

**Intro**

The Olimexino Development board can communicate through the SPI protocol. The Olimexino is connected to an Accelerometer and Gyro Breakout board to demonstrate how to use the HANcoder SPI blocks.

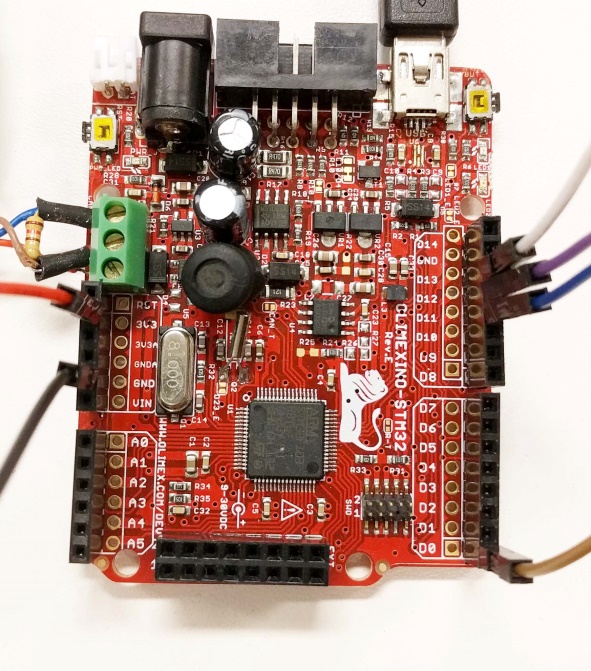
The set up contains:

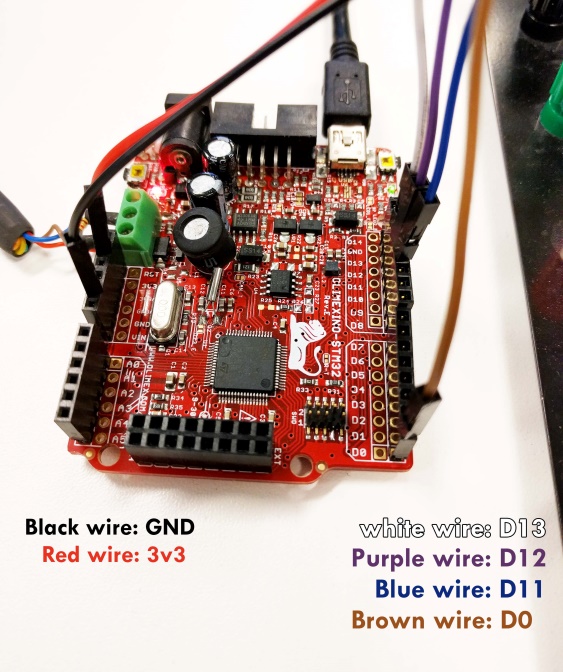
* An Olimexino STM32 Development board
* A break out board with a [MPU 6500 accelerometer and gyro chip](https://www.invensense.com/products/motion-tracking/6-axis/mpu-6500/)
* A breadboard
* A USB to mini USB cable
* Male to male pin wires

