FIRMWARE ESSENTIALS E4357 HOMEWORK 1

Dwayne Dilbeck

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

HOMEWORK REQUIREMENTS FROM SUPPLIED LINK

REQUIREMENTS FROM TODO LIST

1. Order LPC1768
2. Buy 2 text books
3. Register mbed
4. Check-in Homework #1 to github
5. Get gmail account

HW1: C CODE TIMING ANALYSIS PART I

1. Read http://developer.mbed.org/platforms/mbed-LPC1768/
2. Skip reading: schematics and data sheets
3. Write a C program to toggle an assigned GPIO output pin (lookup yours from Grades sheet)
4. Use objdump to lookup machine code http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0337h/CHDDIGAC.html
5. Use ARM data sheet to decode machine code https://ece.uwaterloo.ca/~ece222/ARM/ARM7-TDMI-manual-pt3.pdf
6. Write a page report (include hours that you spend in the homework.) http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337h/DDI0337H_cortex_m3_r2p0_trm.pdf

REQUIREMENTS MENTIONED IN CLASS

1. Get tool chain online
2. Measure the frequency and duty cycle of the toggled pin.

STATUS

TODO LIST

1. Order LPC1768
   Complete- Previously purchased for the USCS USB Device class.
2. Buy 2 text books

   Complete - Fast and Effective Embedded Systems Design: Applying the ARM mbed, Time 0mins

   i. Obtained for the USB Device Class

   Complete - An Embedded Software Primer, David E. Simon,Time 2minutes

   ii. Obtained PDF download.
3. Register mbed
   Complete – Previously done for USB Device Class.
   http://developer.mbed.org/users/jakowisp

4. Check-in Homework #1 to github
   Complete – git@github:jakowisp/FirmwareEssentials_e4357.git

5. Get gmail account
   Complete – jakowisp.dd@gmail.com

TIME SHEET

Task	**Time**
Purchase hardware	0 mins
Purchase Text books	2 mins
Register mbed	0 mins
Register github	0 mins
Register gmail	0 mins
Deploy VM	0 minutes
Install GCC ARM Embedded	58 minutes
Verify Tool Chain	5 minutes
Install pyOCD	25 minuites
Verify pyOCD and openOCD	8 hours
Write C code	5 minutes
Objdump code and analysis	10 minutes
Capture signal waveform	2 hours
Check in to Github	30 minutes
Complete Compile and objdump using the 3 command lines given	5 minutes
Total	**12 hours 20 minutes**

