

Judah Ben-Eliezer

112352727

3/11/2021

Prelab 5:

Clock Control Module CLKCTRL and Software Delays

```
1  /*
2   * one_MHz.c
3   *
4   * Created: 3/5/2021 9:09:18 PM
5   * Author : Judah Ben-Eliezer
6   */
7
8  #include <avr/io.h>
9
10 #define CLKCTRL_PDIV_disable 0x00  ↗
11     // mask for disabling prescalar division.
12
13 int main(void)
14 {
15     PORTA_DIRSET = PIN7_bm;  ↗
16     // CLK_OUT on PA7.
17     CPU_CCP = CCP_IOREG_gc;
18     CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKSEL_OSCHF_gc;  ↗
19     // main clock enabled on CLKOUT, main clock set to internal high frequency  ↗
20     oscillator.
21     CPU_CCP = CCP_IOREG_gc;
22     CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_disable;  ↗
23     // prescalar division disabled.
24     CPU_CCP = CCP_IOREG_gc;
25     CLKCTRL.MCLKCTRLC = CLKCTRL_CFDSRC_CLKMAIN_gc | CLKCTRL_CFDEN_bm;  ↗
26     // clock failure source set to main clock, clock failure detection  ↗
27     enabled.
28     CPU_CCP = CCP_IOREG_gc;
29     CLKCTRL.MCLKINTCTRL = CLKCTRL_INTTYPE_INT_gc;  ↗
30     // regular interrupt type.
31     CPU_CCP = CCP_IOREG_gc;
32     CLKCTRL.OSCHFCTRLA = CLKCTRL_RUNSTBY_bm | CLKCTRL_AUTOTUNE_bm;  ↗
33     // run on standby enabled, autotune enabled.
34
35     while (1)
36     {
37     }
38 }
```

```
1
2 is_CLK_CPU_independent_of_CLK_PER.elf:      file format elf32-avr
3
4 Sections:
5  Idx Name          Size      VMA      LMA      File off  Algn
6   0 .data          00000000 00804000 00804000 0000018c 2**0
7                   CONTENTS, ALLOC, LOAD, DATA
8   1 .text          00000138 00000000 00000000 00000054 2**1
9                   CONTENTS, ALLOC, LOAD, READONLY, CODE
10  2 .comment        00000030 00000000 00000000 0000018c 2**0
11                   CONTENTS, READONLY
12  3 .note.gnu.avr.deviceinfo 00000040 00000000 00000000 000001bc 2**2
13                   CONTENTS, READONLY
14  4 .debug_aranges  00000020 00000000 00000000 000001fc 2**0
15                   CONTENTS, READONLY, DEBUGGING
16  5 .debug_info      00003403 00000000 00000000 0000021c 2**0
17                   CONTENTS, READONLY, DEBUGGING
18  6 .debug_abbrev    00002d85 00000000 00000000 0000361f 2**0
19                   CONTENTS, READONLY, DEBUGGING
20  7 .debug_line      00000346 00000000 00000000 000063a4 2**0
21                   CONTENTS, READONLY, DEBUGGING
22  8 .debug_frame     00000024 00000000 00000000 000066ec 2**2
23                   CONTENTS, READONLY, DEBUGGING
24  9 .debug_str       000019d2 00000000 00000000 00006710 2**0
25                   CONTENTS, READONLY, DEBUGGING
26 10 .debug_ranges    00000010 00000000 00000000 000080e2 2**0
27                   CONTENTS, READONLY, DEBUGGING
28
29 Disassembly of section .text:
30
31 00000000 <__vectors>:
32  0:  0c 94 7a 00      jmp 0xf4      ; 0xf4 <__ctors_end>
33  4:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
34  8:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
35  c:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
36 10:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
37 14:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
38 18:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
39 1c:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
40 20:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
41 24:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
42 28:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
43 2c:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
44 30:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
45 34:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
46 38:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
47 3c:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
48 40:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
49 44:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
```

```

50  48:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
51  4c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
52  50:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
53  54:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
54  58:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
55  5c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
56  60:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
57  64:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
58  68:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
59  6c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
60  70:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
61  74:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
62  78:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
63  7c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
64  80:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
65  84:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
66  88:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
67  8c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
68  90:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
69  94:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
70  98:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
71  9c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
72  a0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
73  a4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
74  a8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
75  ac:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
76  b0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
77  b4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
78  b8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
79  bc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
80  c0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
81  c4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
82  c8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
83  cc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
84  d0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
85  d4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
86  d8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
87  dc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
88  e0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
89  e4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
90  e8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
91  ec:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
92  f0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
93
94  000000f4 <__ctors_end>:
95  f4:  11 24          eor r1, r1
96  f6:  1f be          out 0x3f, r1      ; 63
97  f8:  cf ef          ldi r28, 0xFF      ; 255
98  fa:  cd bf          out 0x3d, r28     ; 61

