STONY BROOK UNIVERSITY
DEPARTMENT OF COMPUTER AND ELECTRICAL ENGINEERING

ESE 381.L02

Lab 3: Clock Control Module CLKCTRL and Software Delays

Name: Jason Tan
SBU ID #: 112319102
Due Date: 17 Feb. 2022 by 9PM

Design Task 3: My way of verifying whether changing CLK_OUT using the main clock prescaler also changes CLK_CPU is by setting a delay using an avr-libc delay function that toggles between for 1Hz. Then I can start with CLK_OUT where I use default frequency of 4 MHz with prescaler and then I can record the CLK_CPU frequency from what was done from avr-libc delay function generated in my chosen output I/O pin which has LED0 as the output. Then I will use a different prescaler value to change the clock frequency at CLK_OUT. Then I will observe if there's any change in the frequency generated from the avr-libc delay function compared to before. This will help me determine whether changing CLK_OUT using the main clock prescaler also changes CLK_CPU generated from the delay functions by observing whether the LED0 blinks normally or if it blinks faster or slower than normal. I can even measure based on whether the CLK_CPU changed using an oscilloscope by observing my chosen I/O pin.

Design Task 4: My way of verifying on the toggle bit at one of three different rates: every 52.08333, 104.1666, or 208.333 us is to rely on the oscilloscope at one of the I/O pin that I want to output one of three different rate. In this case, since I configured PC3 as my output pin where I want to check my toggle rate, I will turn on and off either two of the DIP switches to verify whether my corresponding 3 pulse widths to be outputted are correct by observing from the oscilloscope.

Design Task 5: My way of verifying whether the output frequency output is correct is by relying on the oscilloscope where I will be able to check on the waveform output from PA7. Then I will be able to configure whether the displayed approximate frequency on the oscilloscope is mainly correct or not as it should by seeing if it's approximately closed to 32.768 kHz.

```c
/*
 * one_MHz.c
 *
 * Created: 2/7/2022 12:26:01 AM
 * Author : Jason Tan
 */

#include <avr/io.h>

#define DISABLE_PRESCALAR 0x00;

int main(void)
{
    PORTA.DIR = PIN7_bm;   //PA7 as the output
    CPU_CCP = CCP_IOREG_gc;     //change protected IO register
    CLKCTRL.MCLKCTRLB = CLKCTRL_PEN_bm | CLKCTRL_PDIV0_bm;    //Enable CLK_MAIN
      prescalar and divide by 4 for prescalar
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm;      //enable CLKOUT
    while (1)
    {
    }
}
```

```c
/*
 * blink_LED0_1sec.c
 *
 * Created: 2/13/2022 9:53:42 AM
 * Author : jason
 */

#define F_CPU 4000000
#include <avr/io.h>
#include <util/delay.h>



int main(void)
{
    PORTA_DIR = PIN7_bm;    //PA7 as the output
    PORTB_DIR = PIN3_bm;      //Set PB3 as the output
    CPU_CCP = CCP_IOREG_gc;  //change protected IO register
    CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm;  //enable CLKOUT

    while (1)
    {
        PORTB_OUT |= PIN3_bm;
        _delay_ms(500);
        PORTB_OUT &= ~PIN3_bm;
        _delay_ms(500);
    }
}
```