C CODE TIMING ANALYSIS

SOURCE CODE

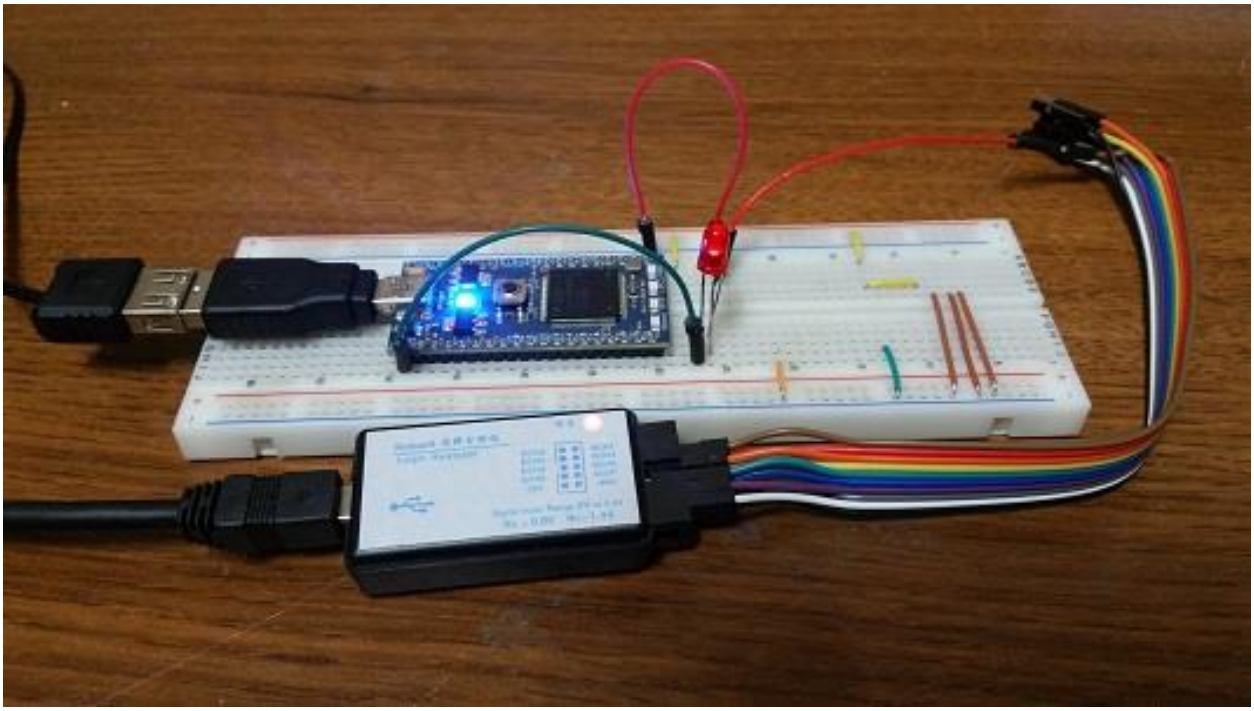
```
/* Program Example 14.1 Sets up a digital output pin using control
 * registers and flashes a led.
 * */


// define addresses of digital i/o control registers,
// as pointers to volatile data

#define FIO2DIR0  (*(volatile unsigned char *)(0x2009c040))
#define FIO2PIN0  (*(volatile unsigned char *)(0x2009c054))

int main() {
   FIO2DIR0=0x20;
   while(1) {
       FIO2PIN0 |= 0x20;
       FIO2PIN0 &= ~0x20;
   }
}
```

To fulfill the code writing requirement, the code example 14.1 was partially used. The delay function calls and declaration were removed. The pin masks were changed to make p21 be the pin toggled. (Port 2 pin5)

