4357. Embedded Firmware Essentials

Homework #2

Jae Yang Park ([jaeyangp@gmail.com](mailto:jaeyangp@gmail.com))

**Source**

//

// HW#02

// Jae Yang Park

//

// Input: Pin# 12 (P0.17)

// Output: Pin# 29 (P0.5)

//

#include "mbed.h"

#include "lpc1768\_gpio.h"

#define P0\_5\_OUT (unsigned char)(1 << 5)

#define P0\_17\_IN (unsigned char)(1 << 1)

#define BIT2\_H (unsigned char)(1 << 1)

#define BIT5\_H (unsigned char)(1 << 5)

Serial pc(USBTX, USBRX);

unsigned char buffer[1024];

void toggle(void)

{

GPIO0\_FIO0DIR0 = P0\_5\_OUT; // set P0.5 to output

while (1) {

GPIO0\_FIO0SET0 |= BIT5\_H;

wait\_ms(50);

GPIO0\_FIO0CLR0 |= BIT5\_H;

wait\_ms(50);

}

}

void sampling(unsigned char \*buf)

{

register unsigned char r0, r1, r2, r3, r4, r5, r6, r7;

GPIO0\_FIO0DIR2 = 0x00;

r0 = GPIO0\_FIO0PIN2;

r1 = GPIO0\_FIO0PIN2;

r2 = GPIO0\_FIO0PIN2;

r3 = GPIO0\_FIO0PIN2;

r4 = GPIO0\_FIO0PIN2;

r5 = GPIO0\_FIO0PIN2;

r6 = GPIO0\_FIO0PIN2;

r7 = GPIO0\_FIO0PIN2;

buf[0] = r0;

buf[1] = r1;

buf[2] = r2;

buf[3] = r3;

buf[4] = r4;

buf[5] = r5;

buf[6] = r6;

buf[7] = r7;

}

void serial\_prt(void)

{

int i;

for (i = 0; i < 1024; i++) {

if (\*(buffer + i) == 0xfd)

pc.printf("0");

else

pc.printf("1");

if ((i > 0) && (i % 128) == 0) pc.printf("\n\r");

}

pc.printf("\n\r");

pc.printf("done\n\r");

}

int main(void)

{

//toggle();

unsigned char \*p = buffer;

while (p < &buffer[1023]) {

sampling(p);

p += 8;

}

serial\_prt();

return 0;

}

**Dump file (after removed mbed functions)**

-O0

LPC1768.elf: file format elf32-littlearm

Disassembly of section .text:

00000000 <g\_pfnVectors>:

0: 00 24 00 10 c5 00 00 00 00 00 00 00 00 00 00 00 .$..............

10: dd 00 00 00 dd 00 00 00 dd 00 00 00 00 00 00 00 ................

...

2c: dd 00 00 00 dd 00 00 00 00 00 00 00 dd 00 00 00 ................

3c: dd 00 00 00 dd 00 00 00 dd 00 00 00 dd 00 00 00 ................

4c: dd 00 00 00 dd 00 00 00 dd 00 00 00 dd 00 00 00 ................

5c: dd 00 00 00 dd 00 00 00 dd 00 00 00 dd 00 00 00 ................

6c: dd 00 00 00 dd 00 00 00 dd 00 00 00 dd 00 00 00 ................

7c: dd 00 00 00 dd 00 00 00 dd 00 00 00 dd 00 00 00 ................

8c: dd 00 00 00 dd 00 00 00 dd 00 00 00 dd 00 00 00 ................

9c: dd 00 00 00 dd 00 00 00 dd 00 00 00 dd 00 00 00 ................

ac: dd 00 00 00 dd 00 00 00 dd 00 00 00 dd 00 00 00 ................

bc: dd 00 00 00 dd 00 00 00 ........