**Hook up**

**Olimexino**

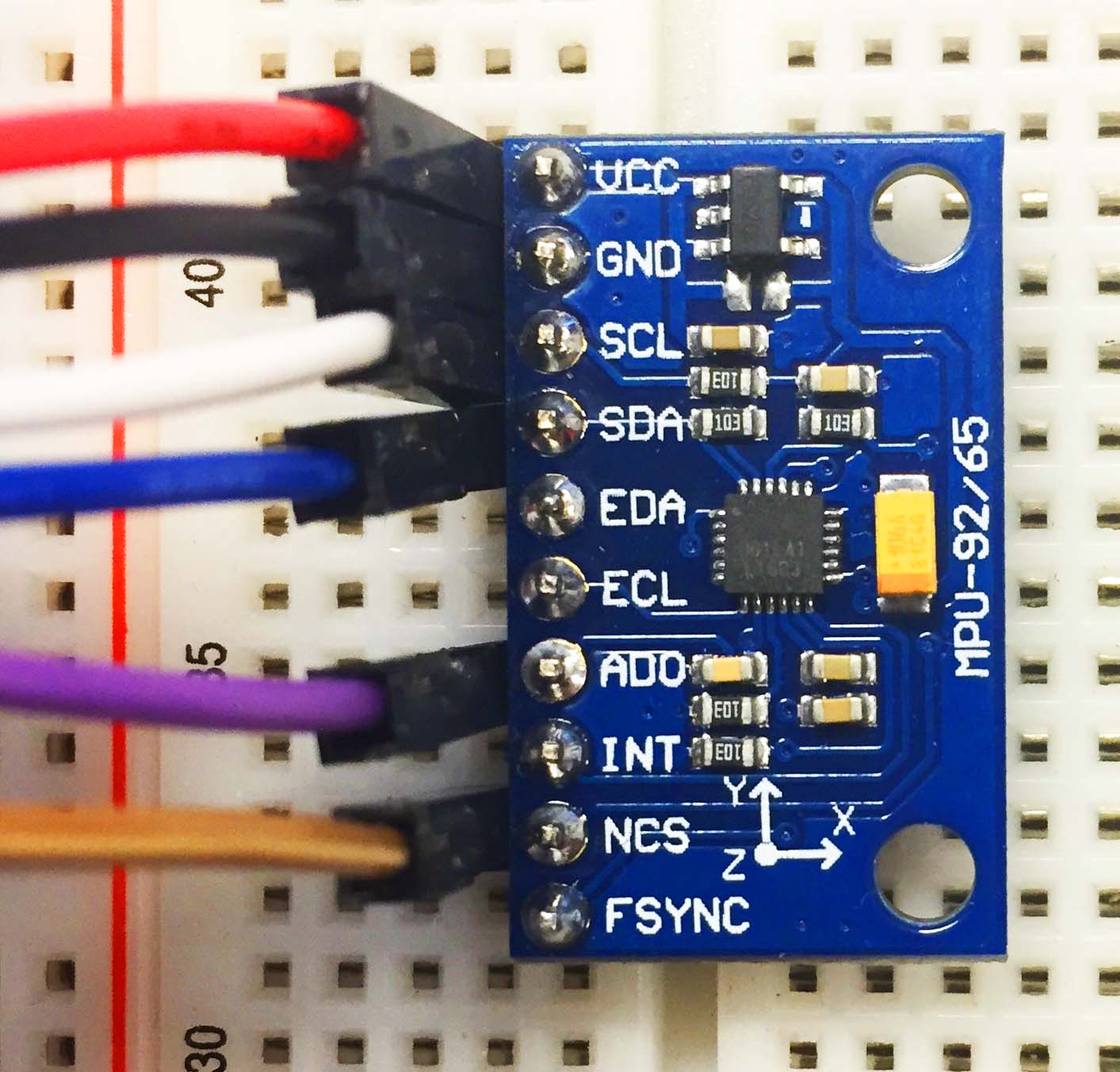
The pin wires are connected with the SPI pins, which pins depends on the SPI bus used. Simulink model. In the given example SPI bus 1 is used which uses the following pins:

* D13 Clock
* D12 MISO
* D11 MOSI
* D0 Chip/Slave Select (CS/SS)
* 3v3
* GND



**MPU 6500**

The MPU 6500 can measure the acceleration and angular rotation around 3 axis: the x,y and z. This means that is can measure 6 different movements or differently said it measures 6 degrees of freedom (6DOF)

At the MPU 6500, the pin wires are connected as following:

* Red wire 3v3 🡪 VCC – The power supply
* Black wire GND🡪 GND - Ground
* White wire D13 🡪 SCL - The Serial CLock
* Blue wire D11 🡪 SDA – Serial data input
* Purple wire D12 🡪 AD0 Serial data output
* Brown wire D0 🡪 NCS - Not Chip Select, signal low means active

Remaining slots (not used in our example):

* EDA🡪 I2C master serial data, for connecting to external sensors
* ECL 🡪I2C Master serial clock, for connecting to external sensors
* Fsync🡪 Frame synchronisation digital input.
* INT 🡪 Interrupt digital output

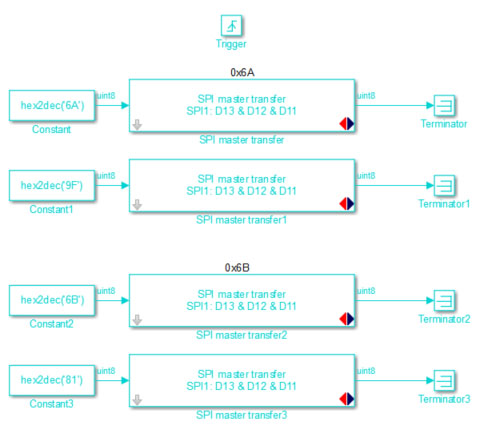
**The Simulink model**

This SPI model is made for the MPU-6500 Chip

**The model**

A signal is send to all subsystems. The signal is a counter value that goes up to 32 bits. On page 42, chapter 4.34, the [register map](https://www.invensense.com/wp-content/uploads/2015/02/MPU-6500-Register-Map2.pdf) tells us to wait 100 milliseconds between writing certain registers when using SPI.

The delay is implemented by waiting until the counter reaches a set value. After the green initialization blocks are triggered, the orange read value subsystem is activated, unlike the green initialization blocks this subsystem is kept active.

