST control not allowed from programming connector J6) or SWD mode (JP7 shorted – NRST control allowed from connector J6).
9. J6 connector can be used to reprogram the STM32 and debug the code through the JTAG or SWD protocols.
10. Jumper J4 can be used to directly supply the board (from 4.5 V to 5.5 V) instead of through the USB connector.
11. LED D6 lights up when the board is powered.
12. J8 connector can be used for UART RX/TX communication.

13. LEDs D1, D2, and D3 are general-purpose LEDs used to indicate firmware states; e.g.:
    a. LED D3 YELOW light up when specific firmware is selected from those available
    b. LED D2 RED on indicates that the microcontroller is properly configured for communication with the sensor
    c. LED D1 GREEN blinks according to the sensor data rate selected
14. J9 connector can be used for general purpose SPI bus.
15. LEDs D4 and D5 are directly connected to the interrupt pins of the MEMS digital adapters (if available on the sensor mounted on the adapter board).

Figure 4. How to plug the DIL24 adapter on STEVAL-MKI109V3 shows how to plug the DIL 24 adapter MEMS module on the Professional MEMS Tool. VDD and VDDIO are in the top left corner (pins 1 and 2) and GND is in the bottom right corner (pin 13).

Figure 4. How to plug the DIL24 adapter on STEVAL-MKI109V3



2 Professional MEMS Tool board installation

Professional MEMS Tool board can be used with Unico-GUI graphical user interface.

The Unico-GUI package (.zip) downloaded from st.com contains a "FIRMWARE" folder with .dfu/.bin files for firmware upgrade. Similarly in MEMS Studio a proper firmware must be used to program the motherboard (firmware\mems-studio\).

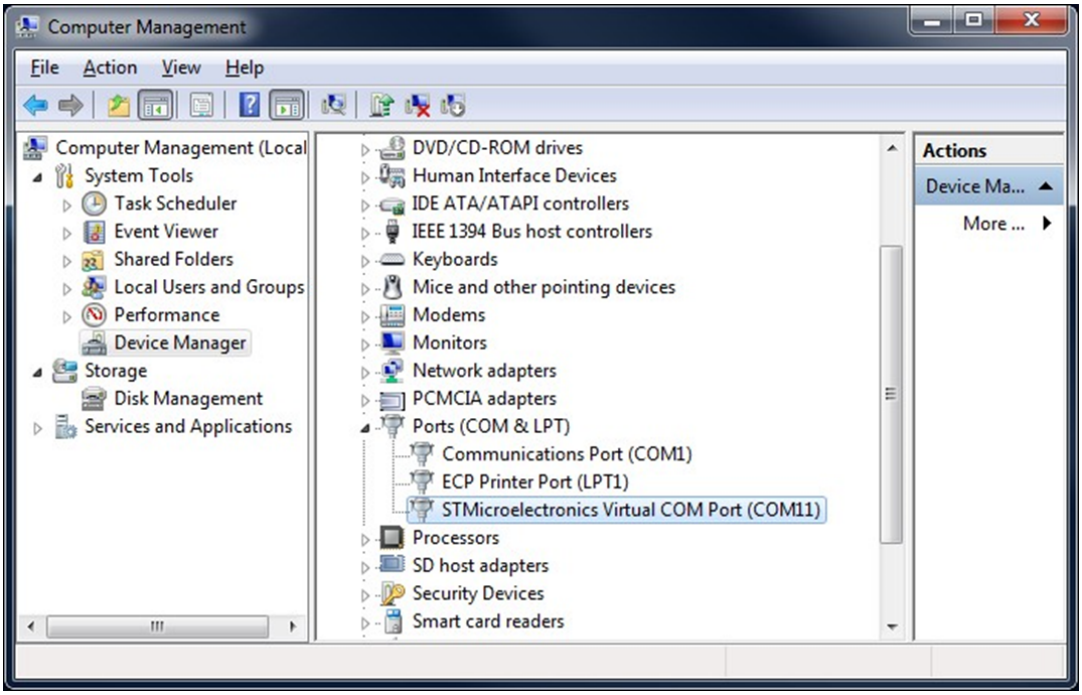
2.1 Hardware installation

- For Windows 10, Linux® and Mac OS® platforms, no driver installation is required.
- For older Windows platforms, install the STM32 virtual COM port driver from the STSW-STM32102 package.

Once the driver is installed, connect the demonstration kit board to a free USB port. A confirmation message should appear.

Note: On Windows 7 platform, confirm which COM port has been assigned to the board: right click on My Computer and select Manage, then select Device Manager and scroll through the list to Ports (COM & LPT). The STM32 virtual COM port driver for Windows 7 platform and related documents are packaged with the STSW-STM32102 software downloadable at www.st.com.

Figure 5. Virtual COM port assignment



2.2 Firmware upgrade

You can reprogram the STEVAL-MKI109V3 MEMS evaluation board both by the ST-LINK hardware programmer using J6 connector, or by simply connecting to the USB connector using the STM32CubeProgrammer software and using the DFU mode of the evaluation board. This is in accordance with the DFU class specification defined by the USB Implementers Forum.

The following steps show the most simple way using the STM32CubeProgrammer software and the DFU mode of STEVAL-MKI109V3. The input files can be both *.bin or *.dfu for the STM32CubeProgrammer.

*Note: The STM32CubeProgrammer software uses *.bin source files for firmware upgrade.*

Note: In case you have installed older DfuSe software on your machine, you must uninstall it before installing the STM32CubeProgrammer software.

For all details on the use of STM32CubeProgrammer software, refer to user manual UM2237 on www.st.com.

2.2.1 Entering DFU mode

The direct reprogramming of the microcontroller is particularly suited to USB applications where the same USB connector can be used both for the standard operating mode and the reprogramming process.

To configure the Professional MEMS Tool board in DFU mode:

- press button BT2 before supplying the board and release it when LED D6 lights up
- or
  1. press BT3 (Reset) and BT2 together
  2. release BT3 followed by BT2

Led D6 lights up and the device should appear in the Windows Device manager as “STM device in DFU mode”.

It is necessary to switch the Professional MEMS Tool board in DFU mode before executing the firmware upgrade procedures described in the following sections.

2.2.2 Firmware upgrade on Windows with STM32CubeProgrammer using *.bin source files

To upgrade the firmware by using the *.bin source files, follow the steps below.

Step 1. Attach the STEVAL-MKI109V3 MEMS board to the USB port while you are holding the BT2 button on it (procedure described in Section 2.2.1).

The STEVAL-MKI109V3 becomes then visible in your system as a device in DFU mode.

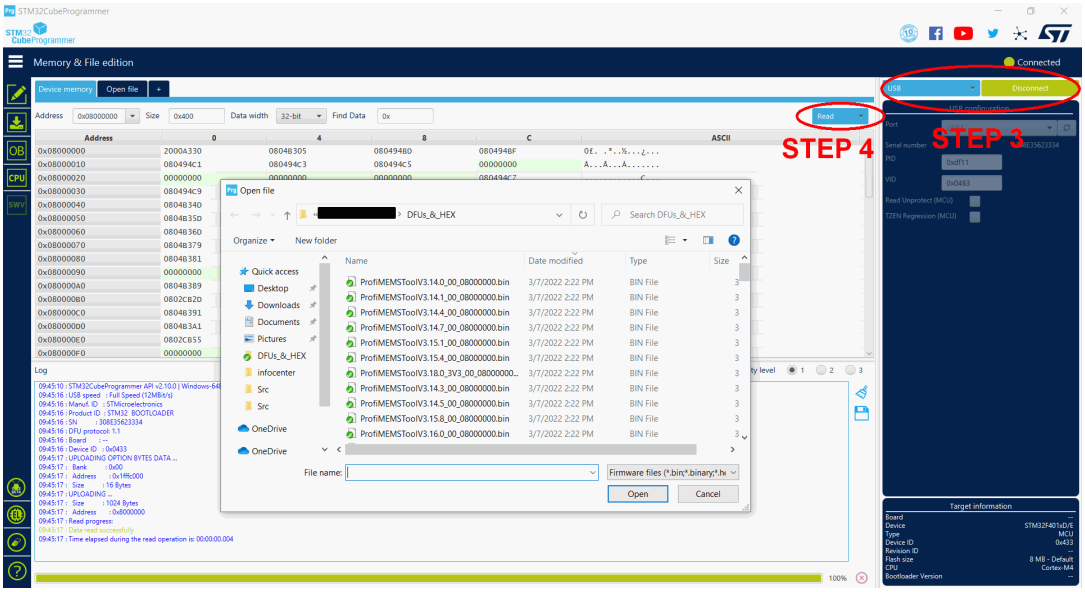
Step 2. Open the STM32CubeProgrammer software application.

Step 3. Select [**USB**] at the top-right corner roller blue button and then click on the [**Connect**] green button at the right side.

In the log window, the “Data read successfully” message should appear. The connection state indicator should show the “Connected” green state.

Step 4. Click on [**Open File**] and select the proper *.bin file that you want to download into the STEVAL-MKI109V3.

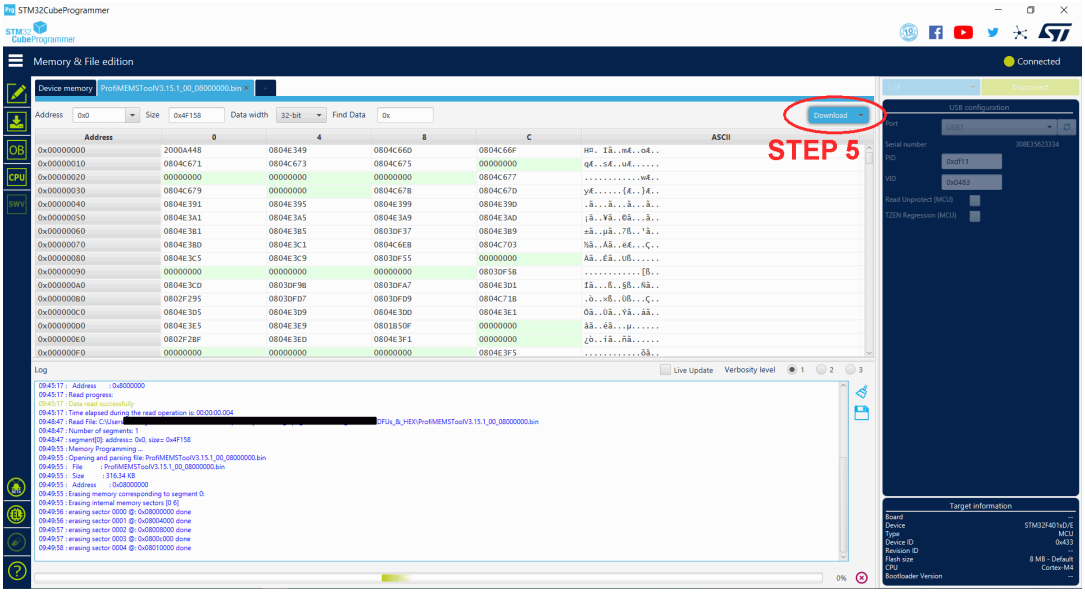
Figure 6. Opening the selected .bin file in STM32CubeProgrammer



Step 5. After choosing the tab with the selected .bin file, click on the [**Download**] blue button at the top right (there you can check the proper starting address "0x08000000").

The programming process then starts.

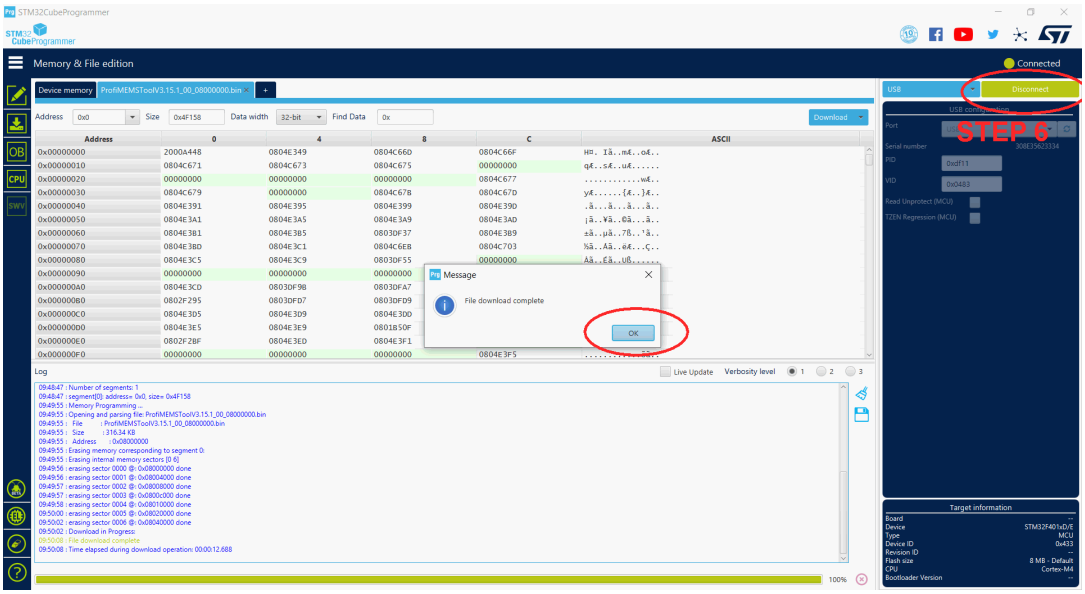
Figure 7. Programming process in STM32CubeProgrammer



Step 6. After the successful update of the STEVAL-MKI109V3, the “File download complete” message should appear.

The firmware update is then complete.
Click on the [**OK**] button and then click on the [**Disconnect**] green button at the top right.

Figure 8. File download complete message after successful firmware update



2.2.3 DFU on Windows using the older DfuSe software

To upgrade the firmware by using the *.dfu source files, follow the steps below.