000000c4 <Reset\_Handler>:

c4: 4668 mov r0, sp

c6: f020 0107 bic.w r1, r0, #7

ca: 468d mov sp, r1

cc: b501 push {r0, lr}

ce: f000 f851 bl 174 <main>

d2: e8bd 4001 ldmia.w sp!, {r0, lr}

d6: 4685 mov sp, r0

d8: 4770 bx lr

da: bf00 nop

000000dc <Default\_Handler>:

dc: e7fe b.n dc <Default\_Handler>

de: bf00 nop

000000e0 <\_Z8samplingPh>:

e0: e92d 0ff0 stmdb sp!, {r4, r5, r6, r7, r8, r9, sl, fp}

e4: b082 sub sp, #8

e6: 9001 str r0, [sp, #4]

e8: 4b20 ldr r3, [pc, #128] ; (16c <\_Z8samplingPh+0x8c>)

ea: 2200 movs r2, #0

ec: 701a strb r2, [r3, #0]

**ee: 4b20 ldr r3, [pc, #128] ; (170 <\_Z8samplingPh+0x90>)**

**f0: 781b ldrb r3, [r3, #0]**

**f2: fa5f fb83 uxtb.w fp, r3**

**f6: 4b1e ldr r3, [pc, #120] ; (170 <\_Z8samplingPh+0x90>)**

**f8: 781b ldrb r3, [r3, #0]**

**fa: fa5f fa83 uxtb.w sl, r3**

**fe: 4b1c ldr r3, [pc, #112] ; (170 <\_Z8samplingPh+0x90>)**

**100: 781b ldrb r3, [r3, #0]**

**102: fa5f f983 uxtb.w r9, r3**

**106: 4b1a ldr r3, [pc, #104] ; (170 <\_Z8samplingPh+0x90>)**

**108: 781b ldrb r3, [r3, #0]**

**10a: fa5f f883 uxtb.w r8, r3**

**10e: 4b18 ldr r3, [pc, #96] ; (170 <\_Z8samplingPh+0x90>)**

**110: 781b ldrb r3, [r3, #0]**

**112: b2df uxtb r7, r3**

**114: 4b16 ldr r3, [pc, #88] ; (170 <\_Z8samplingPh+0x90>)**

**116: 781b ldrb r3, [r3, #0]**

**118: b2de uxtb r6, r3**

**11a: 4b15 ldr r3, [pc, #84] ; (170 <\_Z8samplingPh+0x90>)**

**11c: 781b ldrb r3, [r3, #0]**

**11e: b2dd uxtb r5, r3**

**120: 4b13 ldr r3, [pc, #76] ; (170 <\_Z8samplingPh+0x90>)**

**122: 781b ldrb r3, [r3, #0]**

**124: b2dc uxtb r4, r3**

126: 9b01 ldr r3, [sp, #4]

128: 465a mov r2, fp

12a: 701a strb r2, [r3, #0]

12c: 9b01 ldr r3, [sp, #4]

12e: 3301 adds r3, #1

130: 4652 mov r2, sl

132: 701a strb r2, [r3, #0]

134: 9b01 ldr r3, [sp, #4]

136: 3302 adds r3, #2

138: 464a mov r2, r9

13a: 701a strb r2, [r3, #0]

13c: 9b01 ldr r3, [sp, #4]

13e: 3303 adds r3, #3

140: 4642 mov r2, r8

142: 701a strb r2, [r3, #0]

144: 9b01 ldr r3, [sp, #4]

146: 3304 adds r3, #4

148: 463a mov r2, r7

14a: 701a strb r2, [r3, #0]

14c: 9b01 ldr r3, [sp, #4]

14e: 3305 adds r3, #5

150: 4632 mov r2, r6

152: 701a strb r2, [r3, #0]

154: 9b01 ldr r3, [sp, #4]

156: 3306 adds r3, #6

158: 462a mov r2, r5

15a: 701a strb r2, [r3, #0]

15c: 9b01 ldr r3, [sp, #4]

15e: 3307 adds r3, #7

160: 4622 mov r2, r4

162: 701a strb r2, [r3, #0]

