Getting started with STM32F746G-DISCO x BNO055 in Matlab and Simulink

Nikolas Schroeder
Institute for Systems Theory and Automatic Control
lrt86824@stud.uni-stuttgart.de

August 16, 2017

Abstract

This paper gives an overview on how to implement code and sensor interfaces on the STM32F746G Discovery Board using Matlab Embedded Coder and the proper hardware support package. After a short installation and setup guide it is shown how to interface an inertial measurement unit via I2C with the STM-Discovery and monitor the results in Simulink External Mode.

Background: The BNO055 IMU was selected for the IST Self-Driving Bicycle project in order to estimate a measure for the current roll angle ϕ and the corresponding angular rate $\dot{\phi}$.

1 Installation and Setup

In general the installation and setup of the tool chain were pretty straight forward. There are many good tutorials on mathworks.com which explain each necessary step in detail. However, a few minor problems occurred which made the whole process quite time consuming. Therefore, this section includes some hints to accelerate installation and setup. In the following it is recommended to read along the tutorial "Getting started with Embedded Coder support package for STMicroelectronics Discovery Boards".

1.1 Matlab and Simulink

Under version R2016b, Matlab was not able to detect the Discovery board. After updating to version R2017a this problem didn't occur anymore. Of course, this problem may originated from something different than the version but nevertheless, it is recommended to work with the most recent one. In order to automatically generate code for embedded systems out of Matlab files and Simulink models make sure that the following toolboxes are installed:

- Matlab Coder
- Simulink Coder
- Embedded Coder

1.2 Hardware Support Package

For the STM Discovery boards STM32F4 and STMF32F746G, a hardware support package, which comes along with custom Simulink blocks, is provided by Mathworks. In order to work with the "Embedded Coder Support Package for STM Discovery Boards" an additional support package called "Embedded Coder Support Package for ARM Cortex-M Processors" needs to be installed as well. The most convenient way to install the packages is to navigate to the Matlab Add-On explorer, browse for them and then use the auto install function.

Moreover the STM32F4 Discovery Board Firmware, which can be downloaded directly from STM, is required. During the setup of the hardware support package for the Discovery boards you'll be asked to provide the installation path of the firmware. In case any errors occur (also later on in code deployment when the package is already installed) the following steps can be checked for troubleshooting:

- Navigate to the board support package setup wizard and assure that the right installation path for the Discovery board firmware is included.

  Add-On Explorer → Manage Add-Ons → browse for board support package and click on Setup

- Check if the package installation path

  \ProgramData\MATLAB\SupportPackages\R2017a\3P.instrset

  has the 2 following folders "cmsis.instrset" and "gnuarm-armcortex.instrset".

- In case not, unpack the files from the two folders "gnuarm-armcortex. instrset_win64_xxxx" and "cmsis.instrset_common_xxxx" from the package installation files

  \Downloads\MathWorks\SupportPackages\R2017a\archives\3p

  (path may differ) and move them to the matching folders from the path mentioned in the bullet point above. Since the folders weren't existing they need to be created.

- In case those zipped files don't exist in the installation files either, uninstall the board support package, manually download the installation files, check the folders and then re-install the package.

2 Hardware Setup

The following table shows the pin-out between the STM-Discovery and the BNO055-Shuttleboard.

BNO055	STM32F746G-DISCO
VVD (1)	3V3
VVDIO (2)	3V3
GND (3)	GND
PS0 (8)	GND
PS1 (9)	GND
I2C_MODE_ADR_SEL (22)	GND
SCL (18)	D15
SCK (17)	D14

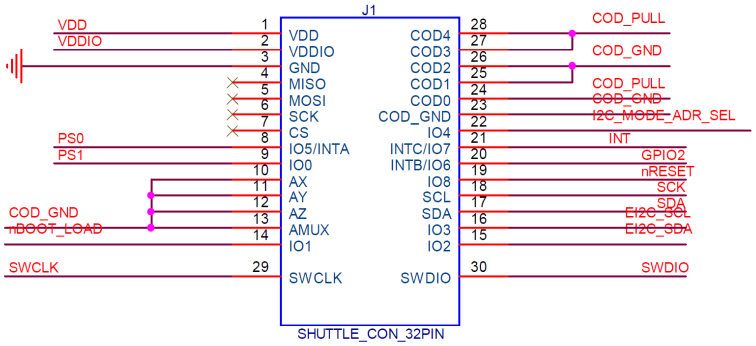

Figure 1: Pin nomenclature of the BNO055-Shuttleboard [2]

The STM-Discovery itself is powered and also programmed via a mini USB which is connected to the Laptop which runs the Matlab/Simulink environment.