Step 1. To install the DFU software, launch the “DfuSe_Demo_V3.0.5_Setup.exe” included in the software package and follow the instructions on the screen.

Step 2. Attach the STEVAL-MKI109V3 MEMS board to the USB port while you are holding the BT2 button on it (procedure described in Section 2.2.1).

The STEVAL-MKI109V3 becomes then visible in your system as a device in DFU mode.

Step 3. Launch newly installed DfuSeDemo software by selecting [**Start**]>[**STMicroelectronics**]>[**DfuSe**]>[**DfuSeDemo**]. (The software is located at C:\Program Files (x86)\STMicroelectronics\Software\DfuSe v3.0.5\Bin\DfuSeDemo.exe").

Step 4. In the [**Upgrade or Verify Action**] section of the DfuseDemo too, click on the [**Choose...**] button and select the target.dfu file; then, click on the [**Upgrade**] button to start the firmware upgrade.

For more details regarding the DFU and the microcontroller ST GUI, see the related user manual located at C:\Program Files (x86)\STMicroelectronics\Software\DfuSe v3.0.5\Bin\Doc\UM0412.pdf or downloadable from [**Start**]>[**STMicroelectronics**]>[**DfuSe**]>[**Docs**]>[**UM0412.pdf**].

Note: You can find the DFU utility tool and the related documents at the related ST web page.

2.2.4 DFU on Linux®

The DFU program for Linux operating systems is dfu-util. The procedure for Ubuntu Linux operating systems is described below.

Step 1. Open a terminal and run (with sudo to ensure the correct permissions):

```
sudo apt-get install dfu-util
```

Step 2. Create a udev rules file:

```
sudo gedit /etc/udev/45-Professional MEMS Tool.rules
```

Step 3. Fill it with the following content:

```
# 0483:5740 - STM32F4 in USB Serial Mode (CN5)
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="5740",
ENV{ID_MM_DEVICE_IGNORE}="1"
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="5740",
ENV{MTP_NO_PROBE}="1" SUBSYSTEMS=="usb",
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="5740",
MODE:="0666"
KERNEL=="ttyACM*", ATTRS{idVendor}=="0483",
ATTRS{idProduct}=="5740", MODE:="0666"
# 0483:df11 - STM32F4 in DFU mode (CN5) SUBSYSTEMS=="usb",
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="df11", MODE:="0666"
```

Step 4. Instruct udev to reload its rules:

```
sudo udevadm control --reload-rules
```

You should now be able to program the board.

Step 5. Attach the STEVAL-MKI109V3 MEMS board to the USB port while you are holding the BT2 button on it (procedure described in Section 2.2.1). Then, run the command:

```
sudo dfu-util -a 0 -D dfu_path/file.dfu -d 0483:df11
```

where:

– dfu_path is the path to the dfu file
– file.dfu is the dfu file name

example: `sudo dfu-util -a 0 -D Desktop/Professional MEMS ToolV2_REL_4_0.dfu -d 0483:df11`.

Step 6. Disconnect and reconnect the board to exit DFU mode and start using the board with the new firmware.

2.2.5 DFU on Mac OS®

The DFU program used for Mac operating systems is dfu-util.

Step 1. Before installing DFU on your Mac OS, you need to install Homebrew. Open a terminal and run:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/
install/master/install)"
```

Step 2. Install dfu-utils:

```
brew install dfu-util
```

You should now be able to program the board

Step 3. Attach the STEVAL-MKI109V3 MEMS board to the USB port while you are holding the BT2 button on it (procedure described in Section 2.2.1). Then, run the command:

```
dfu-util -a 0 -D dfu_path/file.dfu -d 0483:0000
```

where:

– dfu_path is the path to the dfu file

– file.dfu is the dfu file name

example: `dfu-util -a 0 -D Desktop/Professional MEMS ToolV2_REL_4_0.dfu -d0483:0000`

Step 4. Disconnect and reconnect the board to exit DFU mode and start using the board with the new firmware.

3 Supported MEMS adapter boards

Table 1. **List of supported MEMS adapter boards**

Adapter board	Device	*setb command string
STEVAL-MET001V1	LPS22HB	001V1
STEVAL-MKI087V1	LIS331DL	087V1
STEVAL-MKI089V1	LIS331DLH	089V1
STEVAL-MKI092V1	LIS331HH	092V1
STEVAL-MKI105V1	LIS3DH	105V1
STEVAL-MKI106V1	LSM303DLHC	106V1
STEVAL-MKI107V1	L3G4200D	107V1
STEVAL-MKI107V2	L3GD20	107V2
STEVAL-MKI108V2	9AXISMODULE v2 [LSM303DLHC + L3GD20]	108V2
STEVAL-MKI110V1	AIS328DQ	110V1
STEVAL-MKI122V1	LSM330DLC	122V1
STEVAL-MKI125V1	A3G4250D	125V1
STEVAL-MKI134V1	LIS3DSH	134V1
STEVAL-MKI135V1	LIS2DH	135V1
STEVAL-MKI136V1	L3GD20H	136V1
STEVAL-MKI137V1	LIS3MDL	137V1
STEVAL-MKI141V1	HTS221	141V1
STEVAL-MKI142V1	LPS25H	142V1
STEVAL-MKI151V1	LIS2DH12	151V1
STEVAL-MKI154V1	LSM9DS0	154V1
STEVAL-MKI158V1	AIS3624DQ	158V1
STEVAL-MKI159V1	LSM9DS1	159V1
STEVAL-MKI160V1	LSM6DS3	160V1
STEVAL-MKI161V1	LSM6DS0	161V1
STEVAL-MKI163V1	LSM303C	163V1
STEVAL-MKI164V1	LIS2HH12	164V1
STEVAL-MKI165V1	LPS25HB	165V1
STEVAL-MKI166V1	H3LIS100DL	166V1
STEVAL-MKI167V1	H3LIS200DL	167V1
STEVAL-MKI168V1	IIS2DH	168V1
STEVAL-MKI169V1	I3G4250D	169V1
STEVAL-MKI170V1	IIS328DQ	170V1
STEVAL-MKI172V1	LSM303AGR	172V1
STEVAL-MKI173V1	LSM303AH	173V1
STEVAL-MKI174V1	LIS2DS12	174V1
STEVAL-MKI175V1	LIS2DE12	175V1
STEVAL-MKI176V1	LSM6DS3H	176V1
STEVAL-MKI177V1	LPS35HW	177V1

Adapter board	Device	*setb command string
STEVAL-MKI178V1	LSM6DSL	178V1
STEVAL-MKI178V2	LSM6DSL	178V2
STEVAL-MKI179V1	LIS2DW12	179V1
STEVAL-MKI180V1	LIS3DHH	180V1
STEVAL-MKI181V1	LIS2MDL	181V1
STEVAL-MKI182V1	ISM330DLC	182V1
STEVAL-MKI183V1	LPS33HW	183V1
STEVAL-MKI184V1	ISM303DAC	184V1
STEVAL-MKI185V1	IIS2MDC	185V1
STEVAL-MKI186V1	IIS3DHHC	186V1
STEVAL-MKI188V1	L20G20IS	188V1
STEVAL-MKI189V1	LSM6DSM	189V1
STEVAL-MKI190V1	LIS2DTW12	190V1
STEVAL-MKI191V1	IIS2DLPC	191V1
STEVAL-MKI192V1	LPS22HH	192V1
STEVAL-MKI193V1	ASM330LHH	193V1
STEVAL-MKI194V1	LSM6DSR	194V1
STEVAL-MKI195V1	LSM6DSRX	195V1
STEVAL-MKI196V1	LSM6DSO	196V1
STEVAL-MKI197V1	LSM6DSOX	197V1
STEVAL-MKI198V1K	STTS751	198V1K
STEVAL-MKI199V1K	STLM20	199V1K
STEVAL-MKI200V1K	STTS22H	200V1K
STEVAL-MKI201V1K	STTS75	201V1K
STEVAL-MKI202V1K	STDS75	202V1K
STEVAL-MKI203V1K	STCN75	203V1K
STEVAL-MKI204V1K	STLM75	204V1K
STEVAL-MKI205V1	LPS33W	205V1
STEVAL-MKI206V1	AIS2DW12	206V1
STEVAL-MKI207V1	ISM330DHCX	207V1
STEVAL-MKI208V1K	IIS3DWB	208V1K
STEVAL-MKI209V1K	IIS2ICLX	209V1K
STEVAL-MKI210V1K	ISM330DHCX	210V1K
STEVAL-MKI211V1K	LIS25BA	211V1K
STEVAL-MKI212V1	ASM330LHHX	212V1
STEVAL-MKI213V1	LPS27HHW	213V1
STEVAL-MKI214V1	LPS33K	214V1
STEVAL-MKI215V1	LSM6DSO32	215V1
STEVAL-MKI216V1	IIS3DHHC	216V1
STEVAL-MKI217V1	LSM6DSOX+LIS2MDL	217V1
STEVAL-MKI218V1	AIS2IH	218V1

Adapter board	Device	*setb command string
STEVAL-MKI219V1	LPS22CH	219V1
STEVAL-MKI220V1	LPS27HHTW	220V1
STEVAL-MKI221V1	LSM6DSO32X	221V1
STEVAL-MKI222V1	LISDU12	222V1
STEVAL-MKI223A	ILPS28QSW	223A
STEVAL-MKI224V1	LPS22DF	224V1
STEVAL-MKI225A	LPS28DFW	225A
STEVAL-MKI226KA	AIS25BA	226KA
STEVAL-MKI227KA	LSM6DSV16X	227KA
STEVAL-MKI228KA	ILPS22QS	228KA
STEVAL-MKI229A	LSM6DSO16IS	229A
STEVAL-MKI230KA	ISM330IS	230KA
STEVAL-MKI231KA	STHS34PF80	231KA
STEVAL-MKI232A	LSM6DSO16ISN	232A
STEVAL-MKI233KA	ISM330ISN	233KA
STEVAL-MKI234KA	LSM6DSV16BX	234KA
STEVAL-MKI235KA	LIS2DUXS12	235KA
STEVAL-MKI236A	2x ASM330LHB	236A
STEVAL-MKI237KA	LSM6DSV16BX	237KA
STEVAL-MKI238A	LIS2DUX12	238A
STEVAL-MKI239A	LSM6DSV	239A
STEVAL-MKI241KA	LSM6DSV16B	241KA
STEVAL-MKI242A	ST1VFE6AX	242A
STEVAL-MKI243A	ASM330LHHXG1	243A

4 Supported commands

The microcontroller mounted on the Professional MEMS Tool board is equipped with dedicated firmware that allows control of the digital output MEMS sensor, and acquisition of the measured data.

The firmware also handles the communication between the board and the PC through the USB bus.

4.1 Getting started

Before using the commands supported by the firmware, the following procedure must be performed:

Step 1. Connect the Professional MEMS Tool to the USB port

Step 2. Launch an application that allows sending commands through the virtual serial port. The remainder of this document assumes the use of Microsoft® HyperTerminal program available with the Windows XP operating system, but you can use any similar tool.

Step 3. Create a new connection, enter a name (e.g. STEVAL-MKI109V3), and click `OK`.

Step 4. In the `Connect Using` field, select the virtual COM port to which the USB port has been mapped, and click `OK`.

Step 5. In port settings, set `bits per second` to 115200, `data bits` to 8, `parity` to none, `stop bits` to 1, and `flow control` to none; click `OK`

Step 6. In the HyperTerminal application window, select `files > properties > settings`, then click `ASCII Setup`.

Step 7. Select `Send line ends with line feeds` and `Echo typed characters locally`

Step 8. Click `OK` to close the ASCII Setup window

Step 9. Click `OK` button to close the Properties window.

Once this procedure has been completed you can use the commands described in the following sections by typing them in the “HyperTerminal” window.

4.1.1 Quick start

The basic sequence of commands (based on the LIS3DH accelerometer) to start a data communication session and to retrieve X, Y, and Z acceleration data from the demonstration kit is:

Step 1. Connect the Professional MEMS Tool to the USB port

Step 2. Start “Microsoft© HyperTerminal” (or another similar application) and configure it as described in Section 4.1: Getting started

Step 3. Enter the `*setdb105v1` command in the HyperTerminal” window, (supposing the LIS3DH adapter board is used – for other adapters see the relevant datasheets to check the register configuration), enter the command `* Zoff` to enable the control of the device by the STM32F401VE microcontroller, and `*w2047` to switch on the LIS3DH and to set the data rate to 50 Hz

Step 4. Send the `*debug` command to get the X, Y, and Z data measured by the sensor

Step 5. Send `*stop` to end the continuous acquisition and visualization.

4.2 Supported commands

The firmware supports a wide range of MEMS adapters; the complete list of supported commands and their descriptions are given below. Commands are not case sensitive.

Table 2. List of supported commands

Command	Description	Returned value(1)
*setdbXXXVY	Selects firmware according to the adapter connected	Device name e.g.: LIS3DH
*start	Starts continuous data acquisition	(see Table 4. Returned values for *start command)
*debug	Returns the output data in readable text format	
*stop	Stops data acquisition	
*zon	Forces High impedance state. Turns off VDD and VDDIO power supply.	
*zoff	Exits from High impedance state. Turns on VDD and VDDIO power supply	
*dev	Device name	e.g.: LIS3DH
*ver	Firmware version	e.g.: V1.5.2
*board	Returns board name	
*rAA	Accelerometer register read	e.g.: RAAhDDh
*wAADD	Accelerometer register write	
*grAA	Gyroscope register read	e.g.: GRAAhDDh
*gwAADD	Gyroscope register write	
*mrAA	Magnetometer register read	e.g.: MRAAhDDh
*mwAADD	Magnetometer register write	
*prAAx	Pressure sensor register read	e.g.: PRAAhDDh
*pwAADD	Pressure sensor register write	
*hrAA	Humidity sensor register read	e.g.: HRAAhDDh
*hwAADD	Humidity sensor register write	
*trAA	Temperature sensorregister read	e.g.: TRAAhDDh
*twAADD	Temperaturesensor register write	
*single	It gets a single point data acquisition	(see Table 4. Returned values for *start command)
*sindebug	It gets a single point data acquisition in readable text	
*list	Prints the list of MKIs supported	e.g.: MKI001V1→LPS22HB MKI087V1→LIS331DL MKI089V1→ LIS331DLH
*listdev	Prints the list of devices supported	e.g.: LIS331DL LIS331DLH LIS331HH…
*echoon	Activates the write verbose mode	e.g.: RAAhDDh
*echooff	Deactivates the write verbose mode	
*fiforst	Accelerometer “Reset mode” enable	st XH XL YH YL ZH ZL IR FC FS
*fifomde	Accelerometer “FIFO mode” enable	st XH XL YH YL ZH ZL IR FC FS
*fifostr	Accelerometer “FIFO stream” enable	st XH XL YH YL ZH ZL IR FC FS
*fifostf	Accelerometer “Stream to FIFO” enable	st XH XL YH YL ZH ZL IR FC FS
*fifobtf	Accelerometer “Bypass to FIFO” enable	st XH XL YH YL ZH ZL IR FC FS

Command	Description	Returned value(1)
*fifobts	Accelerometer “Bypass to stream” enable	st XH XL YH YL ZH ZL IR FC FS
*fifodstr	Accelerometer “Dynamic stream” enable	st XH XL YH YL ZH ZL IR FC FS
*gfiforst	Gyroscope “Reset mode” enable	st XH XL YH YL ZH ZL IR FC FS
*gfifomde	Gyroscope “FIFO mode” enable	st XH XL YH YL ZH ZL IR FC FS
*gfifostr	Gyroscope “FIFO stream” enable	st XH XL YH YL ZH ZL IR FC FS
*gfifostf	Gyroscope “Stream to FIFO” enable	st XH XL YH YL ZH ZL IR FC FS
*gfifobtf	Gyroscope “Bypass to FIFO” enable	st XH XL YH YL ZH ZL IR FC FS
*gfifobts	Gyroscope “Bypass to stream” enable	st XH XL YH YL ZH ZL IR FC FS
*gfifodstr	Gyroscope “Dynamic stream” enable	st XH XL YH YL ZH ZL IR FC FS
*pfiforst	Pressure sensor “Reset mode” enable	st XH XL YH YL ZH ZL IR FC FS
*pfifomde	Pressure sensor “FIFO mode” enable	st XH XL YH YL ZH ZL IR FC FS
*pfifostr	Pressure sensor “FIFO stream” enable	st XH XL YH YL ZH ZL IR FC FS
*pfifostf	Pressure sensor “Stream to FIFO” enable	st XH XL YH YL ZH ZL IR FC FS
*pfifobtf	Pressure sensor “Bypass to FIFO” enable	st XH XL YH YL ZH ZL IR FC FS
*pfifobts	Pressure sensor “Bypass to stream” enable	st XH XL YH YL ZH ZL IR FC FS
*pfifodstr	Pressure sensor “Dynamic stream” enable	st XH XL YH YL ZH ZL IR FC FS
*rmAA_1NN	Multiple read of NN Accelerometer successive registers	RMAA_1hNNhDD$_1$hDD$_2$...DD$_{NN}$h
*mutli-rAA_1AA_2 AA_3…	Accelerometer registers multiple read	MULTI-RAA_1hDD$_1$h AA_2hDD$_2$h…. AAnDDnh
*grmAA_1NN	Multiple read of NN Gyroscope successive registers	GRMAA_1hNNhDD$_1$hDD$_2$...DD$_{NN}$h
*multi-grAA_1AA_2 AA_3…	Gyroscope registers multiple read	MULTI-GRAA_1hDD$_1$h AA_2hDD$_2$h…. AAnDDnh
*mrmAA_1NN	Multiple read of NN Magnetometer successive registers	MRMAA_1hNNhDD$_1$hDD$_2$...DD$_{NN}$h
*multi-mrAA_1AA_2 AA_3…	Magnetometer registers multiple read	MULTI-MRAA_1hDD$_1$h AA_2hDD$_2$h…. AAnDDnh
*prmAA_1NN	Multiple read of NN Pressure sensor successive registers	PRMAA_1hNNhDD$_1$hDD$_2$...DD$_{NN}$h
*multi-prAA_1AA_2 AA_3…	Pressure sensor registers multiple read	MULTI-PRAA_1hDD$_1$h AA_2hDD$_2$h…. AAnDDnh
*hrmAA_1NN	Multiple read of NN Humidity sensor successive registers	HRMAA_1hNNhDD$_1$hDD$_2$...DD$_{NN}$h
*multi-hrAA_1AA_2 AA_3…	Humidity sensor registers multiple read	MULTI-HRAA_1hDD$_1$h AA_2hDD$_2$h…. AAnDDnh
*trmAA_1NN	Multiple read of NN Temperature sensor successive registers	TRMAA_1hNNhDD$_1$hDD$_2$...DD$_{NN}$h
*multi-trAA_1AA_2 AA_3…	Temperature sensor registers multiple read	MULTI-TRAA_1hDD$_1$h AA_2hDD$_2$h…. AA_nDD$_n$h
*odr [param]	Sets ODR speed	

1. *RP: Reference pressure XLSB.MSB, IR: interrupt byte; FC: FIFO control register; FS: FIFO source register*

Table 3. **List of supported commands for power supply control and I/V measurement**

Command	Description	Returned value
*power_on	Turns on VDD and VDDIO power supply	
*power_off	Turns off VDD and VDDIO power supply	
*set_vddaV.VVV	Sets VDD voltage value “V.VVV” Volts. Value defined by 1 or 2 decimal places is possible. e.g. 1.8; 1.86; 1.825 are all valid inputs	
*set_vddioV.VVV	Sets VDDIO voltage value “V.VVV” Volts. Value defined by 1 or 2 decimal places is possible. e.g. 1.8; 1.86; 1.825 are all valid inputs	
*get_i [param]	Measures IDD, IDDIO with automatic / semiautomatic current ranges toggling and measurement method selection. Output values format selected by parameters [F / B / A] [S] [C] [L]	e.g. param = A ida Idd = 152 µA IddIO = 0.092 µA
