Judah Ben-Eliezer

112352727
3/11/2021

Prelab 5:

Clock Control Module CLKCTRL and Software Delays

```c
/*
 * one_MHz.c
 *
 * Created: 3/5/2021 9:09:18 PM
 * Author : Judah Ben-Eliezer
 */

#include <avr/io.h>

#define CLKCTRL_PDIV_disable 0x00
  // mask for disabling prescalar division.

int main(void)
{
    PORTA_DIRSET = PIN7_bm;
      // CLK_OUT on PA7.
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKSEL_OSCHF_gc;
      // main clock enabled on CLKOUT, main clock set to internal high frequency
       oscillator.
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_disable;
      // prescalar division disabled.
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.MCLKCTRLC = CLKCTRL_CFDSRC_CLKMAIN_gc | CLKCTRL_CFDEN_bm;
      // clock failure source set to main clock, clock failure detection
      enabled.
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.MCLKINTCTRL = CLKCTRL_INTTYPE_INT_gc;
      // regular interrupt type.
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.OSCHFCTRLA = CLKCTRL_RUNSTBY_bm | CLKCTRL_AUTOTUNE_bm;
      // run on standby enabled, autotune enabled.

    while (1)
    {
    }
}


```

```
is_CLK_CPU_independent_of_CLK_PER.elf:     file format elf32-avr

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .data         00000000  00804000  00804000  0000018c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  1 .text         00000138  00000000  00000000  00000054  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .comment      00000030  00000000  00000000  0000018c  2**0
                  CONTENTS, READONLY
  3 .note.gnu.avr.deviceinfo 00000040  00000000  00000000  000001bc  2**2
                  CONTENTS, READONLY
  4 .debug_aranges 00000020  00000000  00000000  000001fc  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_info   00003403  00000000  00000000  0000021c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_abbrev 00002d85  00000000  00000000  0000361f  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000346  00000000  00000000  000063a4  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_frame  00000024  00000000  00000000  000066ec  2**2
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_str    000019d2  00000000  00000000  00006710  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_ranges 00000010  00000000  00000000  000080e2  2**0
                  CONTENTS, READONLY, DEBUGGING

Disassembly of section .text:

00000000 <__vectors>:
   0:   0c 94 7a 00     jmp 0xf4    ; 0xf4 <__ctors_end>
   4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
   8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
   c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  10:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  14:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  18:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  1c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  20:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  24:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  28:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  2c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  30:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  34:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  38:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  3c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  40:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  44:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
```

```
  48:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  4c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  50:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  54:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  58:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  5c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  60:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  64:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  68:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  6c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  70:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  74:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  78:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  7c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  80:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  84:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  88:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  8c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  90:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  94:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  98:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  9c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  a0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  a4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  a8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  ac:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  b0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  b4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  b8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  bc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  c0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  c4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  c8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  cc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  d0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  d4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  d8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  dc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  e0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  e4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  e8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  ec:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  f0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>

000000f4 <__ctors_end>:
  f4:   11 24           eor r1, r1
  f6:   1f be           out 0x3f, r1    ; 63
  f8:   cf ef           ldi r28, 0xFF   ; 255
  fa:   cd bf           out 0x3d, r28   ; 61
```

```
 fc:   df e7           ldi r29, 0x7F   ; 127
 fe:   de bf           out 0x3e, r29   ; 62
100:   0e 94 86 00     call    0x10c   ; 0x10c <main>
104:   0c 94 9a 00     jmp 0x134   ; 0x134 <_exit>

00000108 <__bad_interrupt>:
108:   0c 94 00 00     jmp 0   ; 0x0 <__vectors>

0000010c <main>:

#define CLKCTRL_PDIV_enable 0x01

int main(void)
{
    PORTA.DIRSET = PIN7_bm;
10c:   90 e8           ldi r25, 0x80   ; 128
10e:   90 93 01 04     sts 0x0401, r25 ; 0x800401 <__TEXT_REGION_LENGTH__
  +0x7e0401>
    CPU_CCP = CCP_IOREG_gc;
112:   88 ed           ldi r24, 0xD8   ; 216
114:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKSEL_OSCHF_gc;
          // main clock enabled on CLKOUT, main clock set to internal high
      frequency oscillator.