```

```

99  fc:  df e7          ldi r29, 0x7F    ; 127
100  fe:  de bf          out 0x3e, r29    ; 62
101  100: 0e 94 86 00    call 0x10c      ; 0x10c <main>
102  104: 0c 94 9a 00    jmp 0x134    ; 0x134 <_exit>
103
104  00000108 <__bad_interrupt>:
105  108: 0c 94 00 00    jmp 0      ; 0x0 <__vectors>
106
107  0000010c <main>:
108
109  #define CLKCTRL_PDIV_enable 0x01
110
111  int main(void)
112  {
113      PORTA.DIRSET = PIN7_bm;
114  10c: 90 e8          ldi r25, 0x80    ; 128
115  10e: 90 93 01 04    sts 0x0401, r25 ; 0x800401 <__TEXT_REGION_LENGTH__
+0x7e0401>
116      CPU_CCP = CCP_IOREG_gc;
117  112: 88 ed          ldi r24, 0xD8    ; 216
118  114: 84 bf          out 0x34, r24    ; 52
119      CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKSEL_OSCHF_gc;
// main clock enabled on CLKOUT, main clock set to internal high
frequency oscillator.
120  116: e0 e6          ldi r30, 0x60    ; 96
121  118: f0 e0          ldi r31, 0x00    ; 0
122  11a: 90 83          st Z, r25
123      CPU_CCP = CCP_IOREG_gc;
124  11c: 84 bf          out 0x34, r24    ; 52
125      CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_enable | CLKCTRL_PDIV_4X_gc;
// prescaler division enabled, prescaler set to 2
126  11e: 93 e0          ldi r25, 0x03    ; 3
127  120: 91 83          std Z+1, r25    ; 0x01
128      CPU_CCP = CCP_IOREG_gc;
129  122: 84 bf          out 0x34, r24    ; 52
130      CLKCTRL.MCLKCTRLC = CLKCTRL_CFD SRC_CLKMAIN_gc | CLKCTRL_CFDEN_bm;
// clock failure source set to main clock, clock failure detection
enabled.
131  124: 91 e0          ldi r25, 0x01    ; 1
132  126: 92 83          std Z+2, r25    ; 0x02
133      CPU_CCP = CCP_IOREG_gc;
134  128: 84 bf          out 0x34, r24    ; 52
135      CLKCTRL.MCLKINTCTRL = CLKCTRL_INTTYPE_INT_gc;
// regular interrupt type.
136  12a: 13 82          std Z+3, r1 ; 0x03
137      CPU_CCP = CCP_IOREG_gc;
138  12c: 84 bf          out 0x34, r24    ; 52
139      CLKCTRL.OSCHFCTRLA = CLKCTRL_RUNSTBY_bm | CLKCTRL_AUTOTUNE_bm;
// run on standby enabled, autotune enabled.