*get_v [param]	Measures VDD, VDDIO Output values format selected by parameters [F/B/A]	e.g. param = A vda Vdd = 1.807 V VddIO = 1.806 V
*ranges_auto	Turns on automatic IDD/IDDIO ranges selection (default)	
*ranges_man	Turns off automatic IDD/IDDIO ranges selection (manual IDD/IDDIO ranges selection possible)	
*range_ma [param]	Selects current range for IDD	
*range_mb [param]	Selects current range for IDDIO	
*adc_run_time [param]	Sets time period during which ADC samples the measured current waveforms	
*adc_uni_stop	Stops all measurements	

The commands listed in Table 3 work regardless of whether an adapter is physically inserted in the DIL24 socket or not, and also whether the STEVAL-MKIxxxxx of the adapter is selected or not (using the *Setdb command).

4.2.1 *setdbXXXVY

This command selects the part of the firmware able to handle the adapter board sensor connected to the board. For example, *setdb105V1 selects the firmware for the LIS3DH.

The D3 LED (yellow) switches on automatically.

4.2.2 *start

This command initiates continuous data acquisition. When sent, the device returns a string of bytes (plus carriage return and line feed) like `st OUT1 OUT2 OUT3 IR STP BT`.

The first two bytes are always the ASCII char `s` and `t` which correspond to the hexadecimal values {73h 74h}.

`OUT1`, `OUT2`, and `OUT3` contain the values measured at device outputs; if the output data is represented in more than 8 bits, `OUT1`, `OUT2`, and `OUT3` are split into high byte (e.g., `XH`) and low byte (e.g., `XL`). In case of 24-bit resolution for some sensors, there is also an extra-low byte (e.g., pressure data: `PXL PL PH`).

`IR` (INT1 INT2) contains the interrupt bytes and `BT` SW1|SW2 contains the bytes that describe the state of the buttons integrated on the board.

Specifically, bit#0 of the SW1|SW2 data corresponds to the status of the SW1 button on the demonstration kit board: it is set to 1 when the SW1 is pressed (otherwise 0). Bit#1 has the same behavior but is dedicated to the SW2.

`STP` (STPL STPH) contains the step counter bytes for the internal device step counter value.

The string is ended with the carriage return (`\r`) and line feed (`\n`) bytes.

Before sending the `*start` command, the device must be out of 3-state (high impedance) and some registers must be configured according to user needs. Therefore, `*start` must be preceded by a `* zoff` and some `Register Write` commands.

As data is continuously acquired, LED D1 (green) blinks according to sensor data rate selected.
Table 4. Returned values for *start command shows the format of the string returned for each device when a `*start` command is sent. Similar byte strings are returned for groups of commands related to FIFO, as is shown in Table 5. Digital output accelerometers: supported commands list, Table 6. Digital output gyroscopes: supported commands list, Table 7. Digital output magnetometer: supported commands list, Table 8. Digital output pressure sensor: supported commands list, Table 9. Digital output humidity sensor: supported commands list, Table 10. Digital output temperature sensor: supported commands list and Table 11. Analog output temperature sensor: supported commands list.

Table 4. Returned values for *start command

STEVAL # (Device)	Returned value (1)
STEVAL-MKI089V1 (LIS331DLH) STEVAL-MKI092V1 (LIS331HH) STEVAL-MKI105V1 (LIS3DH) STEVAL-MKI110V1 (AIS328DQ) STEVAL-MKI125V1 (A3G4250D) STEVAL-MKI134V1 (LIS3DSH) STEVAL-MKI135V1 (LIS2DH) STEVAL-MKI136V1 (L3GD20H) STEVAL-MKI151V1 (LIS2DH12) STEVAL-MKI158V1 (AIS3624DQ) STEVAL-MKI164V1 (LIS2HH12) STEVAL-MKI168V1 (IIS2DH) STEVAL-MKI170V1 (IIS328DQ) STEVAL-MKI179V1 (LIS2DW12) STEVAL-MKI180V1 (LIS3DHH) STEVAL-MKI186V1 (IIS3DHHC) STEVAL-MKI191V1 (IIS2DLPC) STEVAL-MKI206V1 (AIS2DW12) STEVAL-MKI216V1 (IIS3DHHC) STEVAL-MKI218V1 (AIS2IH)	s t XH XL YH YL ZH ZL int1 int2 sw1|sw2 \r \n
STEVAL-MKI087V1 (LIS331DL) STEVAL-MKI175V1 (LIS2DE12)	s t X Y Z int1 int2 sw1|sw2 \r \n
STEVAL-MKI166V1 (H3LIS100DL) STEVAL-MKI167V1 (H3LIS200DL)	s t X 0 Y 0 Z 0 int1 int2 sw1|sw2 \r \n
STEVAL-MKI208V1K (IIS3DWB)	s t XH XL YH YL ZH ZL int1 sw1|sw2 \r \n
STEVAL-MKI174V1 (LIS2DS12) STEVAL-MKI176V1 (LSM6DS3H)	s t XH XL YH YL ZH ZL int1 int2 stpL stpH 0 sw1|sw2 \r \n
STEVAL-MKI137V1 (LIS3MDL) STEVAL-MKI181V1 (LIS2MDL) STEVAL-MKI185V1 (IIS2MDC)	s t M_XH M_XL M_YH M_YL M_ZH M_ZL int1 sw1|sw2 \r \n
STEVAL-MKI107V1 (L3G4200D) STEVAL-MKI107V2 (L3GD20) STEVAL-MKI169V2 (I3G4250D) STEVAL-MKI188V1 (L20G20IS)	s t G_XH G_XL G_YH G_YL G_ZH G_ZL G_int1 G_int2 sw1|sw2 \r \n
STEVAL-MKI122V1 (LSM330DLC)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZLA_int1 A_int2 G_int1 G_int2 sw1|sw2 \r \n
STEVAL-MKI106V1(LSM303DLHC)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL M_XH M_XL M_YH M_YL M_ZH M_ZL A_int1 A_int2 sw1|sw2 \r \n
STEVAL-MKI108V2 (9AXIS MODULE) STEVAL-MKI159V1 (LSM9DS1)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL M_XH M_XL M_YH M_YL M_ZH M_ZL A_int1 G_int2 G_int3 0 sw1|sw2 \r \n
STEVAL-MKI154V1 (LSM9DS0)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL M_XH M_XL M_YH M_YL M_ZH M_ZL A_int1 A_int2 sw1|sw2 \r \n
STEVAL-MKI159V1 (LSM9DS1)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL M_XH M_XL M_YH M_YL M_ZH M_ZL A_int1 A_int2 G_int3 0 sw1|sw2 \r \n

STEVAL # (Device)	Returned value [1]
STEVAL-MKI160V1 (LSM6DS3) STEVAL-MKI161V1 (LSM6DS0)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL Int1 Int2 sw1|sw2 \r \n
STEVAL-MKI178V1 (LSM6DSL) STEVAL-MKI178V2 (LSM6DSL) STEVAL-MKI182V1 (ISM330DLC) STEVAL-MKI189V1 (LSM6DSM)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL Int1 Int2 StpL StpH 0 sw1|sw2 \r \n
STEVAL-MKI163V1 (LSM303C)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL M_XH M_XL M_YH M_YL M_ZH M_ZL A_int G_int sw1|sw2 \r \n
STEVAL-MKI172V1 (LSM303AGR)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL M_XH M_XL M_YH M_YL M_ZH M_ZL A_int1 Aint2 M_int sw1|sw2 \r \n
STEVAL-MKI173V1 (LSM303AH) STEVAL-MKI184V1 (ISM303DAC)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL M_XH M_XL M_YH M_YL M_ZH M_ZL A_int1 A_int2 M_int StpL StpH sw1|sw2 \r \n
STEVAL-MKI142V1 (LPS25H) STEVAL-MKI165V1 (LPS25HB)	s t PXL PL PH TL TH REF_PXL REF_PL REF_PH P_int1 sw1|sw2 \r \n
STEVAL-MET001V1 (LPS22HB) STEVAL-MKI177V1 (LPS35HW) STEVAL-MKI183V1 (LPS33HW) STEVAL-MKI192V1 (LPS22HH) STEVAL-MKI205V1 (LPS33W) STEVAL-MKI213V1 (LPS27HHW) STEVAL-MKI214V1 (LPS33K) STEVAL-MKI219V1 (LPS22CH) STEVAL-MKI220V1 (LPS27HHTW) STEVAL-MKI224V1 (LPS22DF) STEVAL-MKI225A (LPS28DFW)	s t PXL PL PH TL TH REF_PXL REF_PL REF_PH P_int1 sw1|sw2 \r \n
STEVAL-MKI141V1 (HTS221)	s t HL HH TL TH H_int1 sw1|sw2 \r \n
STEVAL-MKI190V1 (LIS2DTW12) STEVAL-MKI222V1 (LISDU12) STEVAL-MKI235KA (LIS2DUXS12) STEVAL-MKI238A (LIS2DUX12)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL Int1 Int2 TL TH sw1|sw2 \r \n
STEVAL-MKI198V1K (STTS751)	s t TH TL Int1 sw1|sw2 \r \n
STEVAL-MKI199V1K (STLM20)	s t TH TL sw1|sw2 \r \n
STEVAL-MKI200V1K (STTS22H) STEVAL-MKI201V1K (STTS75) STEVAL-MKI202V1K (STDS75) STEVAL-MKI203V1K (STCN75) STEVAL-MKI204V1K (STLM75)	s t TH TL Int1 sw1|sw2 \r \n
STEVAL-MKI194V1 (LSM6DSR) STEVAL-MKI196V1 (LSM6DSO) STEVAL-MKI215V1 (LSM6DSO32) STEVAL-MKI221V1 (LSM6DSO32X)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL Int1 Int2 FSMLC_INT FSM1 FSM2 StpL StpH TL TH sw1|sw2 \r \n
STEVAL-MKI211V1K (LIS25BA) STEVAL-MKI226A (AIS25BA)	s t XH XL YH YL ZH ZL sw1|sw2 \r \n
STEVAL-MKI195V1 (LSM6DSRX) STEVAL-MKI197V1 (LSM6DSOX) STEVAL-MKI207V1 (ISM330DHCX) STEVAL-MKI210V1K (ISM330DHCX) STEVAL-MKI212V1 (ASM330LHHX) STEVAL-MKI243A (ASM330LHHXG1)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL Int1 Int2 FSMLC_INT FSM1 FSM2 StpL StpH MLC0 MLC1 MLC2 MLC3 MLC4 MLC5 MLC6 MLC7 TL TH sw1|sw2 \r \n
STEVAL-MKI209V1 (IIS2ICLX)	s t A_XH A_XL A_YH A_YL Int1 Int2 FSMLC_INT FSM1 FSM2 StpL StpH MLC0 MLC1 MLC2 MLC3 MLC4 MLC5 MLC6 MLC7 TL TH sw1|sw2 \r \n
STEVAL-MKI217V1 (LSM6DSOX + LIS2MDL hub)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL M_XH M_XL M_YH M_YL M_ZH M_ZL Int1 Int2 FSMLC_INT FSM1 FSM2 MGN_INT StpL StpH MLC0 MLC1 MLC2 MLC3 MLC4 MLC5 MLC6 MLC7 0 sw1|sw2 \r \n
STEVAL-MKI223A (ILPS28QSW) STEVAL-MKI228KA (ILPS22QS)	st PXL PL PH TL TH 0 REF_PL REF_PH SGNB QXL QL QH INT1 sw1|sw2 \r \n

STEVAL # (Device)	Returned value [1]
STEVAL-MKI193V1 (ASM330LHH) STEVAL-MKI229A (LSM6DSO16IS) STEVAL-MKI230KA (ISM330IS)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL Int1 Int2 TL TH sw1|sw2 \r \n
STEVAL-MKI231KA (STHS34PF80)	s t Tobj_H Tobj L Tobj_comp_H Tobj_comp_L Tambient_H Tambient_L Tpresence_H Tpresence_L Tmotion_H Tmotion L Tambient_shock_H Tambient_shock_L Int1 Int_status sw1|sw2 \r\n
STEVAL-MKI234KA (LSM6DSV16BX) STEVAL-MKI237KA (LSM6DSV16BX) STEVAL-MKI241KA (LSM6DSV16BX)	UI string: s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL QH QL Q_FH Q_FL A_Int1 A_Int2 FSMLC_INT FSM1 StpL StpH DT1 DT2 DT3 DT4 TL TH sw1|sw2 \r \n TDM string: t d A_XH A_XL A_YH A_YL A_ZH A_ZL 'T' \r \n
STEVAL-MKI242A (ST1VAFE6AX)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL QH QL Q_FH Q_FL A_Int1 A_Int2 FSMLC_INT FSM1 StpL StpH DT1 DT2 DT3 DT4 TL TH sw1|sw2 \r \n
STEVAL-MKI236A (2x ASM330LHB)	s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL A2_XH A2_XL A2_YH A2_YL A2_ZH A2_ZL 2G_XH G2_XL G2_YH G2_YL G2_ZH G2_ZL A_Int1 A_Int2 FSMLC_INT FSM1 FSM2 StpL StpH DT1 DT2 DT3 DT4 DT5 DT6 DT7 DT8 TL TH sw1|sw2 \r \n
STEVAL-MKI239A (LSM6DSV)	UI string: s t A_XH A_XL A_YH A_YL A_ZH A_ZL G_XH G_XL G_YH G_YL G_ZH G_ZL A_Int1 A_Int2 FSMLC_INT FSM1 StpL StpH DT1 DT2 DT3 DT4 TL TH sw1|sw2 'U' \r \n OIS string: s t A_XH A_XL A_YH A_YL A_ZH A_ZL 'O' \r \n
STEVAL-MKI232A (LSM6DSO16ISN) STEVAL-MKI233KA (ISM330ISN)	Output string is defined by Unico / MEMS studio. See proper documentation of device setting with .UCF file

1. *XH: X-axis output high byte (same for Y axis, Z axis, G gyroscope, M Magnetometer, P pressure, H humidity, T temperature, Stp step counter). XL: X-axis output low byte (same for Y axis, Z axis, G gyroscope, M Magnetometer, P pressure, H humidity, T temperature, Q Qvar data, Stp step counter). FSMx: Finite state machine bytes. MLCx: Machine Learning core source registers bytes DTx : data bytes. sw1|sw2: User buttons BT1, BT2 state. Standalone ASCII characters: 'U', 'O', 'T' for data type row recognition.*

4.2.3 *debug

This command starts continuous data acquisition in debug mode. When sent to the board, it returns the output values measured by the device formatted in a readable text format.

4.2.4 *stop

This command interrupts any acquisition session that has been started with either the `*start` or `*debug` commands.

4.2.5 *zon and *zoff

These commands put the STM32F401VE microcontroller on the demonstration kit in 3-state (high impedance). They allow the isolation of the sensor from the microprocessor and let the user interact with the sensor in a purely analog fashion.

When the kit is first turned on, the lines are in 3-state (high impedance) mode and you must send the `*zoff` command to allow communication between the sensor and the microcontroller.

Note that the Zoff command also internally calls *power_on command which enables sensor Vdd and Vddio at default level defined internally in FW. In case user wants to change Vdd and Vddio levels he has to use *set_vdda and *set_vddio commands with proper values. See also example below.

After this command has been executed, LED D2 (Red) is turned on. If the `*zoff` command has not been launched, the firmware ignores any other command sent to sensor.

Correct example of *zoff command folowed by different supply voltage setting is following :