116:   e0 e6           ldi r30, 0x60   ; 96
118:   f0 e0           ldi r31, 0x00   ; 0
11a:   90 83           st  Z, r25
    CPU_CCP = CCP_IOREG_gc;
11c:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_enable | CLKCTRL_PDIV_4X_gc;
          // prescalar division enabled, prescaler set to 2
11e:   93 e0           ldi r25, 0x03   ; 3
120:   91 83           std Z+1, r25    ; 0x01
    CPU_CCP = CCP_IOREG_gc;
122:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKCTRLC = CLKCTRL_CFDSRC_CLKMAIN_gc | CLKCTRL_CFDEN_bm;
          // clock failure source set to main clock, clock failure detection
      enabled.
124:   91 e0           ldi r25, 0x01   ; 1
126:   92 83           std Z+2, r25    ; 0x02
    CPU_CCP = CCP_IOREG_gc;
128:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKINTCTRL = CLKCTRL_INTTYPE_INT_gc;
          // regular interrupt type.
12a:   13 82           std Z+3, r1 ; 0x03
    CPU_CCP = CCP_IOREG_gc;
12c:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.OSCHFCTRLA = CLKCTRL_RUNSTBY_bm | CLKCTRL_AUTOTUNE_bm;
          // run on standby enabled, autotune enabled.
```

```
 12e:   81 e8           ldi r24, 0x81    ; 129
 130:   80 87           std Z+8, r24     ; 0x08
 132:   ff cf           rjmp    .-2          ; 0x132 <main+0x26>

00000134 <_exit>:
 134:   f8 94           cli

00000136 <__stop_program>:
 136:   ff cf           rjmp    .-2          ; 0x136 <__stop_program>
```

```
toggle_every_xxx_us.elf:     file format elf32-avr

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .data         00000000  00804000  00804000  000001ec  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  1 .text         00000198  00000000  00000000  00000054  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .comment      00000030  00000000  00000000  000001ec  2**0
                  CONTENTS, READONLY
  3 .note.gnu.avr.deviceinfo 00000040  00000000  00000000  0000021c  2**2
                  CONTENTS, READONLY
  4 .debug_aranges 00000020  00000000  00000000  0000025c  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_info   00003506  00000000  00000000  0000027c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_abbrev 00002e26  00000000  00000000  00003782  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   000003cd  00000000  00000000  000065a8  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_frame  00000024  00000000  00000000  00006978  2**2
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_str    000019fc  00000000  00000000  0000699c  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_loc    000000a9  00000000  00000000  00008398  2**0
                  CONTENTS, READONLY, DEBUGGING
 11 .debug_ranges 00000010  00000000  00000000  00008441  2**0
                  CONTENTS, READONLY, DEBUGGING

Disassembly of section .text:

00000000 <__vectors>:
   0:   0c 94 7a 00     jmp 0xf4    ; 0xf4 <__ctors_end>
   4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
   8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
   c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  10:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  14:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  18:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  1c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  20:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  24:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  28:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  2c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  30:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  34:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  38:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  3c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
```

```
  40:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  44:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  48:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  4c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  50:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  54:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  58:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  5c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  60:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  64:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  68:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  6c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  70:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  74:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  78:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  7c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  80:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  84:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  88:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  8c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  90:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  94:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  98:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  9c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  a0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  a4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  a8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  ac:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  b0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  b4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  b8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  bc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  c0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  c4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  c8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  cc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  d0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  d4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  d8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  dc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  e0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  e4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  e8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  ec:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  f0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>

000000f4 <__ctors_end>:
  f4:   11 24           eor r1, r1
  f6:   1f be           out 0x3f, r1    ; 63
```

```
 99    f8:   cf ef           ldi r28, 0xFF   ; 255
100    fa:   cd bf           out 0x3d, r28   ; 61
101    fc:   df e7           ldi r29, 0x7F   ; 127
102    fe:   de bf           out 0x3e, r29   ; 62
103   100:   0e 94 86 00     call    0x10c   ; 0x10c <main>
104   104:   0c 94 ca 00     jmp 0x194   ; 0x194 <_exit>
105
106 00000108 <__bad_interrupt>:
107   108:   0c 94 00 00     jmp 0   ; 0x0 <__vectors>
108
109 0000010c <main>:
110 #define CLKCTRL_PDIV_disable 0x00                                       ↵
        // mask for disable prescalar
111 #define DIPS_gm 0x4F                                                    ↵
        // mask for input
112
113 int main(void)
114 {
115     PORTA.DIRSET &= DIPS_gm;                                            ↵
            // enable input on pin 7 and pin 6 of PORT A.