2.1 BNO055 9-axis absolute Orientation Sensor

The Bosch BNO055 is composed of an accelerometer, a gyroscope and a magnetometer. All three of them offer three degrees of freedom, which is why the

device is called a "9-axis absolute Orientation Sensor". The BNO055 comes along with an on-chip microcontroller which purpose it is to run fusion algorithms at output rates up to 100 Hz. Since those fusion algorithms promise more accurate orientation data and reduce the development time, the BNO055 was chosen for the self-driving bicycle project. However, it is not decided yet if the sensor will be used in fusion mode or if the raw sensor data will be fused as part of the control algorithm which runs on the STM-Discovery. This decision is subject to future tests and validation processes.

Figure 1 shows the pin nomenclature of the BNO055-Shuttleboard. The sensor was purchased in this option because it offers the development of prototypes in a fast manner and the time-consuming design and manufacturing of a PCB can be skipped. In order to power the three sensors and the microcontroller of the BNO055, the VDD/VDDIO pins need to be hooked up to 3V3 and the GND pin to ground.

The digital interface that was chosen to enable the communication between the BNO055 and the STM-Discovery is the I2C-protocol at which the STM-Discovery represents master and the BNO055 slave. Therefor, the PS0 and PS1 pins need to be connected to ground and the SDA/SCL pins to the according STM-Discovery SDA/SCL lines. In this case, pin 22 is connected to ground which means that the I2C slave-address is 0x28. For more information on the pin-out and the I2C-protocol, it is referred to [1].

3 The Data Acquire Simulink Model

The main purpose of the data acquire process is to read out the BNO055 raw sensor data and convert it into SI-units. This section explains in detail how this process is realized in Simulink and how the model needs to be modified in order to change the sensor configuration. Furthermore it is shown how the model is deployed to the STM-Discovery and how it can be run in External Mode.

Before we have a closer look at the models subsystems and their functions it needs to be ensured that the models settings are correct and also that the model runs on the STM-Discovery. Open the "DataAcquire.slx" file, navigate to the Model Configuration Parameters dialogue box and check if the hardware implementation settings are as shown in Figure 2. Make sure to adjust the COM port for the serial communication.

3.1 Running the Model on the STM-Discovery

There are two options to deploy the model to the STM-Discovery. The first option is to run the code which was generated from the model without having the STM-Discovery giving back any data or information to the Simulink environment. That option will mainly be used when the model and its belonging C/C++ Code is fully developed and validated.

The second option is to run the model in External Mode. In this mode the STM-Discovery and the Simulink Environment can communicate via a serial connection. This comes in handy during development and validation because

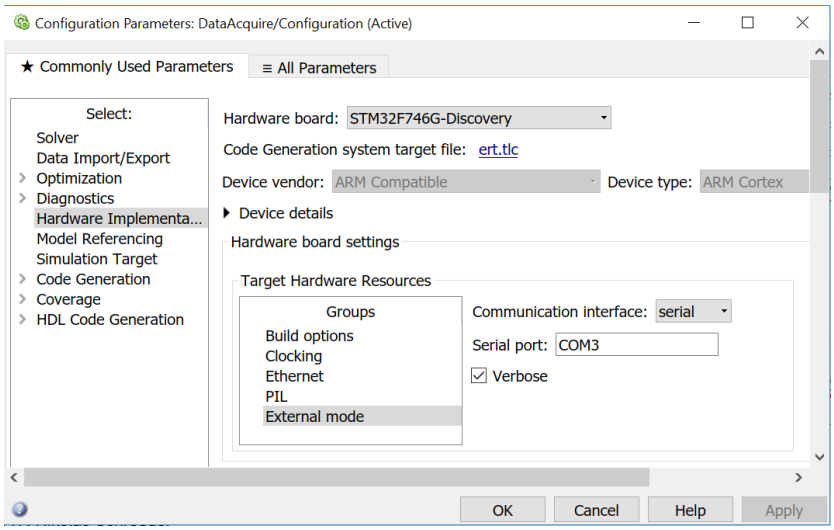

Figure 2: Configuration Parameters

it is possible to monitor and manipulate certain register values of the BNO055, including the sensor data itself, while the model is running on the STM-Disovery. That is the reason why the "DataAcquire.slx" file has many display blocks in its subsystems. Through those displays it is easy to comprehend if the generated code shows the expected behavior. Moreover the External Mode is used to conduct first system tests.