```
*setdb MKI181V1
*zoff // Default Vdd=2.5V Vddio = 2.5V are set
*set_vdda3.6 //settling Vdda to 3.6V
*set_vddio2.8 /settling Vddio to 1.8V
...
*gr4f // read whoami to check correct sensor is present
...
```

4.2.6 *dev

This command retrieves the name of the adapter connected to the demonstration kit; e.g., `LIS3DH`.

4.2.7 *ver

This command returns the version of the firmware loaded in the microprocessor; e.g., V1.5.2.

4.2.8 *rAA

This command reads the contents of the accelerometer registers in the demonstration kit board. The hexadecimal value AA written in upper case represents the address of the register to be read.

Once the read command is issued, the board returns RAAhDDh, where AA is the address sent by the user and DD is the data present in the register.

For example, to read the register at address 0x20, the user issues the command *r20, which would return a result like R20hC7h.

*4.2.8.1 *rmAA1NN*

This command allows the contents of multiple accelerometer registers in the demonstration kit board to be read in single data block. Once this command is issued, the board returns a set of NN values starting with $RMAA_1hNNhDD_1hDD_2h…\ DD_{NN}h$ where AA_1 is the starting address set by user and DD_1 is the data present in this register and so on for next registers.

For example, *rm2006 reads six registers starting from 0x20, which would return a result like RM20h06h27h00h00h00hA0h0Bh.

*4.2.8.2 *multi-rAA1AA2AA3...AAN*

This command reads multiple accelerometer registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of N values starting $RAA_1hDD_1h…\ AA_NhDDNh$ where AA_1 is the starting address set by user and DD_1 is the data present in this register.

For example, *multi-r202425292B2D reads six register starting from 0x20, which would return a result like MULTI-R20h27h24hA0h25h0Bh29hE0h2Bh3Fh2Dh90h.

4.2.9 *wAADD

This command writes the contents of the accelerometer registers in the demonstration kit board. The hexadecimal upper case values AA and DD represent the address of the register and the data to be written, respectively.

For example, *w20C7 writes 0xC7 to the register at address 0x20

4.2.10 *grAA

Section Revision historySection Revision historyThis command allows the contents of the gyroscope registers in the demonstration kit board to be read. The hexadecimal, upper case AA represents the address of the register to be read.

Once the read command is issued, the board returns GRAAhDDh, where AA is the address sent by the user and DD is the data present in the register.

For example, *gr20 reads the register at address 0x20, which would return a result like GR20hC7h.

*4.2.10.1 *grmAA1NN*

This command allows the contents of multiple gyroscope registers in the demonstration kit board to be read in single data block. Once this command is issued, the board returns set of NN values starting with $GRMAA_1hNNhDD_1hDD_2h…\ DD_{NN}h$ where AA_1 is the starting address set by user and DD_1 is the data present in this register and so on.

For example, *grm2006 reads six registers starting from 0x20, which would return a result like GRM20h06h27h00h00h00hA0h0Bh.

*4.2.10.2 *multi-grAA1AA2AA3...AAN*

This command reads multiple gyroscope registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of N values starting with $MULTI\text{-}GRAA_1hDD_1h…\ AA_NhDDNh$ where AA_1 is the starting address set by user and DD_1 is the data present in this register and so on.

For example, *multi-gr202425292B2D reads six registers starting from 0x20, which would return a result like MULTI-GR20h27h24hA0h25h0Bh29hE0h2Bh3Fh2Dh90h.

4.2.11 *gwAADD

This command writes the contents of the gyroscope registers in the demonstration kit board. The hexadecimal, upper case. AA and DD represent the address of the register and the data to be written, respectively.

For example, *gw20C7 writes 0xC7 to the register at address 0x20.

4.2.12 *mrAA

This command allows the contents of the magnetometer registers in the demonstration kit board to be read. The hexadecimal, upper case AA represents the address of the register to be read.

Once the read command is issued, the board returns MRAAhDDh, where AA is the address sent by the user and DD is the data present in the register.

For example, *mr00 reads the register at address 0x00, which would return a result like MR00h10h.

*4.2.12.1 *mrmAA1NN*

This command readss the contents of multiple magnetometer registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of NN values starting with $RMAA_1hNNhDD_1hDD_2h$… $DD_{NN}h$ where AA_1 is the starting address set by user and DD_1 is the data present in this register.

For example, *mrm2006 reads six register starting from 0x20, which would return a result like MRM20h06h27h00h00h00hA0h0Bh.

*4.2.12.2 *multi-mrAA1AA2AA3...AAN*

This command reads multiple magnetometer registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of N values starting with $MULTI-MRAA_1hDD_1h$… AA_NhDDNh, where AA_1 is the starting address set by user and DD_1 is the data present in this register, and so on.

For example, *multi-mr202425292B2D reads six registers starting from 0x20, which would return a result like MULTI-MR20h27h24hA0h25h0Bh29hE0h2Bh3Fh2Dh90h.

4.2.13 *mwAADD

This command writes the contents of the magnetometer registers in the demonstration kit board. Hexadecimal, upper case AA and DD represent the address of the register and the data to be written, respectively.

For example, *mw0120 writes 0x20 to the register at address 0x01.

4.2.14 *prAA

This command reads the contents of the pressure sensor registers in the demonstration kit board. The hexadecimal, upper case AA represents the address of the register to be read.

Once the read command is issued, the board returns PRAAhDDh, where AA is the address sent by the user and DD is the data present in the register.

For example, *pr20 reads the register at address 0x20, which would return a value like PR20h10h.

*4.2.14.1 *prmAA1NN*

This command reads the contents of multiple pressure sensor registers in the demonstration kit in a single data block. Once this command is issued, the board returns set of NN values starting with $PRMAA_1hNNhDD_1hDD_2h$… $DD_{NN}h$ where AA_1 is the starting address set by user and DD_1 is the data present in this register, and so on…

For example, *prm2006 reads six registers starting from 0x20, which would return a value like PRM20h06h27h00h00h00hA0h0Bh.

*4.2.14.2 *multi-prAA1AA2AA3...AAN*

This command reads multiple pressure sensor registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of N values starting with $MULTI-PRAA_1hDD_1h$… AA_NhDDNh where AA_1 is the starting address set by user and DD_1 is the data present in this register, and so on.

For example, *multi-pr202425292B2D reads six registers starting from 0x20, which would return a result like MULTI-PR20h27h24hA0h25h0Bh29hE0h2Bh3Fh2Dh90h.

4.2.15 *pwAADD

This command writes the contents of the pressure sensor registers in the demonstration kit board. The hexadecimal, upper case AA and DD represent the address of the register and the data to be written, respectively.

For example, *pw20C7 writes 0xC7 to the register at address 0x20.

4.2.16 *hrAA

This command reads the contents of the humidity sensor registers in the demonstration kit board. The hexadecimal, upper case AA represents the address of the register to be read.

Once the read command is issued, the board returns HRAAhDDh, where AA is the address sent by the user and DD is the data present in the register.

For example, *hr20 reads the register at address 0x20, which would return a result like HR20h10h.

4.2.16.1 *hrmAA1NN

This command reads the contents of multiple humidity sensor registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of NN values starting with $HRMAA_1hNNhDD_1hDD_2h…$ $DD_{NN}h$ where AA_1 is the starting address set by user and DD_1 is the data present in this register, and so on.

For example, *hrm2006 reads six registers starting from 0x20, which would return a result like HRM20h06h27h00h00h00hA0h0Bh.

4.2.16.2 *multi-hrAA1AA2AA3...AAN

This command reads multiple humidity sensor registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of N values starting with $MULTI-HRAA_1hDD_1h…$ AA_NhDDNh where AA_1 is the starting address set by user and DD_1 is the data present in this register, and so on.

For example, *multi-hr202425292B2D reads six registers starting from 0x20, which would return a result like MULTI-HR20h27h24hA0h25h0Bh29hE0h2Bh3Fh2Dh90h.

4.2.17 *hwAADD

This command writes the contents of the humidity sensor registers in the demonstration kit board. The hexadecimal, upper case AA and DD represent the address of the register and the data to be written, respectively.

For example *hw20C7 writes 0xC7 to the register at address 0x20.

4.2.18 *trAA

This command reads the contents of the temperature sensor registers in the demonstration kit board. The hexadecimal, upper case AA represents the address of the register to be read. Once the read command is issued, the board returns TRAAhDDh, where AA is the address sent by the user and DD is the data present in the register. For example, *tr20 reads the register at address 0x20, which would return a result like TR20h10h.

4.2.18.1 *trmAA1NN

This command reads the contents of multiple temperature sensor registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of NN values starting with $TRMAA_1hNNhDD_1hDD_2h…$ $DD_{NN}h$ where AA_1 is the starting address set by user and DD_1 is the data present in this register, and so on. For example, *trm2006 reads six registers starting from 0x20, which would return a result like TRM20h06h27h00h00h00hA0h0Bh.

4.2.18.2 *multi-trAA1AA2AA3...AAN

This command reads multiple temperature sensor registers in the demonstration kit board in a single data block. Once this command is issued, the board returns set of N values starting with $MULTI-TRAA_1hDD_1h…$ AANhDDNh where AA_1 is the starting address set by user and DD_1 is the data present in this register, and so on. For example, *multi-tr202425292B2D reads six registers starting from 0x20, which would return a result like MULTI-TR20h27h24hA0h25h0Bh29hE0h2Bh3Fh2Dh90h.

4.2.19 *twAADD

This command writes the contents of the temperature sensor registers in the demonstration kit board. The hexadecimal, upper case `AA` and `DD` represent the address of the register and the data to be written, respectively. For example `*tw20C7` writes 0xC7 to the register at address 0x20.

4.2.20 *single

This command may be used to read just one set of data. It returns the read values of one data sample if the sensor is configured properly.

4.2.21 *sindebug

This command returns a single point data acquisition in readable text ASCII format.

4.2.22 *list

The command returns the list of the MKI adapters and devices supported by the firmware in ASCII format.

4.2.23 *listdev

This command returns the list of devices supported by the firmware in ASCII format.

4.2.24 *echoon

This command is used to activate the write command verbose mode so that the firmware automatically reads the contents of a register that has just been written to check if the write was successful.

For example, `* echoon` launched after `*w2027` returns `R2027`.

4.2.25 *echooff

This command stops the write command verbose mode.

4.2.26 *fiforst

This command enables the accelerometer FIFO reset mode. For more details see application note AN3308 on www.st.com.

4.2.27 *fifomde

This command enables the accelerometer FIFO mode. For more details see application note AN3308 on www.st.com.

4.2.28 *fifostr

This command enables the accelerometer FIFO stream mode. For more details see application note AN3308 on www.st.com.

4.2.29 *fifostf

This command enables the accelerometer Stream-to-FIFO mode. For more details see application note AN3308 on www.st.com.

4.2.30 *fifobtf

This command enables the accelerometer Bypass-to-FIFO mode.

4.2.31 *fifobts

This command enables the accelerometer Bypass-to-Stream mode.

4.2.32 *fifodstr

This command enables the accelerometer Dynamic Stream mode.

4.2.33 *gfiforst

This command enables the gyroscope FIFO reset mode.

4.2.34 *gfifomde

This command enables the gyroscope FIFO mode.

4.2.35 *gfifostr

This command enables the gyroscope FIFO stream mode.

4.2.36 *gfifostf

This command enables the gyroscope Stream-to-FIFO mode.

4.2.37 *gfifobtf

This command enables the gyroscope Bypass-to-FIFO mode.

4.2.38 *gfifobts

This command enables the gyroscope Bypass-to-Stream mode.

4.2.39 *gfifodstr

This command enables the gyroscope Dynamic Stream mode.

4.2.40 *pfiforst

This command enables the pressure sensor FIFO reset mode.

4.2.41 *pfifomde

This command enables the pressure sensor FIFO mode.

4.2.42 *pfifostr

This command enables the pressure sensor FIFO stream mode.

4.2.43 *pfifostf

This command enables the pressure sensor Stream-to-FIFO mode.

4.2.44 *pfifobtf

This command is used to enable the pressure sensor Bypass-to-FIFO mode.

4.2.45 *pfifobts

This command is used to enable the pressure sensor Bypass-to-Stream mode.

4.2.46 *pfifodstr

This command enables the pressure sensor Dynamic Stream mode.

4.2.47 *set_vddaV.VVV and *set_vddioV.VVV

These commands set the power supply VDD and VDDIO voltage values of device adapter.

For example, *set_vdda3.6 sets 3.6V on VDDA. Up to 3 decimal places can be used for set of V.VVV voltage value. If * set_vdda and * set_vddio have not been sent before the `* power_on` command, the setting is the default voltage defined in the device datasheet selected during the `* setdb` initialization. Please refer to device datasheet regarding specified VDD and VDDIO values and their relationship (i.e., which of these can be higher, etc.).

As the maximum voltages are defined also by the `* setdb` command, you cannot set values above these limits.

4.2.48 *power_on and *power_off

These commands are used to switch on and to switch off the VDD and VDDIO power supplies of the device adapter together. The proper power-on sequence of both voltages is handled internally by firmware.

Note that command *power_on must be sent after commands *setvddaV.VVV and *setvddioV.VVV otherwise it sets voltages to 0V (default).

Correct example of *power_on and power_off commands is following:

```
*setdb181V1
*zoff // Default Vdd=2.5V Vddio = 2.5V are set
*power_off // physical switch-off of sensor
*set_vdda3.6 //settling Vdda to 3.6V
*set_vddio2.8 /settling Vddio to 1.8V
*power_on // physical switch-on of sensor
...
*gr4f // read whoami to check correct sensor is present
...
```

4.2.49 *odr [p1]

This command is used to set ODR speed when data acquisition is driven by TIMER (analog sensors).

- **P1** ODR in mHz: 1 to 2000000

4.2.50 *get_i [p1] [p2] [p3] [p4]

This command is used to get average IDD and IDDIO during set/calculated time interval. Current ranges toggling and measurement method selection is done automatically as default setup.

- **P1** form of output data:
  - **F** or **f** - floating-point (1 µA base)
  - **B** or **b** - fixed-point (1 nA base)
  - **A** or **a** - ASCII (readable text)
- **P2** status bits:
  - **S** or **s** - Two bytes containing information (overflow, underflow, current range index) will be added at the end of the IDD, IDDIO data
- **P3** continuous (repetitive) measurement:
  - **C** or **c** - firmware will send measured values periodically
- **P4** Long Time Enable:
  - **L** or **l** firmware will wait for waveform edges or Drdy pulses until it appears or until another *get_i or *adc_uni_stop command interrupts it

4.2.51 *get_v [p1]

This command is used to get average VDD and VDDIO VddIO (precise measurement with calibrations).

- **P1** form of output data:
  - **F** or **f** - floating-point (1 V base)
  - **B** or **b** - fixed-point (1 mV base)
  - **A** or **a** - ASCII (readable text)

4.2.52 *ranges_auto and *ranges_man

These commands are used to toggle between automatic and manual current ranges selection. The default command is *ranges_auto.

4.2.53 *range_ma [p1] and *range_mb [p1]

These commands are used to select current ranges manually.

- *range_ ma … IDD, *range_ mb … IDDIO
- **P1** current range: 5, 10, 25, 50, 100, 200, 500, 1000, 2500, 5000, 10000, 20000, 50000, 100000
- Full scale of the selected range = 300000 / P1 [µA]

4.2.54 *adc_run_time [p1]

This command is used to set the time interval during which ADC samples the measured current waveforms.

- **P1** time in milliseconds: 1 to 50000 (default is 500 ms)

4.2.55 *adc_uni_stop

This command is used to stop all measurements.

4.3 Digital output accelerometers: supported commands

Table 5. Digital output accelerometers: supported commands list

Command	Description	Returned value(1)
*setdbXXXVY	Selects firmware according to the adapter connected	
*start	Starts continuous data acquisition	(See Table 4. Returned values for *start command)
*debug	Returns the output data in readable text format	
*stop	Stops data acquisition	
*zon(2)	Forces High impedance state. Turns off VDD and VDDIO power supply	
*zoff	Exits from High impedance state. Turns on VDD and VDDIO power supply, configures IO communication type and sets special registers if needed	
*dev	Device name	e.g.: LIS3DH
*ver	Firmware version	e.g.: V1.5.2
*rAA	Accelerometer register read	e.g.: RAAhDDh
*rmAA$_1$NN	Multiple accelerometer registers read	e.g.: RMAA$_1$hNNhDD$_1$h…DD$_{NN}$h
*multi-rAA$_1$ .. AA$_N$	Multiple accelerometer registers read	e.g.:MULTI-RAA$_1$hDD$_1$h…AA$_N$hDDNh
*wAADD	Accelerometer register write	
*single	It gets a single point data acquisition	(See Table 4. Returned values for *start command)
*sindebug	It gets a single point data acquisition in readable text format	
*list	Prints the list of MKIs supported	e.g.: MKI001V1→LPS22HB MKI087V1→ LIS331DL MKI089V1→ LIS331DLH
*listdev	Prints the list of devices supported	e.g.: LIS331DL LIS331DLH LIS331HH…
*echoon	Activates the write verbose mode	e.g.: RAAhDDh
*echooff	Deactivates the write verbose mode	
*fiforst (3)	Accelerometer “Reset mode” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*fifomde(3)	Accelerometer “FIFO mode” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*fifostr(3)	Accelerometer “FIFO stream” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*fifostf(3)	Accelerometer “Stream-to-FIFO” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*fifobtf(3)	Accelerometer “Bypass-to-FIFO” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*fifobts(3)	Accelerometer “Bypass-to-Stream” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*fifodstr(3)	Accelerometer “Dynamic Stream” enable	st XH XL YH YL ZH ZL IR FC FS \r \n

1. *IR: interrupt bytes; FC: FIFO control register; FS: FIFO source register*
2. *Correctly turns off sensor, deinitializes communication with sensor, forces High impedance state and turns off VDD and VDDIO power supply*
3. *Available only for devices with embedded FIFO*

4.4 Digital output gyroscopes: supported commands

The table below lists the commands supported by the devices/demonstration boards including a digital output gyroscope.

Table 6. Digital output gyroscopes: supported commands list

Command	Description	Returned value(1)
*setdbXXXVY	Selects FW according to the adapter connected	
*start	Starts continuous data acquisition	(See Table 4. Returned values for *start command)
*debug	Returns the output data in readable text format	
*stop	Stops data acquisition	
*zon	Forces High impedance state. Turns off VDD and VDDIO power supply	
*zoff	Exits from High impedance state. Turns on VDD and VDDIO power supply.	
*dev	Device name	e.g.: L3GD20H
*ver	Firmware version	e.g.: V1.5.2
*grAA	Gyroscope register read	e.g.: GRAAhDDh
*grmAA$_1$NN	Multiple gyroscope registers read	e.g.: GRMAA$_1$hNNhDD$_1$h…DD$_{NN}$h
*multi-grAA$_1$ .. AA$_N$	Multiple gyroscope registers read	e.g.:MULTI-GRAA$_1$hDD$_1$h…AA$_N$hDDNh
*gwAADD	Gyroscope register write	
*single	It gets a single point data acquisition	(See Table 4. Returned values for *start command)
*sindebug	It gets a single point data acquisition in readable text format	
*list	Prints the list of MKIs supported	e.g.: MKI001V1→LPS22HB MKI087V1→ LIS331DL MKI089V1→ LIS331DLH
*listdev	Prints the list of devices supported	e.g.: LIS331DL LIS331DLH LIS331HH…
*echoon	Activates the write verbose mode	e.g.: GRAAhDDh
*echooff	Deactivates the write verbose mode	
*gfiforst (2)	Gyroscope “Reset mode” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*gfifomde(2)	Gyroscope “FIFO mode” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*gfifostr(2)	Gyroscope “FIFO stream” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*gfifostf(2)	Gyroscope “Stream to FIFO” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*gfifobtf(2)	Gyroscope “Bypass to FIFO” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*gfifobts(2)	Gyroscope “Bypass to stream” enable	st XH XL YH YL ZH ZL IR FC FS \r \n
*gfifodstr(2)	Gyroscope “Dynamic stream” enable	st XH XL YH YL ZH ZL IR FC FS \r \n

1. *IR: interrupt bytes; FC: FIFO control register; FS: FIFO source register*
2. *Available only for devices with embedded FIFO*

4.5 Digital output magnetometers: supported commands

Table 7. Digital output magnetometer: supported commands list

Command	Description	Returned value(1)
*setdbXXXVY	Selects firmware according to the adapter connected	
*start	Starts continuous data acquisition	(See Table 4. Returned values for *start command)
*debug	Returns the output data in readable text format	
*stop	Stops data acquisition	
*zon	Forces High impedance state. Turns off VDD and VDDIO power supply.	
*zoff	Exits from High impedance state. Turns on VDD and VDDIO power supply.	
*dev	Device name	e.g.: LIS2MDL
*ver	Firmware version	e.g.: V1.5.2
*mrAA	Magnetometer register read	e.g.: MRAAhDDh
*mrmAA$_1$NN	Multiple magnetometer registers read	e.g.: MRMAA$_1$hNNhDD$_1$h…DD$_{NN}$h
*multi-mrAA$_1$...AA$_N$	Multiple magnetometer registers read	e.g.:MULTI-MRAA$_1$hDD$_1$h…AA$_N$hDDNh