116   10c:   e0 e0           ldi r30, 0x00   ; 0
117   10e:   f4 e0           ldi r31, 0x04   ; 4
118   110:   81 81           ldd r24, Z+1    ; 0x01
119   112:   8f 74           andi    r24, 0x4F   ; 79
120   114:   81 83           std Z+1, r24    ; 0x01
121     PORTC.DIRSET |= PIN7_bm;                                            ↵
            // enable output on pin 7 of PORT C.
122   116:   e0 e4           ldi r30, 0x40   ; 64
123   118:   f4 e0           ldi r31, 0x04   ; 4
124   11a:   81 81           ldd r24, Z+1    ; 0x01
125   11c:   80 68           ori r24, 0x80   ; 128
126   11e:   81 83           std Z+1, r24    ; 0x01
127
128     CPU_CCP = CCP_IOREG_gc;
129   120:   88 ed           ldi r24, 0xD8   ; 216
130   122:   84 bf           out 0x34, r24   ; 52
131     CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKSEL_OSCHF_gc;            ↵
            // main clock enabled on CLKOUT, main clock set to internal high ↵
          frequency oscillator.
132   124:   e0 e6           ldi r30, 0x60   ; 96
133   126:   f0 e0           ldi r31, 0x00   ; 0
134   128:   90 e8           ldi r25, 0x80   ; 128
135   12a:   90 83           st  Z, r25
136     CPU_CCP = CCP_IOREG_gc;
137   12c:   84 bf           out 0x34, r24   ; 52
138     CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_disable;                           ↵
            // prescalar division disabled.
139   12e:   11 82           std Z+1, r1 ; 0x01
140     CPU_CCP = CCP_IOREG_gc;
```

```
130:   84 bf           out 0x34, r24   ; 52
   CLKCTRL.MCLKCTRLC = CLKCTRL_CFDSRC_CLKMAIN_gc | CLKCTRL_CFDEN_bm;
         // clock failure source set to main clock, clock failure detection
     disabled.
132:   91 e0           ldi r25, 0x01   ; 1
134:   92 83           std Z+2, r25    ; 0x02
   CPU_CCP = CCP_IOREG_gc;
136:   84 bf           out 0x34, r24   ; 52
   CLKCTRL.MCLKINTCTRL = CLKCTRL_INTTYPE_INT_gc;
         // regular interrupt type.
138:   13 82           std Z+3, r1 ; 0x03
   CPU_CCP = CCP_IOREG_gc;
13a:   84 bf           out 0x34, r24   ; 52
   CLKCTRL.OSCHFCTRLA = CLKCTRL_RUNSTBY_bm | CLKCTRL_FREQSEL_4M_gc |
     CLKCTRL_AUTOTUNE_bm;               // run on standby enabled, autotune
     enabled.