```

```
140 12e: 81 e8          ldi r24, 0x81    ; 129
141 130: 80 87          std Z+8, r24     ; 0x08
142 132: ff cf          rjmp  .-2          ; 0x132 <main+0x26>
143
144 00000134 <_exit>:
145 134: f8 94          cli
146
147 00000136 <__stop_program>:
148 136: ff cf          rjmp  .-2          ; 0x136 <__stop_program>
149
```

```
1
2 toggle_every_xxx_us.elf:      file format elf32-avr
3
4 Sections:
5 Idx Name                      Size      VMA      LMA      File off  Algn
6  0 .data                      00000000 00804000 00804000 000001ec 2**0
7                               CONTENTS, ALLOC, LOAD, DATA
8  1 .text                      00000198 00000000 00000000 00000054 2**1
9                               CONTENTS, ALLOC, LOAD, READONLY, CODE
10  2 .comment                   00000030 00000000 00000000 000001ec 2**0
11                               CONTENTS, READONLY
12  3 .note.gnu.avr.deviceinfo 00000040 00000000 00000000 00000000 0000021c 2**2
13                               CONTENTS, READONLY
14  4 .debug_aranges             00000020 00000000 00000000 0000025c 2**0
15                               CONTENTS, READONLY, DEBUGGING
16  5 .debug_info                00003506 00000000 00000000 0000027c 2**0
17                               CONTENTS, READONLY, DEBUGGING
18  6 .debug_abbrev              00002e26 00000000 00000000 00003782 2**0
19                               CONTENTS, READONLY, DEBUGGING
20  7 .debug_line                000003cd 00000000 00000000 000065a8 2**0
21                               CONTENTS, READONLY, DEBUGGING
22  8 .debug_frame               00000024 00000000 00000000 00006978 2**2
23                               CONTENTS, READONLY, DEBUGGING
24  9 .debug_str                 000019fc 00000000 00000000 0000699c 2**0
25                               CONTENTS, READONLY, DEBUGGING
26 10 .debug_loc                 000000a9 00000000 00000000 00008398 2**0
27                               CONTENTS, READONLY, DEBUGGING
28 11 .debug_ranges              00000010 00000000 00000000 00008441 2**0
29                               CONTENTS, READONLY, DEBUGGING
30
31 Disassembly of section .text:
32
33 00000000 <__vectors>:
34  0:  0c 94 7a 00      jmp 0xf4      ; 0xf4 <__ctors_end>
35  4:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
36  8:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
37  c:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
38 10:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
39 14:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
40 18:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
41 1c:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
42 20:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
43 24:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
44 28:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
45 2c:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
46 30:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
47 34:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
48 38:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
49 3c:  0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
```

```

50  40:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
51  44:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
52  48:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
53  4c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
54  50:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
55  54:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
56  58:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
57  5c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
58  60:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
59  64:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
60  68:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
61  6c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
62  70:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
63  74:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
64  78:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
65  7c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
66  80:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
67  84:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
68  88:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
69  8c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
70  90:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
71  94:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
72  98:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
73  9c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
74  a0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
75  a4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
76  a8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
77  ac:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
78  b0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
79  b4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
80  b8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
81  bc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
82  c0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
83  c4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
84  c8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
85  cc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
86  d0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
87  d4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
88  d8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
89  dc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
90  e0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
91  e4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
92  e8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
93  ec:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
94  f0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
95
96  000000f4 <__ctors_end>:
97  f4:  11 24          eor r1, r1
98  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
111 // mask for disable prescaler
112 #define DIPS_gm 0x4F
113 // mask for input
114
115 int main(void)
116 {
117     PORTA.DIRSET &= DIPS_gm;
118     // enable input on pin 7 and pin 6 of PORT A.
119 10c:  e0 e0      ldi r30, 0x00    ; 0
120 10e:  f4 e0      ldi r31, 0x04    ; 4
121 110:  81 81      ldd r24, Z+1      ; 0x01
122 112:  8f 74      andi r24, 0x4F    ; 79
123 114:  81 83      std Z+1, r24    ; 0x01
124     PORTC.DIRSET |= PIN7_bm;
125     // enable output on pin 7 of PORT C.
126 116:  e0 e4      ldi r30, 0x40    ; 64
127 118:  f4 e0      ldi r31, 0x04    ; 4
128 11a:  81 81      ldd r24, Z+1      ; 0x01
129 11c:  80 68      ori r24, 0x80    ; 128
130 11e:  81 83      std Z+1, r24    ; 0x01
131
132     CPU_CCP = CCP_IOREG_gc;
133 120:  88 ed      ldi r24, 0xD8    ; 216
134 122:  84 bf      out 0x34, r24    ; 52
135     CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKSEL_OSCHF_gc;
136     // main clock enabled on CLKOUT, main clock set to internal high
137     frequency oscillator.
138 124:  e0 e6      ldi r30, 0x60    ; 96
139 126:  f0 e0      ldi r31, 0x00    ; 0
140 128:  90 e8      ldi r25, 0x80    ; 128
141 12a:  90 83      st Z, r25
142     CPU_CCP = CCP_IOREG_gc;
143 12c:  84 bf      out 0x34, r24    ; 52
144     CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_disable;
145     // prescaler division disabled.
146 12e:  11 82      std Z+1, r1      ; 0x01
147     CPU_CCP = CCP_IOREG_gc;