164: b002 add sp, #8

166: e8bd 0ff0 ldmia.w sp!, {r4, r5, r6, r7, r8, r9, sl, fp}

16a: 4770 bx lr

16c: 2009c002 .word 0x2009c002

170: 2009c016 .word 0x2009c016

00000174 <main>:

174: b500 push {lr}

176: b083 sub sp, #12

178: 4b08 ldr r3, [pc, #32] ; (19c <main+0x28>)

17a: 9301 str r3, [sp, #4]

17c: e005 b.n 18a <main+0x16>

17e: 9801 ldr r0, [sp, #4]

180: f7ff ffae bl e0 <\_Z8samplingPh>

184: 9b01 ldr r3, [sp, #4]

186: 3308 adds r3, #8

188: 9301 str r3, [sp, #4]

18a: 9b01 ldr r3, [sp, #4]

18c: 4a04 ldr r2, [pc, #16] ; (1a0 <main+0x2c>)

18e: 4293 cmp r3, r2

190: d3f5 bcc.n 17e <main+0xa>

192: 2300 movs r3, #0

194: 4618 mov r0, r3

196: b003 add sp, #12

198: f85d fb04 ldr.w pc, [sp], #4

19c: 10000000 .word 0x10000000

1a0: 100003ff .word 0x100003ff

-O3

LPC1768.elf: file format elf32-littlearm

Disassembly of section .text:

00000000 <g\_pfnVectors>:

0: 00 24 00 10 c9 00 00 00 00 00 00 00 00 00 00 00 .$..............

10: c5 00 00 00 c5 00 00 00 c5 00 00 00 00 00 00 00 ................

...

2c: c5 00 00 00 c5 00 00 00 00 00 00 00 c5 00 00 00 ................

3c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

4c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

5c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

6c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

7c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

8c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

9c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

ac: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

bc: c5 00 00 00 c5 00 00 00 ........

000000c4 <Default\_Handler>:

c4: e7fe b.n c4 <Default\_Handler>

c6: bf00 nop

000000c8 <Reset\_Handler>:

c8: 4668 mov r0, sp

ca: f020 0107 bic.w r1, r0, #7

ce: 468d mov sp, r1

d0: b501 push {r0, lr}

d2: f000 f805 bl e0 <main>

d6: e8bd 4001 ldmia.w sp!, {r0, lr}

da: 4685 mov sp, r0

dc: 4770 bx lr

de: bf00 nop

000000e0 <main>:

e0: e92d 47f0 stmdb sp!, {r4, r5, r6, r7, r8, r9, sl, lr}

e4: f04f 0a00 mov.w sl, #0

e8: 4b13 ldr r3, [pc, #76] ; (138 <main+0x58>)

ea: f8df 9054 ldr.w r9, [pc, #84] ; 140 <main+0x60>

ee: 4a13 ldr r2, [pc, #76] ; (13c <main+0x5c>)

f0: f203 38ff addw r8, r3, #1023 ; 0x3ff

f4: f889 a000 strb.w sl, [r9]

f8: f892 c000 ldrb.w ip, [r2]

fc: f892 e000 ldrb.w lr, [r2]

100: 7817 ldrb r7, [r2, #0]

102: 7816 ldrb r6, [r2, #0]

104: 7815 ldrb r5, [r2, #0]

106: 7814 ldrb r4, [r2, #0]

108: 7810 ldrb r0, [r2, #0]

10a: 7811 ldrb r1, [r2, #0]

10c: 3308 adds r3, #8

10e: f803 cc08 strb.w ip, [r3, #-8]

112: f803 ec07 strb.w lr, [r3, #-7]

116: f803 7c06 strb.w r7, [r3, #-6]

11a: f803 6c05 strb.w r6, [r3, #-5]

11e: f803 5c04 strb.w r5, [r3, #-4]

122: f803 4c03 strb.w r4, [r3, #-3]

126: f803 0c02 strb.w r0, [r3, #-2]

12a: f803 1c01 strb.w r1, [r3, #-1]

12e: 4543 cmp r3, r8

130: d3e0 bcc.n f4 <main+0x14>

132: 2000 movs r0, #0

134: e8bd 87f0 ldmia.w sp!, {r4, r5, r6, r7, r8, r9, sl, pc}

138: 10000000 .word 0x10000000

13c: 2009c016 .word 0x2009c016

140: 2009c002 .word 0x2009c002

-Os

LPC1768.elf: file format elf32-littlearm

Disassembly of section .text:

00000000 <g\_pfnVectors>:

0: 00 24 00 10 c7 00 00 00 00 00 00 00 00 00 00 00 .$..............

