HOCHSCHULE ULM

UNIVERSITY OF APPLIED SCIENCES

PROJECT BUMBLEBEE

TEST COLLECTION

Yvette Groner

Hannes Heinisch

Daniel Klitzke

Simon Krais

Christoph Löser

Christian Müller

30/01/2015

TABLE OF CONTENTS

1 PWM Driver

This part explains how we tested the PWM controller.

1.1 Requirements

To execute this test you need the following components:

- SoPC with a Nios II processor and the PWM controller
- Our PWM driver and the PWM test file

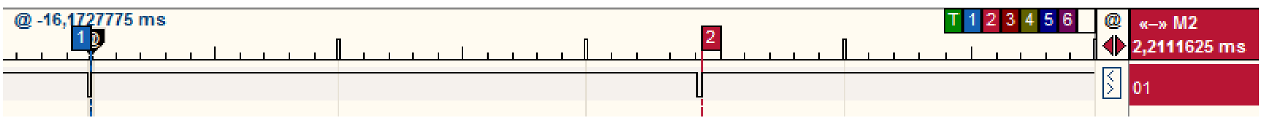
1.2 Setup the Test

First you have to setup the DE1-SoC board and program it with the current SoPC. For more information you can take a look at the Howto documentation at chapter 7. To analyze the signal and check if it's correct we used a “DigiView” logic analyzer. It is easy to use and helps to check the signal for integrity. Therefore also a Howto with setup and execution information exists, see chapter 5.

After these two steps the software can be compiled and finally run on the system. While the program runs the PWM signal always has the same period. Just the duty cycle is changed in different steps. You have to press the return key to go to the next step. The final step executes a loop which increases the duty cycle from 0% to 100%.

1.3 Results

First we checked the frequency of the PWM signal, because it was important to reach 490Hz for the motor controllers. Therefore we set the duty cycle to the maximum value and recorded it with the DigiView. The result is shown in the next graphic.



Graphic 1: Timing of highest duty cycle

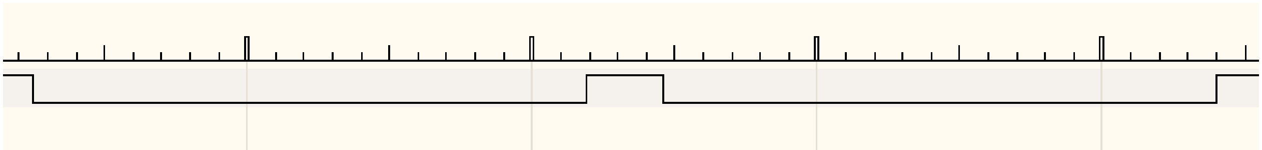
With a marker we measured the time of one period and calculated the frequency of the signal.

$$\frac{1}{2{,}2111625 * 10^{-3}} = 452{,}25079567874365$$

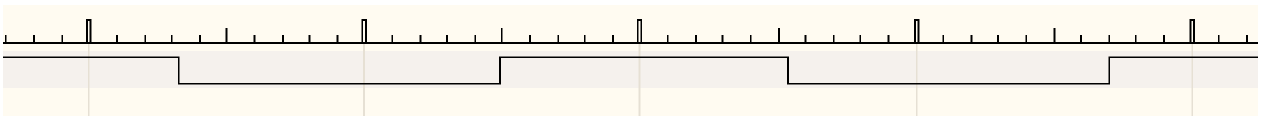
To validate this result, we also measured the period with the minimum value the result. In the following graphic is the PWM signal and the measured period time.

Graphic 2: Timing with lowest duty cycle

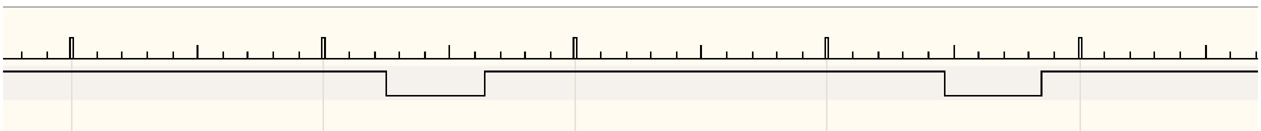
Finally we made some measurements with different duty cycles. The next graphics show the results of the different cycle steps.



Graphic 3: Test with a low duty cycle



Graphic 4: Test with a medium duty cylce



Graphic 5: Test with a high duty cycle

2 Accelerometer Driver

The accelerometer is a sensor on the 9 Degrees of Freedom Stick. It can be connected to a processor via I2C. The following chapters describe the tests we have done with the accelerometer.

2.1 Requirements

For the test the following components are necessary:

- SoPC with a Nios II processor
- Synthesized I2C controller
- I2C driver (b_i2cdriver)
- Accelerometer driver (b_accelerometerdriver)
- Test file (test_acc)

2.2 Setup

To test the accelerometer the SoPC with an I2C controller has to be synthesized into the FPGA. After this the 9DoF stick can be connected to the I2C interface of the board. You should use pull-up resistors for the SCL and SDA signals.

The software just needs a main function which calls a test function for the accelerometer. It's possible to test every single get function of the driver or to call a method to get all axis at one time. Also the 9DoF test can be used to test the accelerometer.

2.3 Running

For the test we had no reference system, to check the results. We decided to use the force of gravity, so we tested all axis pointing to the earth's core. The sensor should detect the same values for every axis. After that we turned the stick the other way around to get the negative values.