```

```

..._xxx_us\toggle_every_xxx_us\Debug\toggle_every_xxx_us.lss 4
141 130: 84 bf          out 0x34, r24      ; 52
142      CLKCTRL.MCLKCTRLC = CLKCTRL_CFD SRC_CLKMAIN_gc | CLKCTRL_CFDEN_bm;
      // clock failure source set to main clock, clock failure detection
      disabled.
143 132: 91 e0          ldi r25, 0x01      ; 1
144 134: 92 83          std Z+2, r25       ; 0x02
145      CPU_CCP = CCP_IOREG_gc;
146 136: 84 bf          out 0x34, r24      ; 52
147      CLKCTRL.MCLKINTCTRL = CLKCTRL_INTTYPE_INT_gc;
      // regular interrupt type.
148 138: 13 82          std Z+3, r1 ; 0x03
149      CPU_CCP = CCP_IOREG_gc;
150 13a: 84 bf          out 0x34, r24      ; 52
151      CLKCTRL.OSCHFCTRLA = CLKCTRL_RUNSTBY_bm | CLKCTRL_FREQSEL_4M_gc |
      CLKCTRL_AUTOTUNE_bm;          // run on standby enabled, autotune
      enabled.
152 13c: 8d e8          ldi r24, 0x8D      ; 141
153 13e: 80 87          std Z+8, r24       ; 0x08
154
155      while (1)
156      {
157          PORTD.OUT = PIN7_bm ^ PORTD.OUT;
          // toggle pin 7 of PORT C
158 140: e0 e6          ldi r30, 0x60      ; 96
159 142: f4 e0          ldi r31, 0x04      ; 4
160 144: 84 81          ldd r24, Z+4       ; 0x04
161 146: 80 58          subi r24, 0x80     ; 128
162 148: 84 83          std Z+4, r24       ; 0x04
163
164          uint8_t delay = (~(PORTA.IN | DIPS_gm) >> 6) & 0x03;
165 14a: 80 91 08 04     lds r24, 0x0408 ; 0x800408 <__TEXT_REGION_LENGTH__
      +0x7e0408>
166 14e: 8f 64          ori r24, 0x4F      ; 79
167 150: 90 e0          ldi r25, 0x00      ; 0
168 152: 80 95          com r24
169 154: 90 95          com r25
170 156: 08 2e          mov r0, r24
171 158: 89 2f          mov r24, r25
172 15a: 00 0c          add r0, r0
173 15c: 88 1f          adc r24, r24
174 15e: 99 0b          sbc r25, r25
175 160: 00 0c          add r0, r0
176 162: 88 1f          adc r24, r24
177 164: 99 1f          adc r25, r25
178 166: 83 70          andi r24, 0x03     ; 3
179
180      switch (delay) {
181 168: 81 30          cpi r24, 0x01      ; 1
182 16a: 49 f0          breq .+18          ; 0x17e <main+0x72>