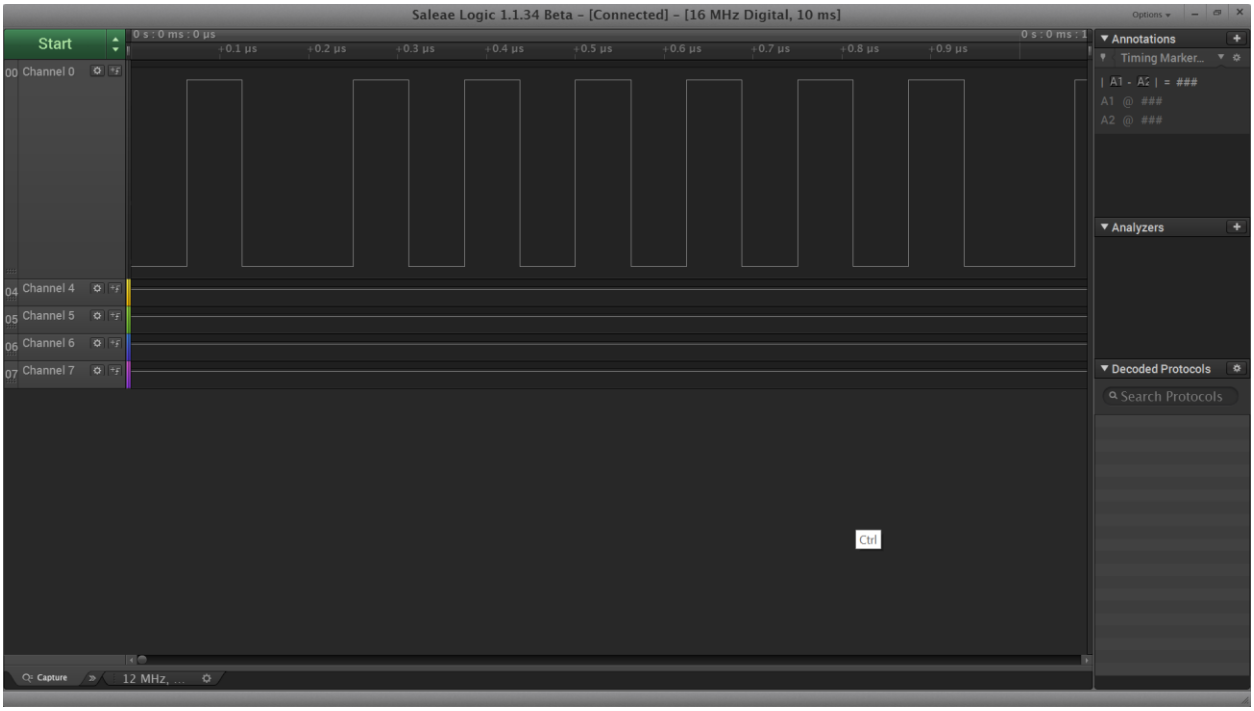
SIGNAL CAPTURE

The blinky mbed project was exported to provide the mbed library and build scripts.

The setup show below was used for capturing the data signal.



The captured logic graph is below. The duty cycle is 50%, the Frequency is 8Mhz. However multiple sampling errors exist and can be seen. I would need a more expensive logic analyzer to get more samples for better resolution.



OPT CODE DECODING

The opcode from the objdump is listed below with inserted comments to explain each line.

```
main.o:     file format elf32-littlearm
Disassembly of section .text.startup.main:
00000000 <main>:
   0:   4b05            ldr     r3, [pc, #20]   ; (18 <main+0x18>)
        //Load  r3 with 0x2009c040
   2:   2220            movs    r2, #32
       // Load r2 with 0010_0000  (Pin 5 mask)
   4:   701a            strb    r2, [r3, #0]
      //Write the configuration of the GPIO pin, storing the mask in r2 into the GPIO memory address pointed to in r3
   6:   4b05            ldr     r3, [pc, #20]   ; (1c <main+0x1c>)
       //Load  r3 with 0x2009c054
   8:   781a            ldrb    r2, [r3, #0]
       // Load the current value of the GPIO value register into r2
   a:   f042 0220       orr.w   r2, r2, #32
       // Or the pin mask 0b0010_0000 with the current GPIO value
   e:   701a            strb    r2, [r3, #0]
       //write the changed GPIO value back.
  10:   781a            ldrb    r2, [r3, #0]
       //read the current GPIO value back. (Due to Volatile setting??)
  12:   f002 02df       and.w   r2, r2, #223    ; 0xdf
       // 'AND' the 1s complement of the pin mask( clear the pin)
  16:   e7f5            b.n     4 <main+0x4>
      // jump back to setting the Pin High
  18:   2009c040        .word   0x2009c040
  1c:   2009c054        .word   0x2009c054
```

APPENDIX – VM ANMD TOOL CHAIN DETAILS

DEPLOY VM

Used a Centos 7 VM. Per my previous job I had VmWare workstation 10 on my laptop and the Centos DVD image. I already had a clean Centos VM deployed.

INSTALL ARM TOOL CHAIN: 58MINS

http://pixhawk.org/dev/toolchain_installation_lin#arm_toolchain_for_all_linux_distros

"time install.sh" was used to perform an unattended tool chain setup

```
1) yum install epel-release-7-5.noarch.rpm
2) yum update
3) yum groupinstall "Development Tools"
4) yum install python-setuptools
5) easy_install pyserial
6) easy_install pexpect
7) yum install glibc.i686 ncurses-libs.i686
8) yum install openocd libftdi-devel libftdi-python python-argparse flex
   bison-devel ncurses-devel ncurses-libs autoconf texinfo libtool zlib-
   devel
9) wget https://launchpad.net/gcc-arm-embedded/4.9/4.9-2014-q4-
   major/+download/gcc-arm-none-eabi-4_9-2014q4-20141203-linux.tar.bz2
10)tar -jxf gcc-arm-none-eabi-4_9-2014q4-20141203-linux.tar.bz2
```

VERIFY TOOL CHAIN FUNCTIONALITY.

