CS4403 Laboratory 2

Dr J. Vaughan

February 22, 2013

1 A possible step: Blink 2

In the implementation of the alarm clock, you will need to use timers and interrupts.

The following code is taken from the Arduino Forum.

Connect an LED in series with a 10kΩ resistor between Pin 13 of the Arduino Due board and ground.

Connect the programmer to the Due board and then connect the USB cable to the computer. Upload the timer code to the Arduino board. The LED should blink.

Visit the Arduino website and forum to investigate how the code works.

In your report on today's practical, answer the following questions:

- How many interrupt levels are available in the Arduino Due?
- How many independent timers are available in the Arduino Due?
- How can the timers be connected to the interrupt levels?

Also, in this report give details of your progress with the Alarm Clock system. Give a schedule to completion and identify the critical path.

```
//timer1.ino code from the Arduino Forum

volatile boolean l;

//TC1 ch 0
void TC3_Handler()
{
        TC_GetStatus(TC1, 0);
        digitalWrite(13, l = !l);
}

void startTimer(Tc *tc, uint32_t channel,
                            IRQn_Type irq, uint32_t frequency) {
        pmc_set_writeprotect(false);
        pmc_enable_periph_clk((uint32_t)irq);
        TC_Configure(tc, channel, TC_CMR_WAVE | TC_CMR_WAVSEL_UP_RC |
        TC_CMR_TCCLKS_TIMER_CLOCK4);
        uint32_t rc = VARIANT_MCK/128/frequency;
        //128 because we selected TIMER_CLOCK4 above
        TC_SetRA(tc, channel, rc/2); //50% high, 50% low
        TC_SetRC(tc, channel, rc);
        TC_Start(tc, channel);
        tc->TC_CHANNEL[channel].TC_IER=TC_IER_CPCS;
        tc->TC_CHANNEL[channel].TC_IDR=~TC_IER_CPCS;
        NVIC_EnableIRQ(irq);
}

void setup(){
        pinMode(13,OUTPUT);
        startTimer(TC1, 0, TC3_IRQn, 4);
        //TC1 channel 0, the IRQ for that channel
        //and the desired frequency
}

void loop(){
}
```