```

```

183 16c: 18 f0      brcs    .+6          ; 0x174 <main+0x68>
184 16e: 82 30      cpi    r24, 0x02      ; 2
185 170: 51 f0      breq    .+20          ; 0x186 <main+0x7a>
186 172: e6 cf      rjmp    .-52         ; 0x140 <main+0x34>
187      #else
188          //round up by default
189          __ticks_dc = (uint32_t)(ceil(fabs(__tmp)));
190      #endif
191
192      __builtin_avr_delay_cycles(__ticks_dc);
193 174: 81 e4      ldi    r24, 0x41      ; 65
194 176: 8a 95      dec    r24
195 178: f1 f7      brne    .-4          ; 0x176 <main+0x6a>
196 17a: 00 00      nop
197 17c: e1 cf      rjmp    .-62         ; 0x140 <main+0x34>
198 17e: 94 e8      ldi    r25, 0x84      ; 132
199 180: 9a 95      dec    r25
200 182: f1 f7      brne    .-4          ; 0x180 <main+0x74>
201 184: dd cf      rjmp    .-70         ; 0x140 <main+0x34>
202 186: 86 ec      ldi    r24, 0xC6      ; 198
203 188: 90 e0      ldi    r25, 0x00      ; 0
204 18a: 01 97      sbiw    r24, 0x01     ; 1
205 18c: f1 f7      brne    .-4          ; 0x18a <main+0x7e>
206 18e: 00 c0      rjmp    .+0          ; 0x190 <main+0x84>
207 190: 00 00      nop
208 192: d6 cf      rjmp    .-84         ; 0x140 <main+0x34>
209
210 00000194 <_exit>:
211 194: f8 94      cli
212
213 00000196 <__stop_program>:
214 196: ff cf      rjmp    .-2          ; 0x196 <__stop_program>
215

```

```

1
2 clk_main_32768Hz.elf:      file format elf32-avr
3
4 Sections:
5  Idx Name                  Size      VMA      LMA      File off  Algn
6    0 .data                 00000000 00804000 00804000 0000018c 2**0
7                          CONTENTS, ALLOC, LOAD, DATA
8    1 .text                 00000138 00000000 00000000 00000054 2**1
9                          CONTENTS, ALLOC, LOAD, READONLY, CODE
10   2 .comment              00000030 00000000 00000000 0000018c 2**0
11                          CONTENTS, READONLY
12   3 .note.gnu.avr.deviceinfo 00000040 00000000 00000000 00000000 000001bc 2**2
13                          CONTENTS, READONLY
14   4 .debug_aranges        00000020 00000000 00000000 000001fc 2**0
15                          CONTENTS, READONLY, DEBUGGING
16   5 .debug_info           00003403 00000000 00000000 0000021c 2**0
17                          CONTENTS, READONLY, DEBUGGING
18   6 .debug_abbrev         00002d85 00000000 00000000 0000361f 2**0
19                          CONTENTS, READONLY, DEBUGGING
20   7 .debug_line           00000346 00000000 00000000 000063a4 2**0
21                          CONTENTS, READONLY, DEBUGGING
22   8 .debug_frame          00000024 00000000 00000000 000066ec 2**2
23                          CONTENTS, READONLY, DEBUGGING
24   9 .debug_str            000019f0 00000000 00000000 00006710 2**0
25                          CONTENTS, READONLY, DEBUGGING
26  10 .debug_ranges         00000010 00000000 00000000 00008100 2**0
27                          CONTENTS, READONLY, DEBUGGING
28
29 Disassembly of section .text:
30
31 00000000 <__vectors>:
32  0: 0c 94 7a 00      jmp 0xf4      ; 0xf4 <__ctors_end>
33  4: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
34  8: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
35  c: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
36 10: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
37 14: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
38 18: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
39 1c: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
40 20: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
41 24: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
42 28: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
43 2c: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
44 30: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
45 34: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
46 38: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
47 3c: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
48 40: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>
49 44: 0c 94 84 00      jmp 0x108     ; 0x108 <__bad_interrupt>