**The green subsystems**

The green subsystems are the initialization subsystems.

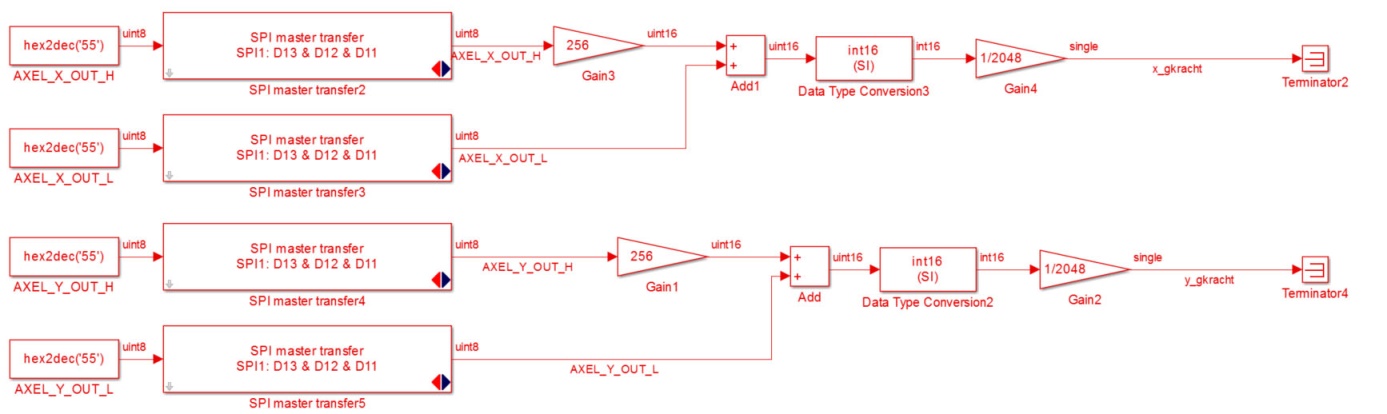
In here the register addresses are set and transferred to the right Output pins.

There are more subsystems placed and called in a specific order. This is done by adding a delay to the signal of the counter before it reaches a subsystem

**The orange subsystem**

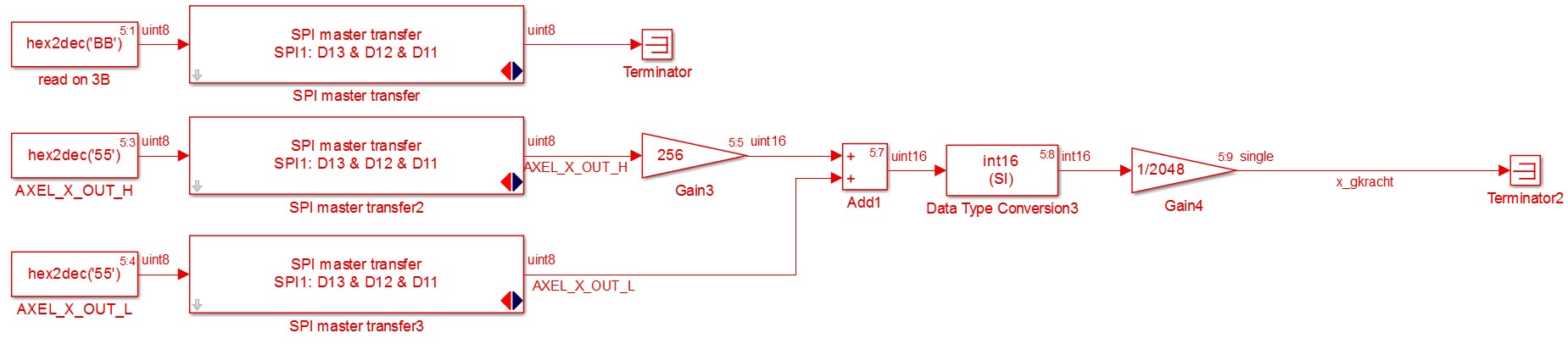
The orange subsystem is the Read subsystem

In here the outputs of the MPU-6500 chip are read and converted to the right unit

The SPI model has a priority order. This is key to SPI communication because the device expects the commands in a certain order. This is because the order in which Simulink executes the blocks in the model is not guaranteed to be from top to bottom or from left to right. By setting a priority in the blocks, Simulink will know which block to execute first.

To see the priority of the block, go to **Display 🡪 Blocks 🡪 Sorted execution order**.

As you can see in the figure below; a few numbers are shown at the right upper corner. The digits before the colon stand for the execution rank of the subsystem. The digits behind the colon stand for the execution rank of the block itself inside the subsystem.

To set priority of the block, **right click on the block 🡪 properties 🡪 priority**. In this way the execution order is set.