10: c5 00 00 00 c5 00 00 00 c5 00 00 00 00 00 00 00 ................

...

2c: c5 00 00 00 c5 00 00 00 00 00 00 00 c5 00 00 00 ................

3c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

4c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

5c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

6c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

7c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

8c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

9c: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

ac: c5 00 00 00 c5 00 00 00 c5 00 00 00 c5 00 00 00 ................

bc: c5 00 00 00 c5 00 00 00 ........

000000c4 <Default\_Handler>:

c4: e7fe b.n c4 <Default\_Handler>

000000c6 <Reset\_Handler>:

c6: 4668 mov r0, sp

c8: f020 0107 bic.w r1, r0, #7

cc: 468d mov sp, r1

ce: b501 push {r0, lr}

d0: f000 f81e bl 110 <main>

d4: e8bd 4001 ldmia.w sp!, {r0, lr}

d8: 4685 mov sp, r0

da: 4770 bx lr

000000dc <\_Z8samplingPh>:

dc: b5f0 push {r4, r5, r6, r7, lr}

de: 4b0b ldr r3, [pc, #44] ; (10c <\_Z8samplingPh+0x30>)

e0: 2200 movs r2, #0

e2: 701a strb r2, [r3, #0]

**e4: f893 e014 ldrb.w lr, [r3, #20]**

**e8: 7d1f ldrb r7, [r3, #20]**

**ea: 7d1e ldrb r6, [r3, #20]**

**ec: 7d1d ldrb r5, [r3, #20]**

**ee: 7d1c ldrb r4, [r3, #20]**

**f0: 7d19 ldrb r1, [r3, #20]**

**f2: 7d1a ldrb r2, [r3, #20]**

f4: 3314 adds r3, #20

f6: 781b ldrb r3, [r3, #0]

f8: f880 e000 strb.w lr, [r0]

fc: 7047 strb r7, [r0, #1]

fe: 7086 strb r6, [r0, #2]

100: 70c5 strb r5, [r0, #3]

102: 7104 strb r4, [r0, #4]

104: 7141 strb r1, [r0, #5]

106: 7182 strb r2, [r0, #6]

108: 71c3 strb r3, [r0, #7]

10a: bdf0 pop {r4, r5, r6, r7, pc}

10c: 2009c002 .word 0x2009c002

00000110 <main>:

110: b510 push {r4, lr}

112: 4c05 ldr r4, [pc, #20] ; (128 <main+0x18>)

114: 4620 mov r0, r4

**116: f7ff ffe1 bl dc <\_Z8samplingPh>**

11a: 4b04 ldr r3, [pc, #16] ; (12c <main+0x1c>)