```

```

50  48:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
51  4c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
52  50:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
53  54:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
54  58:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
55  5c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
56  60:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
57  64:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
58  68:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
59  6c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
60  70:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
61  74:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
62  78:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
63  7c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
64  80:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
65  84:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
66  88:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
67  8c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
68  90:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
69  94:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
70  98:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
71  9c:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
72  a0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
73  a4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
74  a8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
75  ac:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
76  b0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
77  b4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
78  b8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
79  bc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
80  c0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
81  c4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
82  c8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
83  cc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
84  d0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
85  d4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
86  d8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
87  dc:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
88  e0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
89  e4:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
90  e8:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
91  ec:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
92  f0:  0c 94 84 00      jmp 0x108      ; 0x108 <__bad_interrupt>
93
94  000000f4 <__ctors_end>:
95  f4:  11 24          eor r1, r1
96  f6:  1f be          out 0x3f, r1      ; 63
97  f8:  cf ef          ldi r28, 0xFF      ; 255
98  fa:  cd bf          out 0x3d, r28    ; 61

```

```

99  fc:  df e7          ldi r29, 0x7F    ; 127
100 fe:  de bf          out 0x3e, r29    ; 62
101 100: 0e 94 86 00    call 0x10c      ; 0x10c <main>
102 104: 0c 94 9a 00    jmp 0x134    ; 0x134 <_exit>
103
104 00000108 <__bad_interrupt>:
105 108: 0c 94 00 00    jmp 0      ; 0x0 <__vectors>
106
107 0000010c <main>:
108
109 #define CLKCTRL_PDIV_disable 0x00
    // mask for disable prescaler.
110
111 int main(void)
112 {
113     PORTA.DIRSET = PIN7_bm;
    // enable output on pin 7.
114 10c: 80 e8          ldi r24, 0x80    ; 128
115 10e: 80 93 01 04    sts 0x0401, r24 ; 0x800401 <__TEXT_REGION_LENGTH__
    +0x7e0401>
116
117     CPU_CCP = CCP_IOREG_gc;
118 112: 88 ed          ldi r24, 0xD8    ; 216
119 114: 84 bf          out 0x34, r24    ; 52
120     CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKCTRL_CLKSEL_OSC32K_gc;
    // main clock enabled on CLKOUT, main clock set to internal 32.768
    kHz oscillator.
121 116: e0 e6          ldi r30, 0x60    ; 96
122 118: f0 e0          ldi r31, 0x00    ; 0
123 11a: 91 e8          ldi r25, 0x81    ; 129
124 11c: 90 83          st Z, r25
125     CPU_CCP = CCP_IOREG_gc;
126 11e: 84 bf          out 0x34, r24    ; 52
127     CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_disable;
    // prescaler division disabled.
128 120: 11 82          std Z+1, r1 ; 0x01
129     CPU_CCP = CCP_IOREG_gc;
130 122: 84 bf          out 0x34, r24    ; 52
131     CLKCTRL.MCLKCTRLC = CLKCTRL_CFD SRC_CLKMAIN_gc | CLKCTRL_CFDEN_bm;
    // clock failure source set to main clock, clock failure detection
    disabled.
132 124: 91 e0          ldi r25, 0x01    ; 1
133 126: 92 83          std Z+2, r25    ; 0x02
134     CPU_CCP = CCP_IOREG_gc;
135 128: 84 bf          out 0x34, r24    ; 52
136     CLKCTRL.MCLKINTCTRL = CLKCTRL_INTTYPE_INT_gc;
    // regular interrupt type.
137 12a: 13 82          std Z+3, r1 ; 0x03
138     CPU_CCP = CCP_IOREG_gc;

```

```
139 12c: 84 bf          out 0x34, r24      ; 52
140      CLKCTRL.OSCHFCTRLA = CLKCTRL_RUNSTBY_bm | CLKCTRL_FREQSEL_4M_gc |
      CLKCTRL_AUTOTUNE_bm;          // run on standby enabled, autotune
      enabled.
141 12e: 8d e8          ldi r24, 0x8D      ; 141
142 130: 80 87          std Z+8, r24      ; 0x08
143 132: ff cf          rjmp    .-2          ; 0x132 <main+0x26>
144
145 00000134 <_exit>:
146 134: f8 94          cli
147
148 00000136 <__stop_program>:
149 136: ff cf          rjmp    .-2          ; 0x136 <__stop_program>
150
```

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.