I exported a previous mbed project as a GCC ARM project. While attempting to comp[ile a failure occurred.

Failed missing symbol __wrapper_main, solution found on mbed website.

This was determined to be cuased my an older mbed library being exported. After updating the mbed library and re exporting the project compiled correctly and executed correctly on the device.

PYOCD

PYOCD INSTALL

1) Yum install libusb-devel
2) Yum install zlib.i686
3) Tar -jxvf gcc-arm-none-eabi-misc.tar.bz2

VERIFY PYOCD DEBUG FLOW :TIME 10MINUTES

1) pyOCD_linux
    a. Mbed board found

```
[root@localhost test]# ./pyocd_linux
Welcome to the PyOCD GDB Server Beta Version
INFO:root:new board id detected: 1010bebf73074a115edd606be71ae59d0820
id => usbinfo | boardname
0 =>   (0xd28, 0x204) [lpc1768]
INFO:root:DAP SWD MODE initialised
INFO:root:IDCODE: 0x2BA01477
INFO:root:6 hardware breakpoints, 4 literal comparators
INFO:root:CPU core is Cortex-M3
INFO:root:GDB server started at port:3333
```

2) Arm-none-eabi-gdb lab1.elf
    a. Target remote local host:3333

```
INFO:root:GDB server started at port:3333
INFO:root:One client connected!
```

    b. Load

```
(gdb) load
Loading section .text, size 0x6d38 lma 0x0
Loading section .ARM.exidx, size 0x8 lma 0x6d38
Loading section .data, size 0xb4 lma 0x6d40
Start address 0x62c, load size 28148
Transfer rate: 3 KB/sec, 1655 bytes/write.
(gdb)
```

```
INFO:root:CPU core is Cortex-M3
INFO:root:GDB server started at port:3333
INFO:root:One client connected!
[====================] 100%
```

    c. B main

```
(gdb) b main
Breakpoint 1 at 0x1cc: file main.cpp, line 8.
```

    d. C
        i. Result: incorrect program execution. LEDs do not blink.

Initially this was believed to be a problem with the downloaded tool. Later trials found serveral issues.

VERIFY PYOCD #2

1) Load MBED with Blinky.bin
2) pyOCD_linux
    a. Mbed board found

```
[root@localhost test]# ./pyocd_linux
Welcome to the PyOCD GDB Server Beta Version
INFO:root:new board id detected: 1010bebf73074a115edd606be71ae59d0820
id => usbinfo | boardname
0 =>   (0xd28, 0x204) [lpc1768]
INFO:root:DAP SWD MODE initialised
INFO:root:IDCODE: 0x2BA01477
INFO:root:6 hardware breakpoints, 4 literal comparators
INFO:root:CPU core is Cortex-M3
INFO:root:GDB server started at port:3333
```

b.

3) Arm-none-eabi-gdb lab1.elf
    a. Target remote local host:3333

```
INFO:root:GDB server started at port:3333
INFO:root:One client connected!
```