11c: 3408 adds r4, #8

11e: 429c cmp r4, r3

120: d3f8 bcc.n 114 <main+0x4>

122: 2000 movs r0, #0

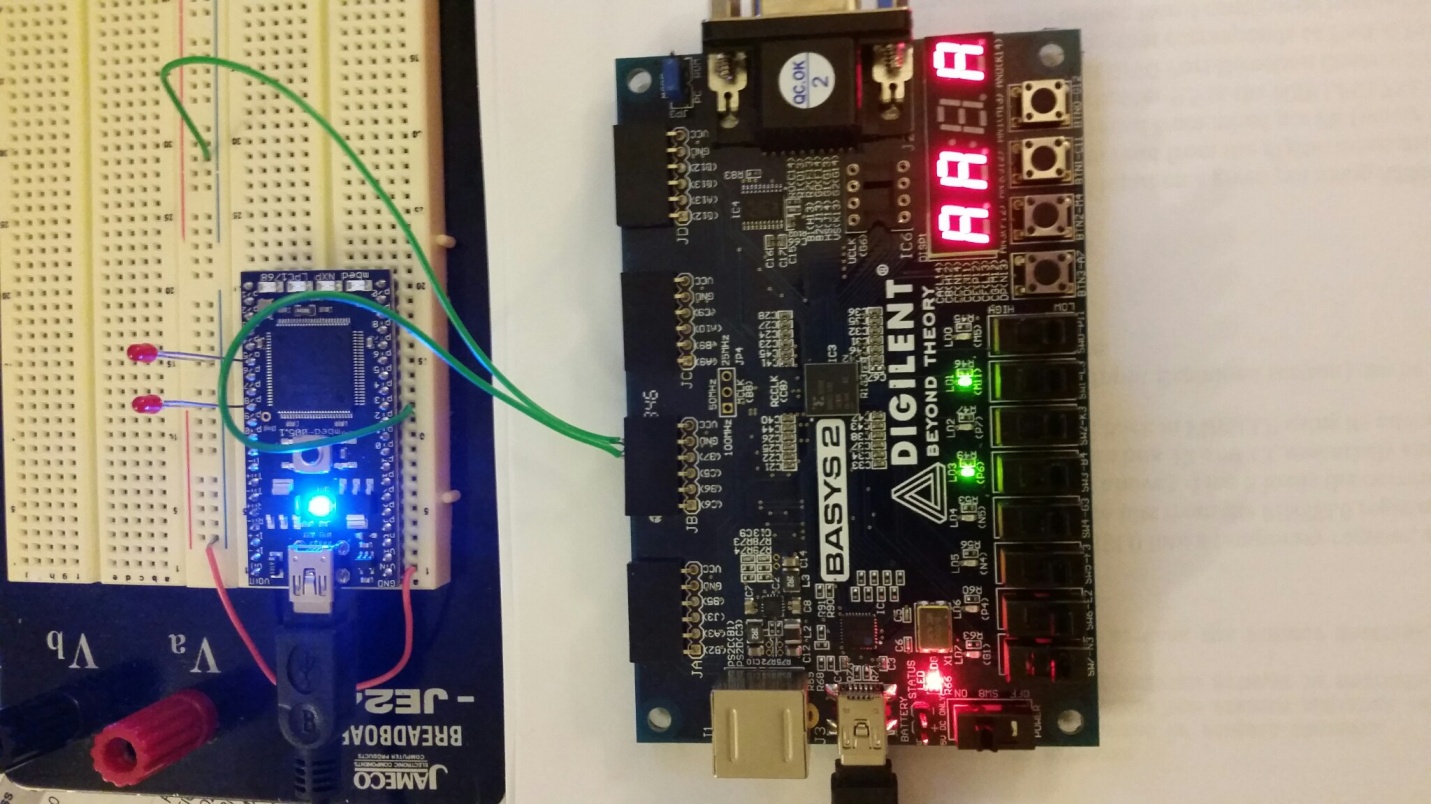
124: bd10 pop {r4, pc}

126: bf00 nop

128: 10000000 .word 0x10000000

12c: 100003ff .word 0x100003ff

**Hardware configuration**

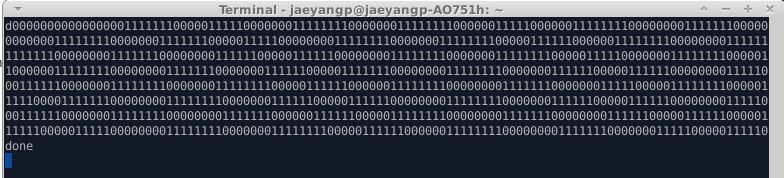


**GND**

**Pin#12 (P0.17)**

**Generating 3MHz clock signal**

**Serial console window**



**Sampling Rate calculation**

Input signal frequency: **3MHz**

# of sampling: **14 samples**

Sampling rate: **42 MHz**

In the above dump file, with **–O0** option 3 instructions are spent for 1 sampling.