13c:   8d e8           ldi r24, 0x8D   ; 141
13e:   80 87           std Z+8, r24    ; 0x08

   while (1)
   {
       PORTD.OUT = PIN7_bm ^ PORTD.OUT;
             // toggle pin 7 of PORT C
140:   e0 e6           ldi r30, 0x60   ; 96
142:   f4 e0           ldi r31, 0x04   ; 4
144:   84 81           ldd r24, Z+4    ; 0x04
146:   80 58           subi    r24, 0x80   ; 128
148:   84 83           std Z+4, r24    ; 0x04

       uint8_t delay = (~(PORTA.IN | DIPS_gm) >> 6) & 0x03;
14a:   80 91 08 04     lds r24, 0x0408 ; 0x800408 <__TEXT_REGION_LENGTH__
  +0x7e0408>
14e:   8f 64           ori r24, 0x4F   ; 79
150:   90 e0           ldi r25, 0x00   ; 0
152:   80 95           com r24
154:   90 95           com r25
156:   08 2e           mov r0, r24
158:   89 2f           mov r24, r25
15a:   00 0c           add r0, r0
15c:   88 1f           adc r24, r24
15e:   99 0b           sbc r25, r25
160:   00 0c           add r0, r0
162:   88 1f           adc r24, r24
164:   99 1f           adc r25, r25
166:   83 70           andi    r24, 0x03   ; 3

       switch (delay) {
168:   81 30           cpi r24, 0x01   ; 1
16a:   49 f0           breq    .+18        ; 0x17e <main+0x72>
```

```
 16c:   18 f0           brcs    .+6             ; 0x174 <main+0x68>
 16e:   82 30           cpi r24, 0x02   ; 2
 170:   51 f0           breq    .+20            ; 0x186 <main+0x7a>
 172:   e6 cf           rjmp    .-52            ; 0x140 <main+0x34>
    #else
        //round up by default
        __ticks_dc = (uint32_t)(ceil(fabs(__tmp)));
    #endif

    __builtin_avr_delay_cycles(__ticks_dc);
 174:   81 e4           ldi r24, 0x41   ; 65
 176:   8a 95           dec r24
 178:   f1 f7           brne    .-4             ; 0x176 <main+0x6a>
 17a:   00 00           nop
 17c:   e1 cf           rjmp    .-62            ; 0x140 <main+0x34>
 17e:   94 e8           ldi r25, 0x84   ; 132
 180:   9a 95           dec r25
 182:   f1 f7           brne    .-4             ; 0x180 <main+0x74>
 184:   dd cf           rjmp    .-70            ; 0x140 <main+0x34>
 186:   86 ec           ldi r24, 0xC6   ; 198
 188:   90 e0           ldi r25, 0x00   ; 0
 18a:   01 97           sbiw    r24, 0x01   ; 1
 18c:   f1 f7           brne    .-4             ; 0x18a <main+0x7e>
 18e:   00 c0           rjmp    .+0             ; 0x190 <main+0x84>
 190:   00 00           nop
 192:   d6 cf           rjmp    .-84            ; 0x140 <main+0x34>

00000194 <_exit>:
 194:   f8 94           cli

00000196 <__stop_program>:
 196:   ff cf           rjmp    .-2             ; 0x196 <__stop_program>
```

```
clk_main_32768Hz.elf:     file format elf32-avr

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .data         00000000  00804000  00804000  0000018c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  1 .text         00000138  00000000  00000000  00000054  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .comment      00000030  00000000  00000000  0000018c  2**0
                  CONTENTS, READONLY
  3 .note.gnu.avr.deviceinfo 00000040  00000000  00000000  000001bc  2**2
                  CONTENTS, READONLY
  4 .debug_aranges 00000020  00000000  00000000  000001fc  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_info   00003403  00000000  00000000  0000021c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_abbrev 00002d85  00000000  00000000  0000361f  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000346  00000000  00000000  000063a4  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_frame  00000024  00000000  00000000  000066ec  2**2
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_str    000019f0  00000000  00000000  00006710  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_ranges 00000010  00000000  00000000  00008100  2**0
                  CONTENTS, READONLY, DEBUGGING

Disassembly of section .text:

00000000 <__vectors>:
   0:   0c 94 7a 00     jmp 0xf4    ; 0xf4 <__ctors_end>
   4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
   8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
   c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  10:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  14:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  18:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  1c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  20:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  24:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  28:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  2c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  30:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  34:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  38:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  3c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  40:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
  44:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
```

```
 48:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 4c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 50:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 54:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 58:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 5c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 60:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 64:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 68:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 6c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 70:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 74:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 78:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 7c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 80:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 84:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 88:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 8c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 90:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 94:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 98:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 9c:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 a0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 a4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 a8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 ac:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 b0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 b4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 b8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 bc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 c0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 c4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 c8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 cc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 d0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 d4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 d8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 dc:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 e0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 e4:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 e8:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 ec:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>
 f0:   0c 94 84 00     jmp 0x108   ; 0x108 <__bad_interrupt>

000000f4 <__ctors_end>:
 f4:   11 24           eor r1, r1
 f6:   1f be           out 0x3f, r1    ; 63
 f8:   cf ef           ldi r28, 0xFF   ; 255
 fa:   cd bf           out 0x3d, r28   ; 61
```

```
 fc:   df e7           ldi r29, 0x7F   ; 127
 fe:   de bf           out 0x3e, r29   ; 62
100:   0e 94 86 00     call    0x10c   ; 0x10c <main>
104:   0c 94 9a 00     jmp 0x134   ; 0x134 <_exit>

00000108 <__bad_interrupt>:
108:   0c 94 00 00     jmp 0   ; 0x0 <__vectors>

0000010c <main>:

#define CLKCTRL_PDIV_disable 0x00
      // mask for disable prescalar.