*mwAADD	Magnetometer register write	
*single	It gets a single point data acquisition	(See Table 4. Returned values for *start command)
*sindebug	It gets a single point data acquisition in readable text format	
*list	Prints the list of MKIs supported	e.g.: MKI001V1→LPS22HB MKI087V1→ LIS331DL MKI089V1→ LIS331DLH
*listdev	Prints the list of devices supported	e.g.: LIS331DL LIS331DLH LIS331HH…
*echoon	Activates the write verbose mode	e.g.: MRAAhDDh
*echooff	Deactivates the write verbose mode	

1. *IR: interrupt bytes; FC: FIFO control register; FS: FIFO source register*

4.6 Digital output pressure sensor: supported commands

Table 8. Digital output pressure sensor: supported commands list

Command	Description	Returned value(1)
*setdbXXXVY	Selects firmware according to the adapter connected	
*start	Starts continuous data acquisition	(See Table 4. Returned values for *start command)
*debug	Returns the output data in readable text format	
*stop	Stops data acquisition	
*zon	Forces High impedance state. Turns off VDD and VDDIO power supply.	
*zoff	Exits from High impedance state. Turns on VDD and VDDIO power supply	
*dev	Device name	e.g.: LPS22HH
*ver	Firmware version	e.g.: V1.5.2
*prAA	Pressure sensor register read	e.g.: PRAAhDDh
*prmAA$_1$NN	Multiple pressure sensor registers read	e.g.: PRMAA$_1$hNNhDD$_1$h…DD$_{NN}$h
*multi-prAA$_1$ .. AA$_N$	Multiple pressure sensor registers read	e.g.:MULTI-PRAA$_1$hDD$_1$h…AA$_N$hDDNh
*pwAADD	Pressure sensor register write	
*single	It gets a single point data acquisition	(See Table 4. Returned values for *start command)
*sindebug	It gets a single point data acquisition in readable text format	
*list	Prints the list of MKIs supported	e.g.: MKI001V1→LPS22HB MKI087V1→ LIS331DL MKI089V1→ LIS331DLH
*listdev	Prints the list of devices supported	e.g.: LIS331DL LIS331DLH LIS331HH…
*echoon	Activates the write verbose mode	e.g.: PRAAhDDh
*echooff	Deactivates the write verbose mode	
*pfiforst (2)	Pressure sensor “Reset mode” enable	st PXL PL PH TL TH IR FC FS \r \n
*pfifomde(2)	Pressure sensor “FIFO mode” enable	st PXL PL PH TL TH IR FC FS \r \n
*pfifostr(2)	Pressure sensor “FIFO stream” enable	st PXL PL PH TL TH IR FC FS \r \n
*pfifostf(2)	Pressure sensor “Stream to FIFO” enable	st PXL PL PH TL TH IR FC FS \r \n
*pfifobtf(2)	Pressure sensor “Bypass to FIFO” enable	st PXL PL PH TL TH IR FC FS \r \n
*pfifobts(2)	Pressure sensor “Bypass to stream” enable	st PXL PL PH TL TH IR FC FS \r \n
*pfifodstr(2)	Pressure sensor “Dynamic stream” enable	st PXL PL PH TL TH IR FC FS \r \n

1. *IR: interrupt bytes; FC: FIFO control register; FS: FIFO source register*
2. *Available only for devices with embedded FIFO*

4.7 Digital output humidity sensor: supported commands

The table below lists the commands supported by the devices/demonstration boards including a digital output humidity sensor.

Table 9. Digital output humidity sensor: supported commands list

Command	Description	Returned value
*setdbXXXVY	Selects firmware according to the adapter connected	
*start	Starts continuous data acquisition	(See Table 4. Returned values for *start command)
*debug	Returns the output data in readable text format	
*stop	Stops data acquisition	
*zon	Forces High impedance state. Turns off VDD and VDDIO power supply.	
*zoff	Exits from High impedance state. Turns on VDD and VDDIO power supply.	
*dev	Device name	e.g.: HTS221
*ver	Firmware version	e.g.: V1.5.2
*hrAA	Humidity sensor register read	e.g.: HRAAhDDh
*hrmAA_1NN	Multiple humidity sensor registers read	e.g.: HRMAA_1hNNhDD_1h…DD_{NN}h
*multi-hrAA_1 .. AA_N	Multiple humidity sensor registers read	e.g.:MULTI-HRAA_1hDD_1h…AA_NhDDNh
*hwAADD	Humidity sensor register write	
*single	It gets a single point data acquisition	(See Table 4. Returned values for *start command)
*sindebug	It gets a single point data acquisition in readable text format	
*list	Prints the list of MKIs supported	e.g.: MKI001V1→LPS22HB MKI087V1→ LIS331DL MKI089V1→ LIS331DLH
*listdev	Prints the list of devices supported	e.g.: LIS331DL LIS331DLH LIS331HH…
*echoon	Activates the write verbose mode	e.g.: HRAAhDDh
*echooff	Deactivates the write verbose mode	

4.8 Digital output temperature sensor: supported commands

The table below lists the commands supported by the devices/evaluation boards including a digital output temperature sensor.

Table 10. Digital output temperature sensor: supported commands list

Command	Description	Returned value
*setdbXXXVY	Selects firmware according to the adapter connected	
*start	Starts continuous data acquisition	(See: Table 4. Returned values for *start command)
*debug	Returns the output data in readable textformat	
*stop	Stops data acquisition	
*zon	Forces High impedance state. Turns off VDD and VDDIO power supply.	
*zoff	Exits from High impedance state. Turns on VDD and VDDIO power supply.	
*dev	Device name	e.g.:STTS22H
*ver	Firmware version	e.g.: V1.5.2
*trAA	Temperature sensor register read	e.g.:TRAAhDDh
*trmAA_1NN	Multiple temperature sensor registers read	e.g.: TRMAA_1hNNhDD_1h…DD_{NN}h
*multi-trAA_1 .. AAN	Multiple temperature sensor registers read	e.g.:MULTI-TRAA_1hDD_1h…AA_NhDD_Nh
*twAADD	Temperature sensor register write	
*single	It gets a single point data acquisition	(See: Table 4. Returned values for *start command)
*sindebug	It gets a single point data acquisition in readable text format	
*list	Prints the list of MKIs supported	e.g.: MKI001V1→LPS22HB MKI087V1→ LIS331DL MKI089V1→ LIS331DLH
*listdev	Prints the list of devices supported	e.g.: LIS331DL LIS331DLH LIS331HH…
*echoon	Activates the write verbose mode	e.g.: TRAAhDDh
*echooff	Deactivates the write verbose mode	

4.9 Analog output temperature sensor: supported commands

The table below lists the commands supported by the devices/evaluation boards that include an analog output temperature sensor.

Table 11. Analog output temperature sensor: supported commands list

Command	Description	Returned value
*setdbXXXVY	Selects firmware according to the adapter connected	
*odr [param]	Sets ODR speed (parameter: ODR in mHz)	
*start	Starts continuous data acquisition	(See Table 4. Returned values for *start command)
*debug	Returns the output data in readable text format	e.g.: Temp. = 22.93 [°C]
*stop	Stops data acquisition	
*zon	Forces High impedance state. Turns off VDD and VDDIO power supply.	
*zoff	Exits from High impedance state. Turns on VDD and VDDIO power supply.	
*dev	Device name	e.g.: STLM20
*ver	Firmware version	e.g..: V3.7.30
*single	Gets a single point data acquisition	(SeeTable 4. Returned values for *start command)
*sindebug	Gets a single point data acquisition in readable text format	e.g.: Temp. = 22.93 [°C]
*list	Prints the list of STEVAL-MKIxxxxx evaluation boards supported	e.g.: MKI001V1→LPS22HB MKI087V1→ LIS331DL MKI089V1→ LIS331DLH
*listdev	Prints the list of devices supported	e.g.: LIS331DL LIS331DLH LIS331HH, etc.

4.10 Configuration of FIFO mode on combined devices

For the configuration of combined sensors as iNEMO class LSM6xxx and similar there are no dedicated commands for the Professional MEMS tool intended for device simple setup.

As devices are becoming more complex in each generation and with dedicated functionalities it is not possible to implement general dedicated commands. Refer to the respective device data sheet for its exact configuration and, particularly, for FIFO configuration. The meaning of data in FIFO registers then depends on complex FIFO setup defined by the end user.

5 Schematic diagrams

Figure 9. **STEVAL-MKI109V3 circuit schematic (1 of 8)**

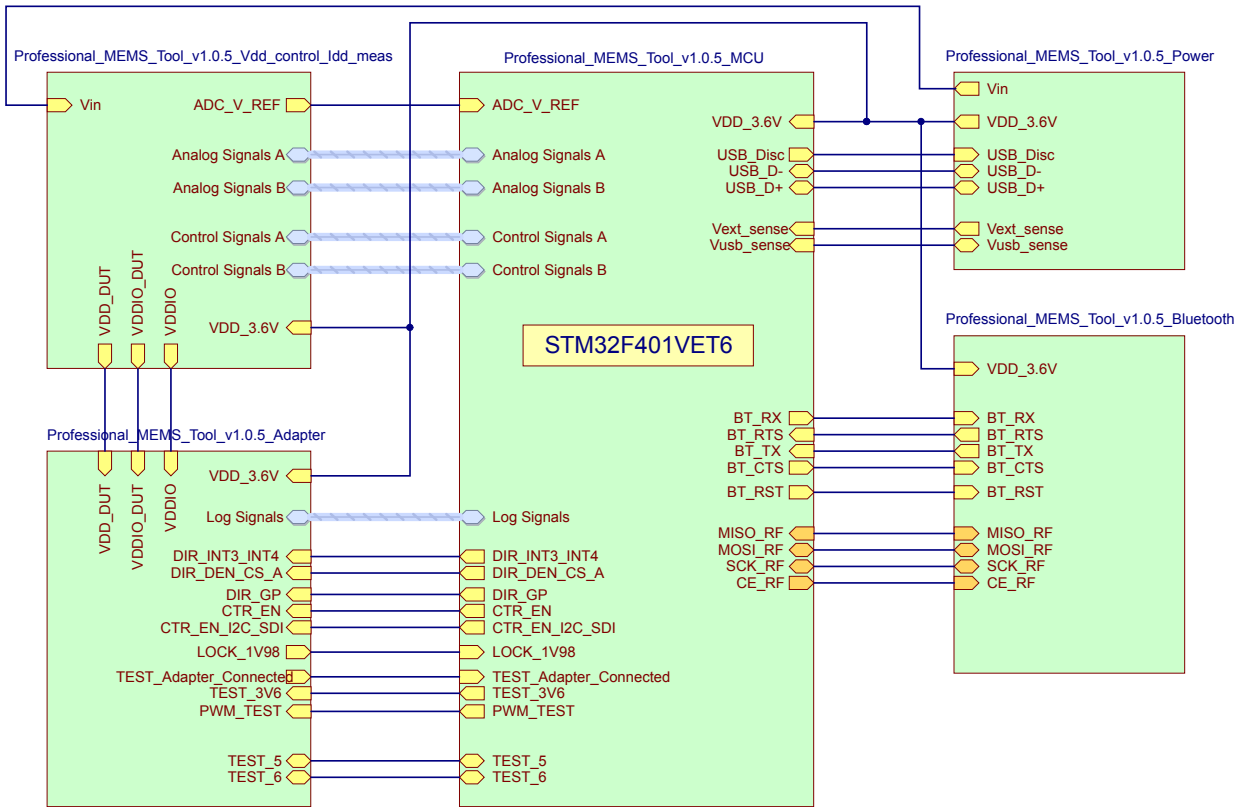


Figure 10. **STEVAL-MKI109V3 circuit schematic (2 of 8)**