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

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .data         00000000  00804000  00804000  000001b6  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  1 .text         00000162  00000000  00000000  00000054  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .comment      00000030  00000000  00000000  000001b6  2**0
                  CONTENTS, READONLY
  3 .note.gnu.avr.deviceinfo 00000040  00000000  00000000  000001e8  2**2
                  CONTENTS, READONLY
  4 .debug_aranges 00000020  00000000  00000000  00000228  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_info   00003251  00000000  00000000  00000248  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_abbrev 00002e08  00000000  00000000  00003499  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   000003bf  00000000  00000000  000062a1  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_frame  00000024  00000000  00000000  00006660  2**2
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_str    0000178c  00000000  00000000  00006684  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_loc    00000048  00000000  00000000  00007e10  2**0
                  CONTENTS, READONLY, DEBUGGING
 11 .debug_ranges 00000010  00000000  00000000  00007e58  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 af 00     jmp 0x15e   ; 0x15e <_exit>
```

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

0000010c <main>:
#include <util/delay.h>


int main(void)
{
    PORTA_DIR = PIN7_bm;    //Set PA7 as the output pin for the CLK_PER (CLK_OUT)
 10c:   80 e8           ldi r24, 0x80   ; 128
 10e:   80 93 00 04     sts 0x0400, r24 ; 0x800400 <__TEXT_REGION_LENGTH__+0x7e0400>
    PORTB_DIR = PIN3_bm;      //Set PB3 as the output
 112:   98 e0           ldi r25, 0x08   ; 8
 114:   90 93 20 04     sts 0x0420, r25 ; 0x800420 <__TEXT_REGION_LENGTH__+0x7e0420>

    CPU_CCP = CCP_IOREG_gc;  //change protected IO register
 118:   98 ed           ldi r25, 0xD8   ; 216
 11a:   94 bf           out 0x34, r25   ; 52
    //I guess enable the CLK_MAIN prescaler and set prescaler
    //div to whatever value to divide by what value (4) to see
    CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV0_bm | CLKCTRL_PEN_bm;
 11c:   e0 e6           ldi r30, 0x60   ; 96
 11e:   f0 e0           ldi r31, 0x00   ; 0
 120:   23 e0           ldi r18, 0x03   ; 3
 122:   21 83           std Z+1, r18    ; 0x01
    CPU_CCP = CCP_IOREG_gc;  //change protected IO register
 124:   94 bf           out 0x34, r25   ; 52
    CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm;  //enable CLKOUT
 126:   80 83           st  Z, r24

    while (1)
    {
        PORTB_OUT |= PIN3_bm;
 128:   e4 e2           ldi r30, 0x24   ; 36
 12a:   f4 e0           ldi r31, 0x04   ; 4
 12c:   80 81           ld  r24, Z
 12e:   88 60           ori r24, 0x08   ; 8
 130:   80 83           st  Z, r24
    #else
        //round up by default
        __ticks_dc = (uint32_t)(ceil(fabs(__tmp)));
    #endif

    __builtin_avr_delay_cycles(__ticks_dc);
 132:   2f e7           ldi r18, 0x7F   ; 127
 134:   8a e1           ldi r24, 0x1A   ; 26
 136:   96 e0           ldi r25, 0x06   ; 6
 138:   21 50           subi    r18, 0x01   ; 1
 13a:   80 40           sbci    r24, 0x00   ; 0
 13c:   90 40           sbci    r25, 0x00   ; 0
```

```
 13e:   e1 f7           brne    .-8         ; 0x138 <main+0x2c>
 140:   00 c0           rjmp    .+0         ; 0x142 <main+0x36>
 142:   00 00           nop
        _delay_ms(500);
        PORTB_OUT &= ~PIN3_bm;
 144:   80 81           ld  r24, Z
 146:   87 7f           andi    r24, 0xF7   ; 247
 148:   80 83           st  Z, r24
 14a:   2f e7           ldi r18, 0x7F   ; 127
 14c:   8a e1           ldi r24, 0x1A   ; 26
 14e:   96 e0           ldi r25, 0x06   ; 6
 150:   21 50           subi    r18, 0x01   ; 1
 152:   80 40           sbci    r24, 0x00   ; 0
 154:   90 40           sbci    r25, 0x00   ; 0
 156:   e1 f7           brne    .-8         ; 0x150 <main+0x44>
 158:   00 c0           rjmp    .+0         ; 0x15a <main+0x4e>
 15a:   00 00           nop
 15c:   e5 cf           rjmp    .-54        ; 0x128 <main+0x1c>

0000015e <_exit>:
 15e:   f8 94           cli

00000160 <__stop_program>:
 160:   ff cf           rjmp    .-2         ; 0x160 <__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  0000019c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  1 .text         00000148  00000000  00000000  00000054  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .comment      00000030  00000000  00000000  0000019c  2**0
                  CONTENTS, READONLY
  3 .note.gnu.avr.deviceinfo 00000040  00000000  00000000  000001cc  2**2
                  CONTENTS, READONLY
  4 .debug_aranges 00000020  00000000  00000000  0000020c  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_info   00003021  00000000  00000000  0000022c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_abbrev 00002d5f  00000000  00000000  0000324d  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000348  00000000  00000000  00005fac  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_frame  00000024  00000000  00000000  000062f4  2**2
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_str    00001663  00000000  00000000  00006318  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_loc    0000002a  00000000  00000000  0000797b  2**0
                  CONTENTS, READONLY, DEBUGGING
 11 .debug_ranges 00000010  00000000  00000000  000079a5  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 a2 00     jmp 0x144   ; 0x144 <_exit>
```

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

0000010c <main>:

#define PULL_UP_VAL 0x08

int main(void)
{
    PORTC_DIR = PIN3_bm;  //Configure PC3 as the output
 10c:   88 e0           ldi r24, 0x08   ; 8
 10e:   80 93 40 04     sts 0x0440, r24 ; 0x800440 <__TEXT_REGION_LENGTH__+0x7e0440>