int main(void)
{
    PORTA.DIRSET = PIN7_bm;
          // enable output on pin 7.
10c:   80 e8           ldi r24, 0x80   ; 128
10e:   80 93 01 04     sts 0x0401, r24 ; 0x800401 <__TEXT_REGION_LENGTH__
  +0x7e0401>

    CPU_CCP = CCP_IOREG_gc;
112:   88 ed           ldi r24, 0xD8   ; 216
114:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKCTRL_CLKSEL_OSC32K_gc;
          // main clock enabled on CLKOUT, main clock set to internal 32.768
      kHz oscillator.
116:   e0 e6           ldi r30, 0x60   ; 96
118:   f0 e0           ldi r31, 0x00   ; 0
11a:   91 e8           ldi r25, 0x81   ; 129
11c:   90 83           st  Z, r25
    CPU_CCP = CCP_IOREG_gc;
11e:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_disable;
          // prescalar division disabled.
120:   11 82           std Z+1, r1 ; 0x01
    CPU_CCP = CCP_IOREG_gc;
122:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKCTRLC = CLKCTRL_CFDSRC_CLKMAIN_gc | CLKCTRL_CFDEN_bm;
          // clock failure source set to main clock, clock failure detection
      disabled.
124:   91 e0           ldi r25, 0x01   ; 1
126:   92 83           std Z+2, r25    ; 0x02
    CPU_CCP = CCP_IOREG_gc;
128:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKINTCTRL = CLKCTRL_INTTYPE_INT_gc;
          // regular interrupt type.
12a:   13 82           std Z+3, r1 ; 0x03
    CPU_CCP = CCP_IOREG_gc;
```

```
  12c:   84 bf           out 0x34, r24   ; 52
     CLKCTRL.OSCHFCTRLA = CLKCTRL_RUNSTBY_bm | CLKCTRL_FREQSEL_4M_gc |
       CLKCTRL_AUTOTUNE_bm;                // run on standby enabled, autotune
       enabled.
  12e:   8d e8           ldi r24, 0x8D   ; 141
  130:   80 87           std Z+8, r24    ; 0x08
  132:   ff cf           rjmp    .-2         ; 0x132 <main+0x26>

00000134 <_exit>:
  134:   f8 94           cli

00000136 <__stop_program>:
  136:   ff cf           rjmp    .-2         ; 0x136 <__stop_program>
```

Verification Strategy:

For part 2:
For verification of part 2, I have enabled output on pin 7 of PORT C. I have also enabled CLK_OUT on pin 7 of PORT A. By toggling pin 7 of PORT C, I will be able to deduce the cpu frequency. I will then use the logic analyzer to compare it to CLK_OUT, which I have prescaled by a factor of 2. Since the toggle is inside a while loop, each period of the toggling output will be four clock cycles. Thus, if prescaler division does not affect the CPU, the difference in frequency will be a factor of eight, otherwise it will be a factor of 4.

For part 3:
Verification is trivial, as it is only the measurement of the frequency of the toggling pin (PC 7). This I can do with the oscilloscope.

For part 4:
Same as part 3, I can measure the frequency of CLK_OUT at PA 7 with the oscilloscope.