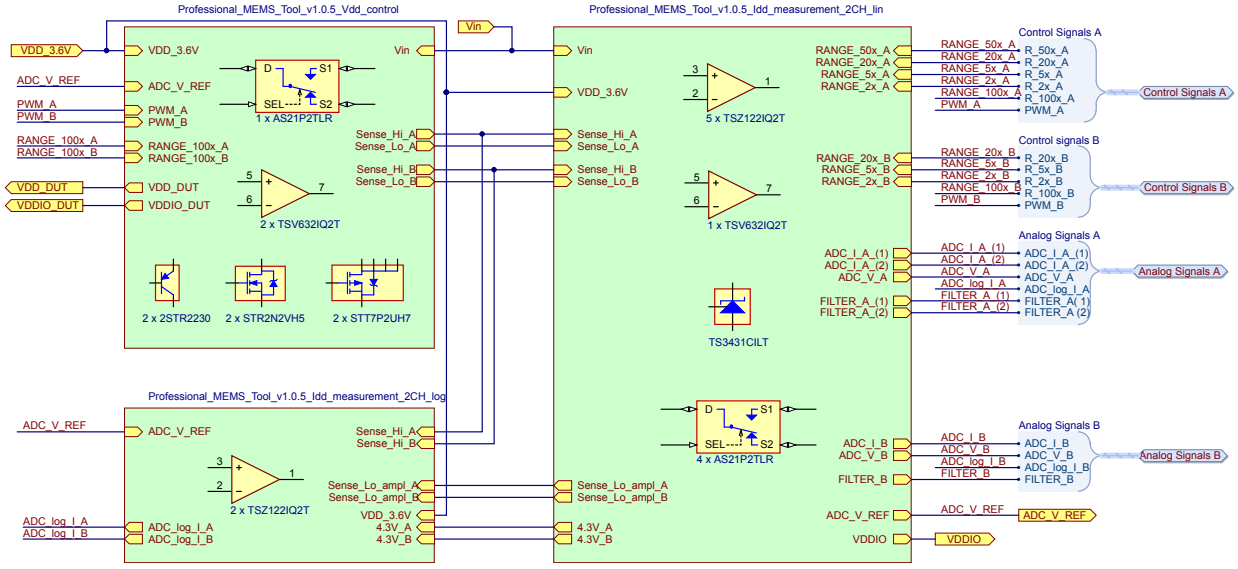


Figure 11. **STEVAL-MKI109V3 circuit schematic (3 of 8)**

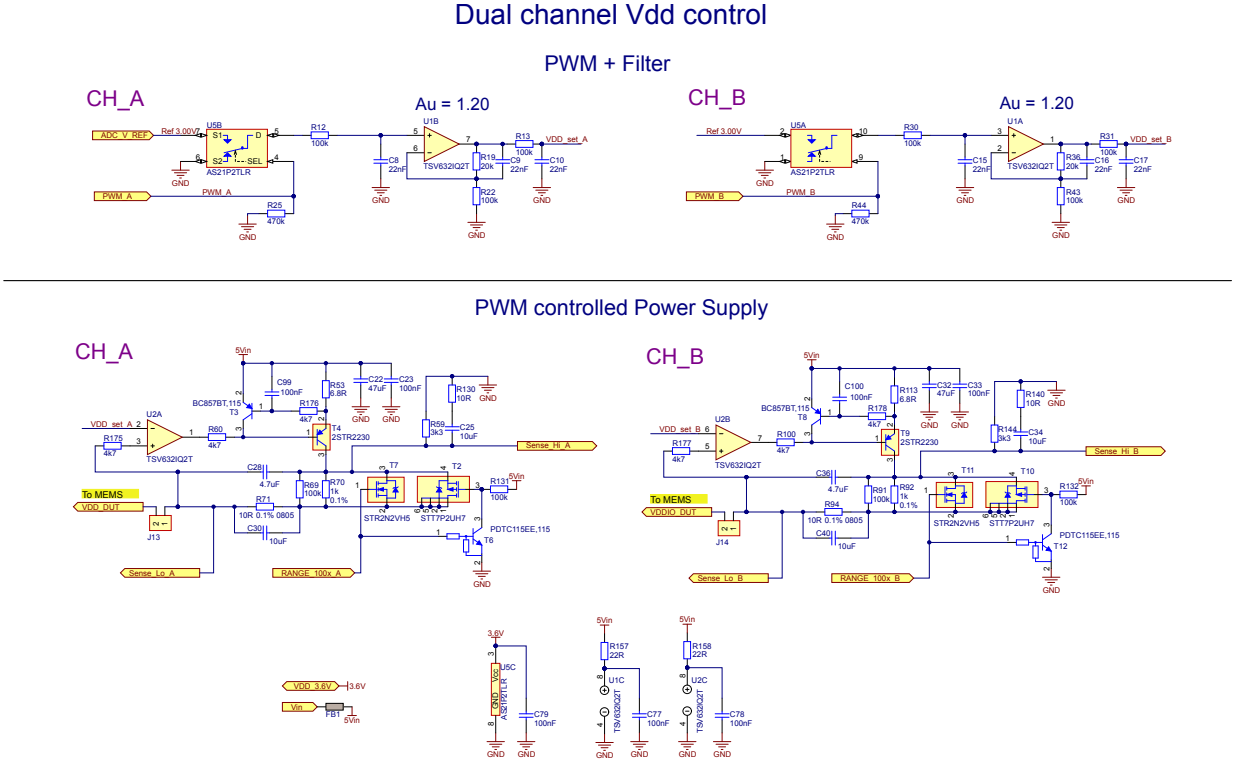


Figure 12. **STEVAL-MKI109V3 circuit schematic (4 of 8)**

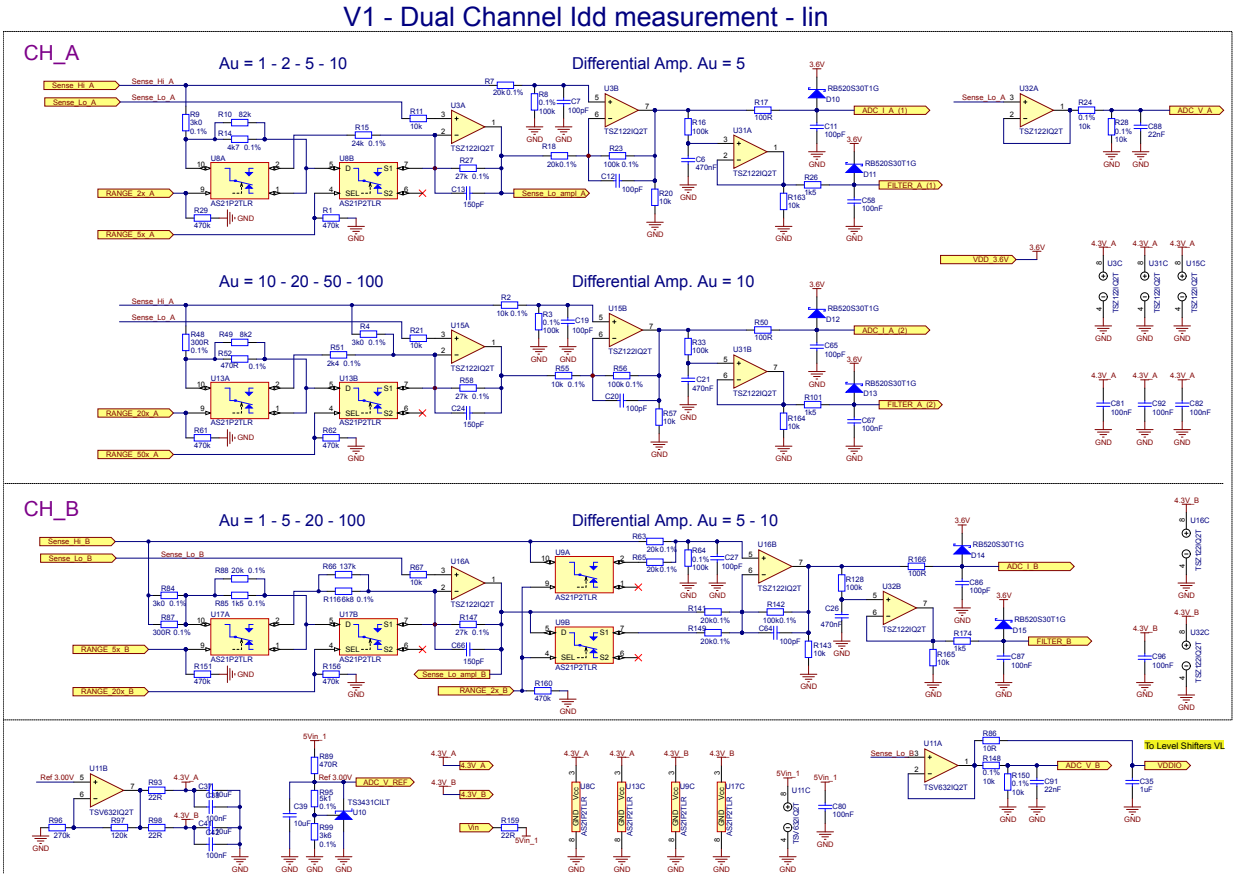


Figure 13. **STEVAL-MKI109V3 circuit schematic (5 of 8)**

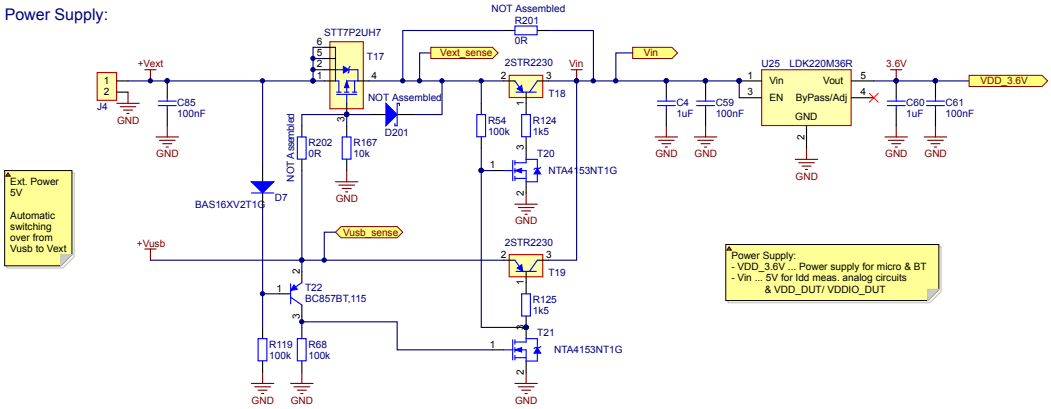


Figure 14. **STEVAL-MKI109V3 circuit schematic (6 of 8)**

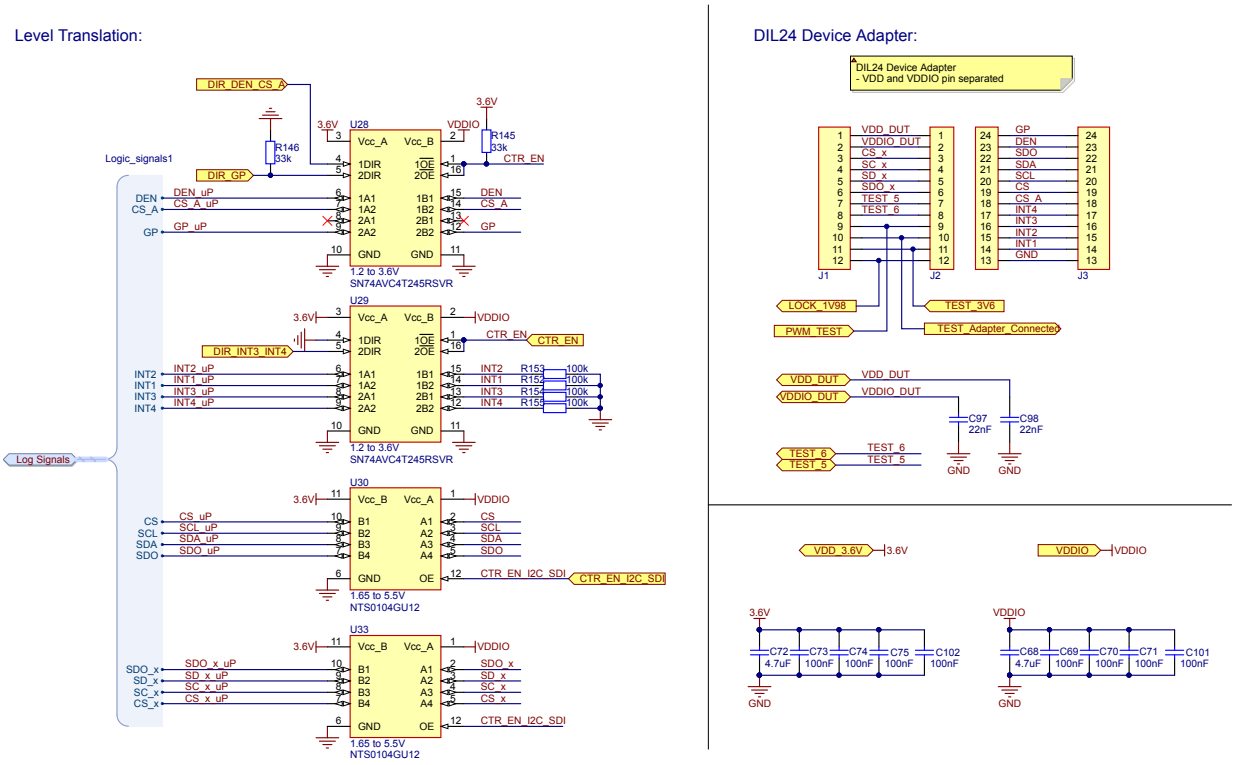


Figure 15. **STEVAL-MKI109V3 circuit schematic (7 of 8)**

Bluetooth Module Connection:

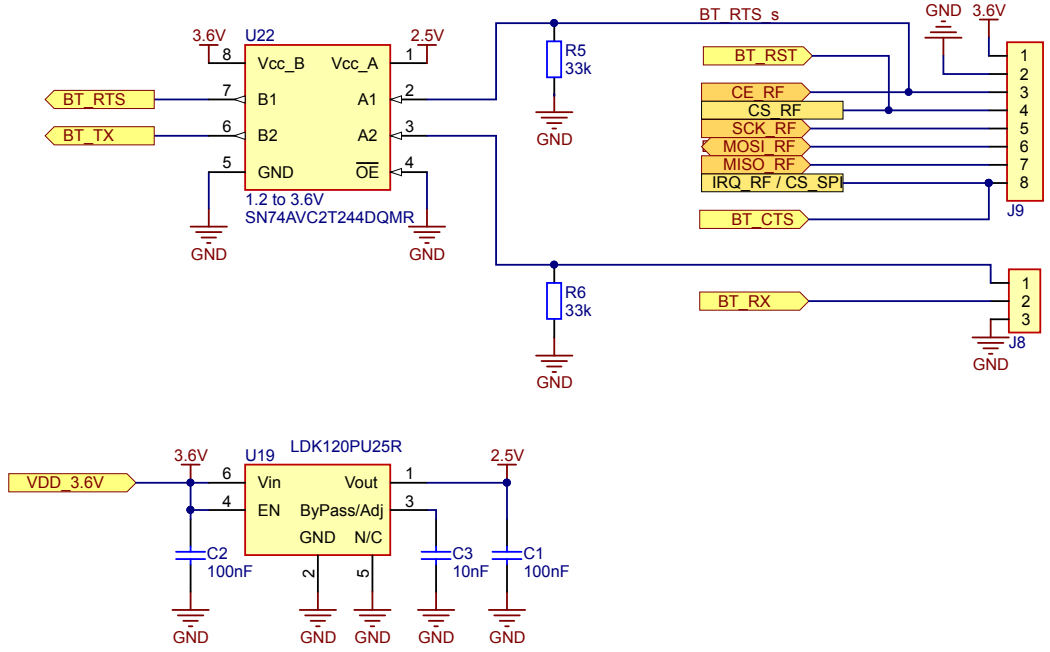
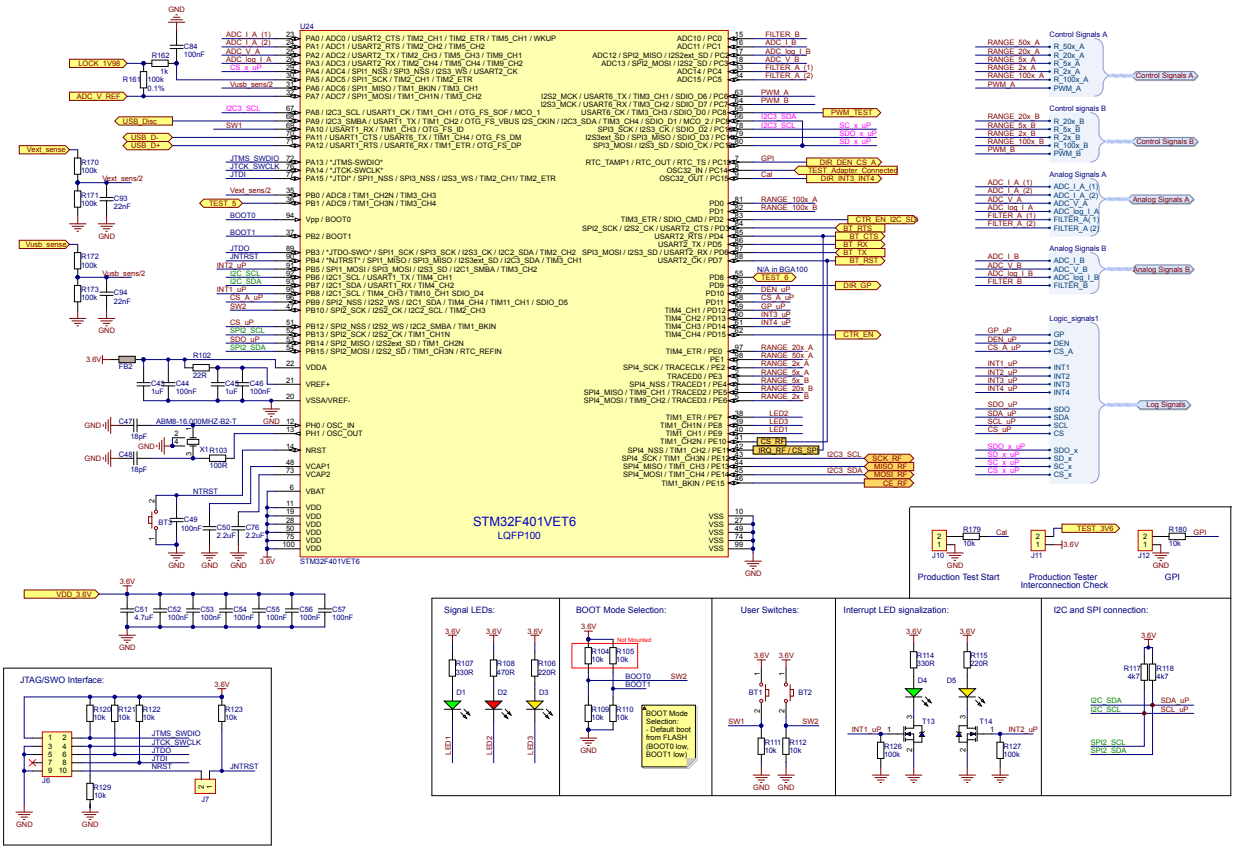


Figure 16. **STEVAL-MKI109V3 circuit schematic (8 of 8)**



6 Certified MEMS evaluation boards/kits

The following MEMS evaluation boards are certified:

- STEVAL-MET001V1
- STEVAL-MKI015V1
- STEVAL-MKI089V1
- STEVAL-MKI092V2
- STEVAL-MKI105V1
- STEVAL-MKI106V1
- STEVAL-MKI107V2
- STEVAL-MKI109V2
- STEVAL-MKI109V3
- STEVAL-MKI110V1
- STEVAL-MKI122V1
- STEVAL-MKI123V1
- STEVAL-MKI125V1
- STEVAL-MKI127V1
- STEVAL-MKI133V1
- STEVAL-MKI134V1
- STEVAL-MKI135V1
- STEVAL-MKI136V1
- STEVAL-MKI137V1
- STEVAL-MKI141V2
- STEVAL-MKI142V1
- STEVAL-MKI151V1
- STEVAL-MKI153V1
- STEVAL-MKI154V1
- STEVAL-MKI158V1
- STEVAL-MKI159V1
- STEVAL-MKI160V1
- STEVAL-MKI164V1
- STEVAL-MKI165V1
- STEVAL-MKI166V1
- STEVAL-MKI167V1
- STEVAL-MKI168V1
- STEVAL-MKI169V1
- STEVAL-MKI170V1
- STEVAL-MKI172V1
- STEVAL-MKI173V1
- STEVAL-MKI174V1
- STEVAL-MKI175V1
- STEVAL-MKI176V1
- STEVAL-MKI177V1
- STEVAL-MKI178V1
- STEVAL-MKI178V2
- STEVAL-MKI179V1
- STEVAL-MKI180V1
- STEVAL-MKI181V1
- STEVAL-MKI182V1
- STEVAL-MKI182V2

- STEVAL-MKI183V1
- STEVAL-MKI184V1
- STEVAL-MKI185V1
- STEVAL-MKI186V1
- STEVAL-MKI188V1
- STEVAL-MKI189V1
- STEVAL-MKI190V1
- STEVAL-MKI191V1
- STEVAL-MKI192V1
- STEVAL-MKI193V1
- STEVAL-MKI194V1
- STEVAL-MKI195V1
- STEVAL-MKI196V1
- STEVAL-MKI197V1
- STEVAL-MKI198V1K
- STEVAL-MKI199V1K
- STEVAL-MKI200V1K
- STEVAL-MKI201V1K
- STEVAL-MKI202V1K
- STEVAL-MKI203V1K
- STEVAL-MKI204V1K
- STEVAL-MKI205V1
- STEVAL-MKI206V1
- STEVAL-MKI207V1
- STEVAL-MKI208V1K
- STEVAL-MKI209V1K
- STEVAL-MKI210V1K
- STEVAL-MKI210V2K
- STEVAL-MKI211V1K
- STEVAL-MKI212V1
- STEVAL-MKI213V1
- STEVAL-MKI214V1
- STEVAL-MKI215V1
- STEVAL-MKI216V1K
- STEVAL-MKI217V1
- STEVAL-MKI218V1
- STEVAL-MKI219V1
- STEVAL-MKI220V1
- STEVAL-MKI221V1
- STEVAL-MKI224V1
- STEVAL-MKI225A
- STEVAL-MKI228KA
- STEVAL-MKI229A
- STEVAL-MKI230KA
- STEVAL-MKI231KA
- STEVAL-MKI232A
- STEVAL-MKI233KA
- STEVAL-MKI234KA
- STEVAL-MKI235KA
- STEVAL-MKI236A

- STEVAL-MKI237KA
- STEVAL-MKI238A
- STEVAL-MKI239A
- STEVAL-MKI241KA
- STEVAL-MKI242A
- STEVAL-MKI243A

7 Regulatory compliance information

Note: This regulatory compliance information applies to all the boards listed in Section 6: Certified MEMS evaluation boards/kits.

Note: The evaluation kit with order code STEVAL-MKIxxxY contains the board whose finished good is STEVAL$MKIxxxYA (x is a number and Y is a letter).

Formal Notice Required by the U.S. Federal Communications Commission

FCC NOTICE

This kit is designed to allow:

(1) Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter 3.1.2.

Formal Product Notice Required by Industry Canada Innovation, Science and Economic Development

Canada compliance:

For evaluation purposes only. This kit generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to Industry Canada (IC) rules.

À des fins d'évaluation uniquement. Ce kit génère, utilise et peut émettre de l'énergie radiofréquence et n'a pas été testé pour sa conformité aux limites des appareils informatiques conformément aux règles d'Industrie Canada (IC).

Formal product notice required by EU

This device is in conformity with the essential requirements of the Directive 2014/30/EU (EMC) and of the Directive 2015/863/EU (RoHS).

Revision history

Table 12. **Document revision history**

Date	Version	Changes
05-Oct-2016	1	Initial release.
26-Feb-2018	2	Updated List of supported MEMS adapter boards, List of supported commands and Returned values for *start command
27-Jul-2018	3	Updated List of supported MEMS adapter boards, List of supported commands and Returned values for *start command Changed all descriptions for *Zon to "Forces High impedance state" (was Forces 3-state) and *Zoff to "Exits from High impedance state" (was Exits from 3-state) Minor text changes
23-Jan-2019	4	Updated Table 1. List of supported MEMS adapter boards and Table 3. Returned values for *start command. Added Section 4.8 Digital output temperature sensor: supported commands.
02-Jul-2019	5	Updated Table 1. List of supported MEMS adapter boards and Table 3. Returned values for *start command.
07-Oct-2019	6	Updated Section 2 Professional MEMS Tool board installation, Table 1. List of supported MEMS adapter boards and Table 3. Returned values for *start command. Added Section 2.2.1 Firmware upgrade on Windows.
03-Mar-2020	7	Updated Table 1. List of supported MEMS adapter boards and Table 3. Returned values for *start command.
15-Jul-2020	8	Updated Section 2.2.3 DFU on Mac OS®, Section 3 Supported MEMS adapter boards, Section 4.2 Supported commands, Section 4.2.2 *start, Section 4.3 Digital output accelerometers: supported commands, Section 4.4 Digital output gyroscopes: supported commands, Section 4.5 Digital output magnetometers: supported commands, Section 4.6 Digital output pressure sensor: supported commands, Section 4.7 Digital output humidity sensor: supported commands and Section 4.8 Digital output temperature sensor: supported commands. Added Section 4.2.21 *sindebug, Section 4.2.50 *odr [p1], Section 4.2.51 *get_i [p1] [p2] [p3] [p4], Section 4.2.52 *get_v [p1], Section 4.2.53 *ranges_auto and *ranges_man, Section 4.2.54 *range_ma [p1] and *range_mb [p1], Section 4.2.55 *adc_run_time [p1], Section 4.2.56 *adc_uni_stop and Section 4.9 Analog output temperature sensor: supported commands.
30-Jun-2021	9	Updated Section 2 Professional MEMS Tool board installation, Section 2.1 Hardware installation, Section 3 Supported MEMS adapter boards and Section 4.2.2 *start. Added Section 4.10 Configuration of FIFO mode on combined devices.
21-Apr-2022	10	Updated *Section 3 Supported MEMS adapter boards*, Firmware upgrade using DFU mode, *Section 4.2 Supported commands, Section 4.2.2 *start, Section 4.2.22 *list, Section 4.3 Digital* *output accelerometers: supported commands, Section 4.4 Digital output gyroscopes: supported* *commands, Section 4.5 Digital output magnetometers: supported commands, Section 4.6 Digital* *output pressure sensor: supported commands, Section 4.7 Digital output humidity sensor:* *supported commands, Section 4.8 Digital output temperature sensor: supported commands, and*

Date	Version	Changes
		Section 4.9 Analog output temperature sensor: supported commands. Added *Section 2.2 Firmware upgrade, Section 2.2.1 Entering DFU mode, Section 2.2.2 Firmware* *upgrade on Windows with STM32CubeProgrammer using *.bin source files, Section 2.2.3 DFU on* *Windows using the older DfuSe software, Section 6 Certified MEMS evaluation boards/kits, and* *Section 7 Regulatory compliance information.* Removed Section 2.2.1 Firmware upgrade on Windows.
25-Jan-2024	11	Updated *Section 3: Supported MEMS adapter boards,* *Section 4: Supported commands, Section 4.2.2: *start, Section 4.2.5: *zon* *and *zoff, Section 4.2.48: *power_on and *power_off, Section 4.3: Digital* *output accelerometers: supported commands* and *Section 6: Certified* *MEMS evaluation boards/kits* Section "*setvddaX.Y and *setvddioX.Y" changed to *Section 4.2.47: *set_vddaV.VVV and *set_vddioV.VVV.*
02-May-2024	12	Added STEVAL-MKI242A.

Contents

List of tables

List of figures