    PORTC_PIN0CTRL = PULL_UP_VAL;    //Enable internal pull up resistor for PC0
 112:   80 93 50 04     sts 0x0450, r24 ; 0x800450 <__TEXT_REGION_LENGTH__+0x7e0450>
    PORTC_PIN1CTRL = PULL_UP_VAL;   //Enable internal pull up resistor for PC1
 116:   80 93 51 04     sts 0x0451, r24 ; 0x800451 <__TEXT_REGION_LENGTH__+0x7e0451>

    while (1)
    {
        PORTC_OUT |= PIN3_bm;   //Set PC3, XOR
 11a:   a4 e4           ldi r26, 0x44   ; 68
 11c:   b4 e0           ldi r27, 0x04   ; 4
 11e:   8c 91           ld  r24, X
 120:   88 60           ori r24, 0x08   ; 8
 122:   8c 93           st  X, r24
        //For two bits to be used for the DIP SWITCH
        //PC0 = 0 and PC1 = 0 for DIP SWITCH


        if((~(PIN0_bm & PORTC_IN)) && (~(PIN1_bm & PORTC_IN))){
 124:   e8 e4           ldi r30, 0x48   ; 72
 126:   f4 e0           ldi r31, 0x04   ; 4
 128:   80 81           ld  r24, Z
 12a:   80 81           ld  r24, Z
    can be achieved.
*/
void
_delay_loop_1(uint8_t __count)
{
    __asm__ volatile (
 12c:   81 e2           ldi r24, 0x21   ; 33
 12e:   98 2f           mov r25, r24
 130:   9a 95           dec r25
 132:   f1 f7           brne    .-4         ; 0x130 <main+0x24>
        else if(((PIN0_bm & PORTC_IN)) && (~(PIN1_bm & PORTC_IN))){
            _delay_loop_1(137);
        }


        PORTC_OUT &= ~PIN3_bm;   //Clear PC3
```

```
 134:   9c 91           ld  r25, X
 136:   97 7f           andi    r25, 0xF7   ; 247
 138:   9c 93           st  X, r25

        //For two bits to be used for the DIP SWITCH
        //PC0 = 0 and PC1 = 0 for DIP SWITCH
        if((~(PIN0_bm & PORTC_IN)) && (~(PIN1_bm & PORTC_IN))){
 13a:   90 81           ld  r25, Z
 13c:   90 81           ld  r25, Z
 13e:   8a 95           dec r24
 140:   f1 f7           brne    .-4         ; 0x13e <main+0x32>
 142:   eb cf           rjmp    .-42        ; 0x11a <main+0xe>

00000144 <_exit>:
 144:   f8 94           cli

00000146 <__stop_program>:
 146:   ff cf           rjmp    .-2         ; 0x146 <__stop_program>
```

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

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .data         00000000  00804000  00804000  0000017c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  1 .text         00000128  00000000  00000000  00000054  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .comment      00000030  00000000  00000000  0000017c  2**0
                  CONTENTS, READONLY
  3 .note.gnu.avr.deviceinfo 00000040  00000000  00000000  000001ac  2**2
                  CONTENTS, READONLY
  4 .debug_aranges 00000020  00000000  00000000  000001ec  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_info   00003195  00000000  00000000  0000020c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_abbrev 00002d76  00000000  00000000  000033a1  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000322  00000000  00000000  00006117  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_frame  00000024  00000000  00000000  0000643c  2**2
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_str    0000173d  00000000  00000000  00006460  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_ranges 00000010  00000000  00000000  00007b9d  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 92 00     jmp 0x124   ; 0x124 <_exit>

00000108 <__bad_interrupt>:
```

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

0000010c <main>:
#define DISABLE_PRESCALAR 0x00


int main(void)
{
    PORTA_DIR = PIN7_bm;        //Configure PA7 as the output
 10c:   80 e8           ldi r24, 0x80   ; 128
 10e:   80 93 00 04     sts 0x0400, r24 ; 0x800400 <__TEXT_REGION_LENGTH__+0x7e0400>
    CPU_CCP = CCP_IOREG_gc;     //change protected IO register
 112:   88 ed           ldi r24, 0xD8   ; 216
 114:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKCTRLB = DISABLE_PRESCALAR;   //CLK_MAIN prescalar disabled
 116:   e0 e6           ldi r30, 0x60   ; 96
 118:   f0 e0           ldi r31, 0x00   ; 0
 11a:   11 82           std Z+1, r1 ; 0x01
    CPU_CCP = CCP_IOREG_gc;  //change protected IO register
 11c:   84 bf           out 0x34, r24   ; 52
    CLKCTRL.MCLKCTRLA = CLKCTRL_CLKOUT_bm | CLKCTRL_CLKSEL0_bm;  //Enable CLKOUT and
      clock select bit 0 mask to 32.768 kHz internal oscillator
 11e:   81 e8           ldi r24, 0x81   ; 129
 120:   80 83           st  Z, r24
 122:   ff cf           rjmp    .-2         ; 0x122 <main+0x16>

00000124 <_exit>:
 124:   f8 94           cli

00000126 <__stop_program>:
 126:   ff cf           rjmp    .-2         ; 0x126 <__stop_program>
```