Firmware Essentials e4357 Homework 2

Dwayne Dilbeck

<https://github.com/Shengliang/e4357/tree/master/spring2015/hw1>

# Homework requirements from supplied link

1. Write a C code to sample a GPIO pin and then compute the input GPIO signal frequency.
2. Write code to sample a gpio as quick as possible
3. Optional counting method for slow signals
4. Save vcd data.

# Status

## Time sheet

|  |  |
| --- | --- |
| **Task** | **Time** |
| Write code to generate a signal with PWM | 30minutes |
| Write code to sample GPIO and store to AHBSRAM | 10 Minutes |
| Create VCD from signal | 30minutes |
| Dump VCD to screen | 3 hours |
| Dump VCD to SD Card | 6 hours |
| Write code to calculate frequency | 1 hour. |
| **Total** | **11hours 10 minutes** |

# Issues/Lessons Learned

Serial pc(USBRX, USBTX) implementation needs debugging

I initially wrote a simple GPIO sampler to determine the general sample rate on a 2MHz signal.

I was surprised when the serial output reported the signal was hanging every #3 samples but the value was always being printed as 255. When I expected a value of 254 and 255 alternating. I used 3 debugging why my code was not working right, only to determine with gdb and openOCd that the value listed by p temp was always correct but the value outputted to the terminal was incorrect. I removed the Serial entries and used the generic printf. The output was now as expected.

## SD Crad library use effects sample rate

I quickly wrote a vcd content generator and decided to write the contents to a file on an SD Card. I spent a long time trying to figure out why the library caused the sampling to behave strangely. Before giving up due to lack of time and returned to terminal output.

## Compile resulted in variang sample rates

I found that from compile to compile my sample rate per 2Mhz signal was floating. Sometimes 6 samples sometimes 8 samples. I tried dropping back to a smaller sample size and a single array, but this resulted in worse sample performance.

## Do the homework in order

I read all the homework and proceeded to miss the part about calculating the input signal frequency. This was due to working on Method 4 and 2 at the same time. When I started writing this report I realized I had not completed the first part. I took an extra hour to code that solution. Based on my earlier testing I used a sample comparison to calculate frequency.

# How my solution works

My solution will take two samples per loop. I store the samples in the AHBSRAM0 and AHBSRAM1 memory locations. This allows me to collect 32K samples. Given my max rate of samples has been 12000000 per second, this gives me about 2 seconds of samples.

Since I saw variance in my sample rate, I did not hard code the ratio for the samples. Instead I capture samples from a known signal and compare the sample ratio to the signal under test.

Roughly I estimate I can sample from 3Mhz to 1HZ, where 3Mhz would allow 4 samples per period.

After the samples are taken they get processed. One function creates a vcd format. The other calculates how many samples were in the known signal period. Then the number of samples in the signal under test for one period is calculated. A ration calculation is performed to determine the sample signal frequency.

# Output

$date

$end

$version

Mbed Logic Analyzer 0.1

$end

$comment

Developed by Dwayne Dilbeck

$end

$timescale 80ns

$scope module logic $end

$var wire 1 ( data0 $end

$upscope $end

$enddefinitions

$dumpvars

0(

$end

#6

1(

#10

0(

#12

1(

#16

0(

#18

1(

#22

0(

#24

1(

#28

0(

#30

1(

Samples Per 2Mhz Reference: 6

Samples Per Sampled signal: 24357

Frequency is: 492