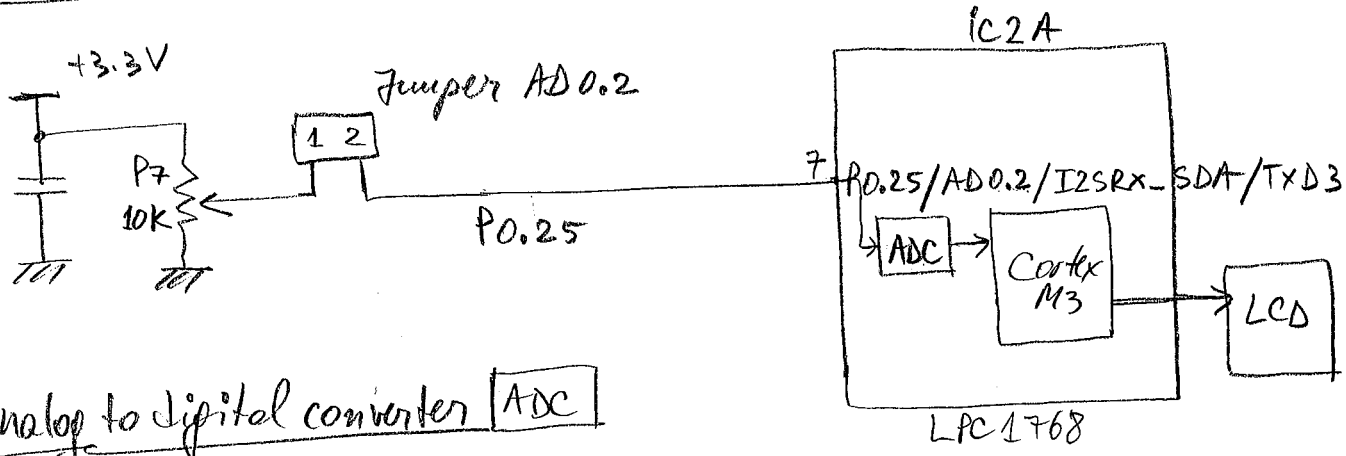
ADC | Analog to Digital Converter

lecture #9
part 3

Potentiometer on Board MCB1700



Analog to digital converter | ADC

- see chapter 29 of User Manual (pp. 574)
- 12-bit successive approximation ADC (conversion rate 200 kHz)
- 0x0000 ÷ 0x0FFF values range.
  (0x0FFF = ADC_VALUE_MAX)

Note: revisit your electronics course about how this ADC works!

Basic configuration of ADC (see page 574 of user manual)

1. Power Up: PCONP register (table 46); select operation mode ADCR
2. Clock: PCLKSEL0 register (table 40)
3. Pins: Enable ADC0 pins through PINSEL registers. Select pin modes through PINMODE registers
4. Interrupts: Enable ADC as source of interrupts in NVIC
5. DMA

Note | Discussion Based on | Blinky 2 | example from lab #1.

```
void ADC_init(void){

    LPC_SC->PCONP |= (1<<12)|(1<<15); //enable power to ADC & iocon
                                      //(1)
    // page 63 of user manual.

                                      // bits 19:18
    LPC_PINCON->PINSEL1 &= ~(3<<18);  // 111...10011...11 = Mask
              ->PINSEL1 |= (1<<18);   // set bit 18 to '1', i.e.,
                                      // make pin P0.25 to
                                      // function as AD0.2
    //(3)                             // page 108 of user manual.
    LPC_PINCON->PINMODE1 &= ~(3<<18); // page 112
              ->PINMODE1 |= (2<<18);  // make "10" bits 19:18 =
                                      // pin P0.25 has no
                                      // pull up/down resistor!

    // Also part of (1):
    LPC_ADC->ADCR = (1<<2)|(4<<8)|(1<<21); //AD Control Register
                                           //select operation mode
    //              15 14 13 12 11 10 9 8
    // 000...010 00...0 1 0 0 0000 0100
    //          bit 21      bit 10     bit 2
    // bit 21: AD converter is operational
    // bit 2: select which of AD0.7:0 pins is selected & converted
    // Bits 15:8 = (00000100)Bin = 4 Dec
    // Clock is divided by this value.
    // See pages 576, 577 of user manual.

    LPC_ADC->ADINTEN = (1<<8); // AD interrupt Enable register
                               //page 578
    NVIC_EnableIRQ(ADC_IRQn); //(4)
    // ADC end of conversion
                               //only individual
                               //channels will
                               //generate
                               //interrupts!
}
```