    b. Mon reset halt
    c. Load
    d. Mon reset
        i. Correctly functioning application. LEDs flash in the Lab1 sequence and not in the blinky sequence.
    e. Mon reset halt
    f. B 10
    g. C
        i. Result SWD Fault

```
[====================] 100%
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/home/build/tonwan01/pyOCD/test/build/gdb_server/out00-PYZ.pyz/threading
", line 532, in __bootstrap_inner
  File "/home/build/tonwan01/pyOCD/test/build/gdb_server/out00-PYZ.pyz/pyOCD.gdb
server.gdbserver", line 167, in run
  File "/home/build/tonwan01/pyOCD/test/build/gdb_server/out00-PYZ.pyz/pyOCD.gdb
server.gdbserver", line 233, in handleMsg
  File "/home/build/tonwan01/pyOCD/test/build/gdb_server/out00-PYZ.pyz/pyOCD.gdb
server.gdbserver", line 289, in resume
  File "/home/build/tonwan01/pyOCD/test/build/gdb_server/out00-PYZ.pyz/pyOCD.tar
get.cortex_m", line 605, in resume
  File "/home/build/tonwan01/pyOCD/test/build/gdb_server/out00-PYZ.pyz/pyOCD.tar
get.cortex_m", line 362, in writeMemory
  File "/home/build/tonwan01/pyOCD/test/build/gdb_server/out00-PYZ.pyz/pyOCD.tra
nsport.cmsis_dap", line 133, in writeMem
  File "/home/build/tonwan01/pyOCD/test/build/gdb_server/out00-PYZ.pyz/pyOCD.tra
nsport.cmsis_dap_core", line 215, in dapTransfer
ValueError: SWD Fault

ERROR:root:exception during uninit
```

It should be noted that if the ‘mon reset halt’ command is issued and the pyOCD_linux is restarted functionality for Debugging will function. But the work around is clumsy.

VERIFY PYOCD #3

1) Repeat pyOCD #2 steps
2) Reconnect pyOCD_linux and gdb
    a. Target remote localhost:3333
    b. B main
    c. B 10
    d. C
        i. Result: LEDs blink in the correct order, and the specified breakpoints are triggered.

INSTALL PYOCD #2

Build attempted for pyOCD instead of using premade pyOCD_linux from GCC Arm embedded.

1. Result: pyOCD was less stable than pre-packaged.

VERIFY OPENOCD.

While investigating ways to get poyOCD to be stable after a 'mon reset halt' I came across postings comparing openOCD flash time to pyOCD.

3) Openocd –f /usr/share/openocd/scripts/board/mbed-lpc1768.cfg

```
Open On-Chip Debugger 0.8.0 (2014-04-29-12:22)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
Error: The specified debug interface was not found (cmsis-dap)
The following debug interfaces are available:
1: parport
2: dummy
3: ftdi
4: usb_blaster
5: jtag_vpi
6: amt_jtagaccel
7: gw16012
8: usbprog
9: jlink
10: vsllink
11: rlink
12: ulink
13: arm-jtag-ew
14: buspirate
15: remote_bitbang
16: hla
17: osbdm
18: opendous
19: sysfsgpio
20: aice
```

The default package for openOCD does not have the cmsis-dap support enabled.

COMPILE OPENOCD 0.8.0 AND VERIFY

The openOCD 0.8.0 installed by 'yum install openOCD' does not have the CMSIS-DAP interface enabled. This requires a recompile from source code

1) Install HIDAPI library
2) In openocd ./configure –enable-cmsis-dap
3) Make || make install
   i. Openocd –f /usr/share/openocd/scripts/board/mbed-lpc1768.cfg

```
[root@localhost pyOCD-master]# /usr/openocd/bin/openocd -f /usr/share/openocd/sc
ripts/board/mbed-lpc1768.cfg
Open On-Chip Debugger 0.8.0 (2015-04-09-04:04)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'cmsis-dap'
Info : CMSIS-DAP: SWD  Supported
Info : CMSIS-DAP: Interface Initialised (SWD)
adapter speed: 10 kHz
adapter_nsrst_delay: 200
cortex_m reset_config sysresetreq
Info : CMSIS-DAP: FW Version = 1.0
Info : SWCLK/TCK = 0 SWDIO/TMS = 1 TDI = 0 TDO = 0 nTRST = 0 nRESET = 1
Info : DAP_SWJ Sequence (reset: 50+ '1' followed by 0)
Info : CMSIS-DAP: Interface ready
Info : clock speed 10 kHz
Info : IDCODE 0x2ba01477
Info : lpc1768.cpu: hardware has 6 breakpoints, 4 watchpoints
```

   ii. Test same gdb steps as pyOCD#2 without loading new elf.
      1. Result: All break points are triggered. No SWD Faults occur.
   iii. Test elf load.
      1. Result: ELF load hangs