2.4 Results

The results show that the accelerometer works fine. The following table contains the results of the described test. Some values are not shown in the table to get a better summary.

X axis	Y axis	Z axis
251	1	-1
-273	1	0
0	262	0
0	-264	3
3	0	283
0	1	-231

To do a more extensive test you could use a reference system that provides an accurate measurement and in result offers reference values.

3 Gyroscope Driver

The gyroscope measures the rotation speed and is mounted on the 9 Degrees of Freedom stick. To validate the values read by the gyroscope driver a test was written and executed.

3.1 Requirements

Beside the sensor the following components are necessary:

- SoPC with a Nios II processor
- Synthesized I2C controller
- I2C driver (b_i2cdriver)
- Accelerometer driver (b_gyroscopedriver)
- Test file (test_gyro)

3.2 Setup

The system has to be programmed with a SoPC which includes the I2C controller. The SDL and SDA signals have to be mapped to an extension header to connect the sensor. You should use pull-up resistors to get a better signal.

To run the software a main function is necessary that starts the tests. For the gyroscope driver a few methods to test the functionality exist. It is possible to test every single axis on its own or all together.

3.3 Running

For the test we had no reference system, so we just checked plausibility of the values by turning it along each axis. Therefore the sensor was turned in positive axis direction. Also the negative direction was checked.

3.4 Results

The directions of the axis are correct, but the values can't been verified till now. The next table shows some measurements while the test.

X axis	Y axis	Z axis
1148	-327	-8
-1507	665	242
-4	1641	177
206	-2853	-84
749	1425	-2355
503	457	2083

The measurements where done with the 9DoF test, so the other values are also printed out. It was not possible to move just one axis.

Also the temperature value wasn't validated till now. But the test returns a value for the temperature.

4 Magnetometer Driver

For the magnetometer driver a test to read the measurement results exists.

4.1 Requirements

The test requires the following components:

- SoPC with an Nios II processor
- A synthesized I2C controller
- I2C driver (b_i2cdriver)
- Compass driver (b_CompassDriver)
- Test software (test_compass or test_9dof)

4.2 Setup

As every time the SoPC has to be synthesized into the FPGA. It is important that the I2C is wired to a header and can be connected to the 9DoF Stick. Add two pull up resistors from the SDL and SDA signals to 3.3 V for better I2C signals.

The software parts should already work together. Just a main function which calls the test functions has to be written. The test file "test_compass" has two functions, one for the raw values and a second for calibrated values. Till now just the raw function works well.

Also the test of the whole 9DoF stick can be run. It uses the raw values function and prints it out with all other values of the sensors on this stick.

4.3 Running

To check the values of the magnetometer we had no reference system. So we made the same test as with the accelerometer. We used the 9DoF test and turned the stick in reasonable directions.

We started with the x-axis and tried to get a positive value. After this we turned the stick 180 degrees to check if the value turns into negative. The same procedure was done for the other two axis.

The quality of the values could not be validated.

4.4 Results

The table below shows the results of the test. All measurements were saved into a log file, but for a better overview some values are not in the table.

X axis	Y axis	Z axis
-50	-3	9
61	-7	-4
33	90	26
44	-109	12
42	45	-53
2	1	52

The results are plausible, so the values for the different axis seem to be correct. The next step would be to validate the values.

5 ARM Linux FIFO Driver

For the FIFO driver a test exists to send and receive data.

5.1 Requirements

The test runs with the following parts

- SOPC with a Nios II processor
- realized FIFO bridge between ARM and Nios II processor
- test program for Nios II processor
- test program for Linux (ARM)

5.2 Setup

First of all, the SoPC has to be flashed into the FPGA. After that the FifoDriver has to be loaded on the Linux side. Now the test program on both sides ARM and Nios II can be started. The test programs provide to send and receive data.

5.3 Running

The test was performed by sending data from the Nios II processor and receiving the data on ARM side and vice versa.

5.4 Results

The log output of the test is shown below.

5.4.1 Transfer from ARM to Nios II

Data send by ARM: 0xf

ARM output:

```
FifoDriver_write(): packet with 4 bytes should be send:
```

```
0xf,
0x0, 0x0,
0x0, <6>
FifoDriver_write(): 1 32-bit data words have to be sent
FifoDriver_write(): to device with  minor number      = 0
FifoDriver_write():            and   FIFO base address = 0xff200000
FifoDriver_write(): word 0: 0xf will be sent
```

Nios II output:

```
FIFO DATA: 0x4
FIFO RECLV: 0x1
FIFO SENDLV: 0x0

FIFO DATA: 0xf
FIFO RECLV: 0x0
FIFO SENDLV: 0x0
```

5.4.2 Transfer from Nios II to ARM

Data send: 0x4, 0xf

Nios II output:

```
FIFO SENDLV: 0x2
```

ARM output:

```
FifoDriver.PH_FIFO_RcvTask: Packet with 0x8 bytes should be received
FifoDriver.PH_FIFO_RcvTask:   - data word 0x4 received
FifoDriver.PH_FIFO_RcvTask:   - data word 0xf received
FifoDriver.PH_FIFO_RcvTask: Packet with 8 bytes has been received
```

The log shows that sending and receiving is working with the ARM Linux FIFO driver.