In SPI there are no separate send and receive blocks, instead there is a transfer block. This is because of the SPI protocol, the master always sends a byte and the slave always sends one back.

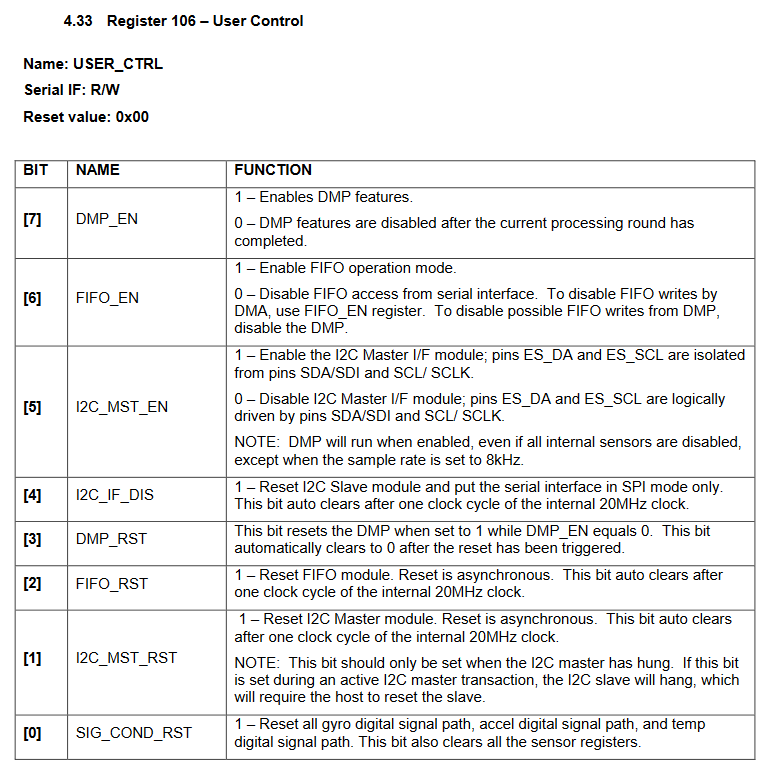
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | Read/ Write | addres | register | value |  |
|  | write | 0x6A | 106 | 0x9F |  |
|  | write | 0x6B | 107 | 0x81 |  |
|  | write | 0X68 | 104 | 0x07 |  |
|  | write | 0x6A | 106 | 0x90 |  |
|  | write | 0x6C | 108 | 0x00 |  |
|  | write | 0x19 | 25 | 0x00 |  |
|  | write | 0x1A | 26 | 0x01 |  |
|  | write | 0x1B | 27 | 0x18 |  |
|  | write | 0X1C | 28 | 0x18 |  |
|  | write | 0x1D | 29 | 0x00 |  |
|  | write | 0X37 | 55 | 0x10 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**The Set Configurations MPU-6500**

For more information about the MPU-6500, please see the [Data sheet](https://store.invensense.com/datasheets/invensense/MPU_6500_Rev1.0.pdf) and the [Register map.](https://www.invensense.com/wp-content/uploads/2015/02/MPU-6500-Register-Map2.pdf)

The MPU-6500 is set in the following configurations:

To set the configurations of the MPU the registers need a value. This value is a hexadecimal conversion of the binary code of the register. The binary code is the setting of the register. Let’s use register 106 for example:



First thing to do is decide what functions are useful. In this case we want:

* The DMP features enabled (1)
* FIFO disabled (0)
* I2C master disabled (0)
* Serial interface in SPI only (1)
* DMP reset on (1)
* FIFO reset on (1)
* I2C master reset on (1)
* Reset digital signal paths on (1)

Now we have the binary code of the register: 1001 1111

The value of the register is in hexadecimal, to convert the binary to hexadecimal see the next page:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| decimal | hexadecimal |  | decimal |  | 8 | 4 | 2 | 1 |  | 8 | 4 | 2 | 1 |
| 1 | 1 |  | Binary | % | 1 | 0 | 0 | 1 |  | 1 | 1 | 1 | 1 |
| 2 | 2 |  | total |  |  |  |  | 9 |  |  |  |  | 15 |
| 3 | 3 |  | hexadecimal | 0x |  |  |  | 9 |  |  |  |  | F |
| 4 | 4 |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | 5 |  | Value of register |  |  |  |  |  | 0x9F |  |  |  |  |
| 6 | 6 |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | 7 |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | 8 |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | 9 |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | A |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 | B |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | C |  |  |  |  |  |  |  |  |  |  |  |  |
| 13 | D |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 | E |  |  |  |  |  |  |  |  |  |  |  |  |
| 15 | F |  |  |  |  |  |  |  |  |  |  |  |  |

**HANtune layout**

Will follow....