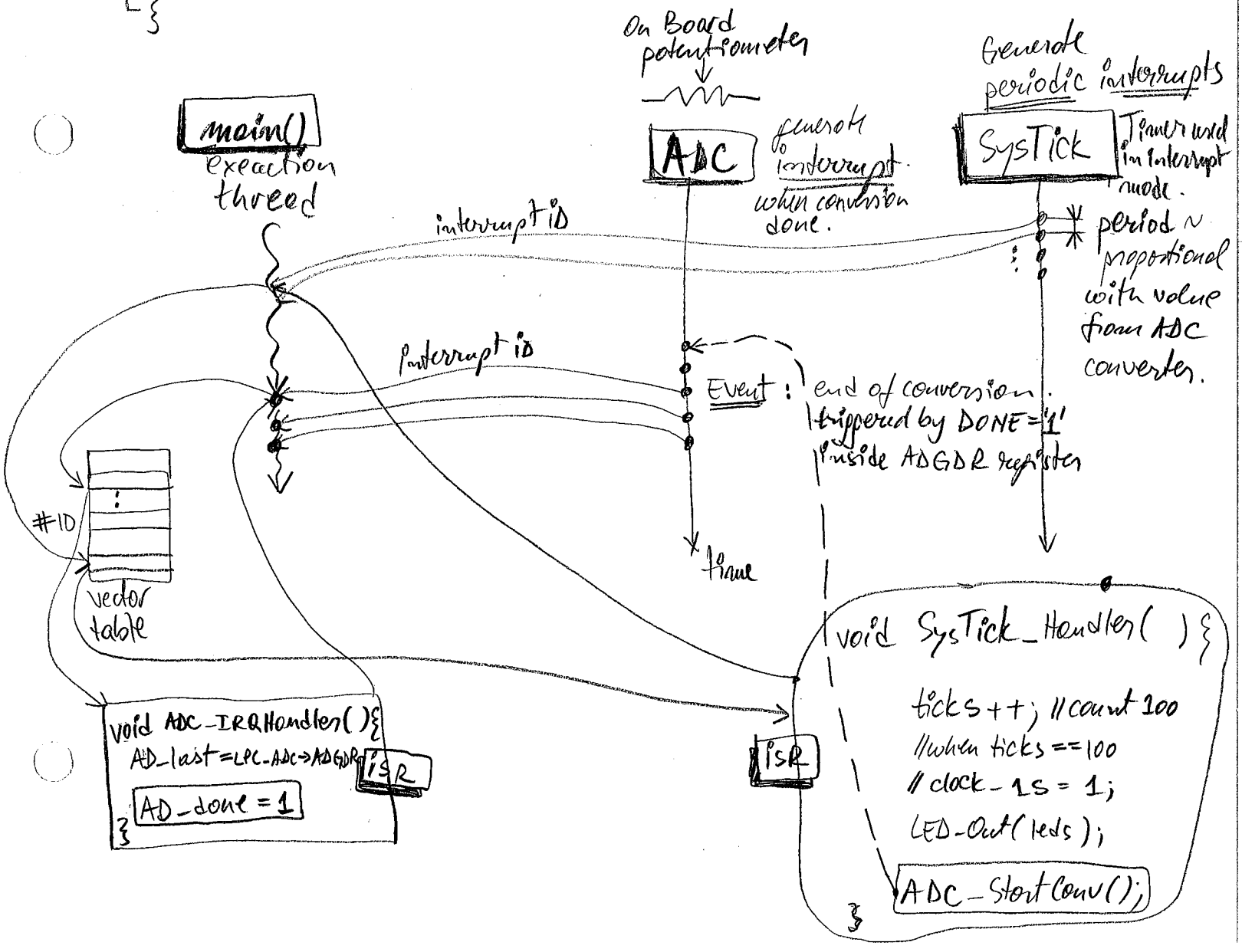
```
int main(void) { // AD_done variable is [key] here!

    ADC_Init();
    SysTick_Config( SystemCoreClock / 100 ); // generate interrupt each 10 ms.
    while (1) { // do forever

        if (AD_done) {          <-- set to 1 by [ISR]
            AD_done = 0; // reset it.        AD_done is a "communication"
            // average over 16 values: [ad_val]   variable.
        }

        // print value
        // print Bargraph.

    }
}
```



```
void ADC_IRQHandler(void) { // this is ISR
    adstat = LPC_ADC->ADSTAT; //read ADC clears interrupt.
    AD_last = (LPC_ADC->ADGDR>>4) & ADC_MAX_VALUE;

    AD_done = 1;
}
```

divide by $2^4 = 16$.

0x0FFF

see page 576
This register contains ADC's DONE bit and the most recent A/D conversion

Questions:

→ How often are the interrupts trigged from ADC?

→ Where inside main() could possibly these interrupts occur and be handled/serviced?

→ where exactly is defined _USE_LCD, _ADC_IRQ?

→ When is the moment when LED's are turned on/off in the (Blinky 2) example?

- See IRQ.c file. ← has the (SysTick_Handler()) whose pointer is stored in vector table
- See startup_LPC17xx.s file ←

inside it LED's are flashed!