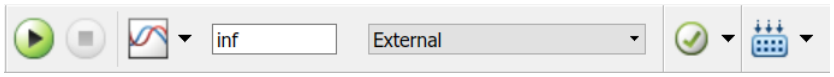

Figure 3: External Mode Setting

To run the model in External Mode make sure that "External" is selected as shown in Figure 3 and that the hardware is wired up as described in section 2. Push the "Run" button to start the execution. The model will be built, compiled and then programmed onto the STM-Discovery. Before the model is deployed to the hardware in external mode it needs to be saved. Otherwise, the most recent copy will be used and the unsaved changes are ignored.

3.2 I2C Read and Write blocks

The essence of the data acquire model is to interface the BNO055 and the STM-Discovery via I2C. Fortunately, the hardware support package for the STM-Discovery comes along with I2C write and read blocks which make it fairly easy to set up the interface. Figure 4 shows the block settings for reading out the calibration register. The two most important panels are "Slave address" and "Slave register address". The BNO055 slave address is always 0x28 (see section 2.1). The register address is nothing different than the registers offset from the slave address. A list of all registers, their addresses and their functions can be found in the BNO055 datasheet [1]. The sample time determines how frequent

the read operation is triggered.

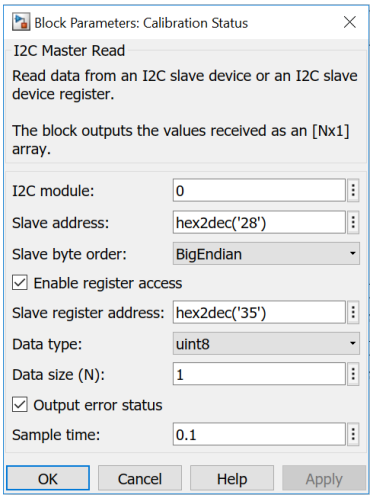

Figure 4: External Mode Setting

Figure 5 shows the blocks appearance in the Simulink environment. It has two output ports. One outputting the data in the format specified in the settings (here: uint8) and one indicating if the operation was successful. The appearance and settings of an I2C write block are exactly the same except that it has a data input port.

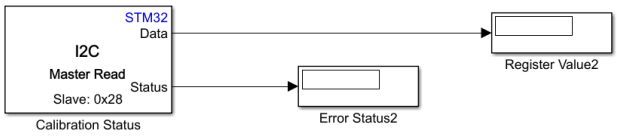

Figure 5: External Mode Setting

3.3 Status Read Outs

- Chip-ID
- Self-Test Result
- Calibration Status
- System Status
- System Error

3.4 Configuration

- Operation Mode
- Unit Selection

- Axis Remap
- Axis Map Signs

3.5 Calibration

...TBD... see section 4

3.6 Data Read Out

- Euler Read Out
- ACC Read Out
- GYRO Read Out

3.6.1 Parser Subsystem

- Bitwise Operation
- Data Type Conversion
- Scaling

4 TODO

Status: August 14th 2017

- Calibration Subsystem: Enable re-using calibration profiles. Read from and writing to offset and radius registers. How does the calibration status register behave when something is written to the offset and radius registers while in config mode? Do those values change again after changing to a fusion mode?
- In [1] it is stated that write operations are forbidden when the operation mode is any other than CONFIGMODE. ATM all I2C write operations (those are limited to the configuration subsystem so far) are executed in an infinite loop. For example, after changing to a fusion mode it is still written to the unit selection register. This results in a system error with error code 6 (register map write error). In order to avoid this it is necessary to change the model in a way that assures that registers can only be written to when the operation mode is CONFIGMODE.
- Data Logging via "Scope" or "To Workspace" Block
- What exactly means "Sample time" in the I2C write/read block settings. What's the unit, what does its value refer to?
- Test if the ACC and GYRO data registers hold different values when (a) in fusion mode or (b) in none fusion mode. Info should also be in BNO055 datasheet.

References

[1] Bosch Sensortech. *BNO055, Intelligent 9-axis absolute orientation sensor*. Bosch Sensortech, Reutlingen, revision 1.4 edition, 2016.

[2] Bosch Sensortech. *BNO055 Shuttle Board*. Bosch Sensortech, Reutlingen, version 1.0 102014 edition, 2016.