However, with optimization, 1 instruction is needed for 1 sampling.

So, in the non-optimization code, sampling speed will be reduced with 30% (about 12MHz)

**Total spent hours: 17 hours**

* mbed hardware setting and testing: 5
* Signal generation and test: 5
* mbed signal sampling test: 4
* Report: 3

QUIZ#1

|  |
| --- |
| -------------------------------------------------------------------- |
| Q1: Is there any compile error with the following code (if any)? |
|  |
| unsigned int Arr[16]; |
| 3[Arr] = 7; |
|  |
| Explain: With gcc, there’s no compile error or warning.  Variable name shouldn’t be started number and number itself. |
|  |
|  |
|  |
|  |
| -------------------------------------------------------------------- |
| Q2: What is the difference between the following 3 statements? |
|  |
| const int \* px; px is pointer to int const |
| int const \* px; px is pointer to const int |
| int \* const px; px is const pointer to int |
|  |
| Is there any compile error for the following cases? |
| case1: no compile error |
| int x = 13; |
| const int \* px; |
| px = & x; |
|  |
| case 2: no compile error |
| int x = 13; |
| int const \* px; |
| px = & x; |
|  |
| case 3: compile error |
| int x = 13; |
| int \* const px; |
| px = & x; |
|  |
| Explain: px is const pointer to int. So, &x cannot be assigned to px. |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| -------------------------------------------------------------------- |
| Q3: Write a function to set or clear ith bit of a 32-bit register. |
| Where ith (0-based) := {0, 1, 2, …, 31 } |
| void reg\_set(volatile unsigned int \* pReg, int ith) |
| {  pReg = pReg | (1 << ith); |
|  |
|  |
|  |
|  |
|  |
| } |
|  |
| void reg\_clear(volatile unsigned int \* pReg, int ith) |
| {  pReg = pReg & (~(1 << ith)); |
|  |
|  |
|  |
|  |
|  |
| } |
|  |
|  |
|  |
| -------------------------------------------------------------------- |
| Q4: Write a swap function in C. |
| void swap(unsigned int \* px, unsigned int \*py)  {  unsigned int ptemp;  ptemp = \*px;  \*px = \*py;  \*py = ptemp; |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| } |
|  |
| -------------------------------------------------------------------- |
| Q5: What is the output of the following code? (Given: sizeof(unsigned int) is 4) Page 34 |
|  |
| unsigned int Arr[16]; |
| unsigned int a0 = (unsigned int) &Arr[0]; |
| unsigned int a3 = (unsigned int) &Arr[3]; |
| printf(“%d\n”, a3 – a0);  output:12 |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| QUIZ #2  -------------------------------------------------------------------- |
| Q1: How many microcontrollers in the mbed LPC1768 board?  about 5 (Cortex-M3, Flash memory controller, Power management, Interface microcontroller, Ethernet controller) |
|  |
|  |
|  |
| -------------------------------------------------------------------- |
| Q2: What is the size (in GB) of the Flash Memory ("USB Dsik") of the LPC1768?  16Mbit = 2MB = 0.002GB |
|  |
|  |
|  |
|  |
| -------------------------------------------------------------------- |
| Q3: Name 3 functions (or features) that mbed USB cable provided: |
|  |
| 1. Power supply |
| 2. USB Disk |
| 3. Serial communication |
|  |
|  |
| -------------------------------------------------------------------- |
| Q4: What is the name of the Ethernet PHY chip in the mbed board (LPC1768)?  TI DP83848J |
|  |
|  |
|  |
|  |
|  |
| -------------------------------------------------------------------- |
| Q5: Reference LPC17xx\_UM10360.pdf (Chapter 2) |
| What are the GPIO address window?  0x2009 C000 – 0x2009 FFFF |
|  |
|  |
|  |
|  |
|  |