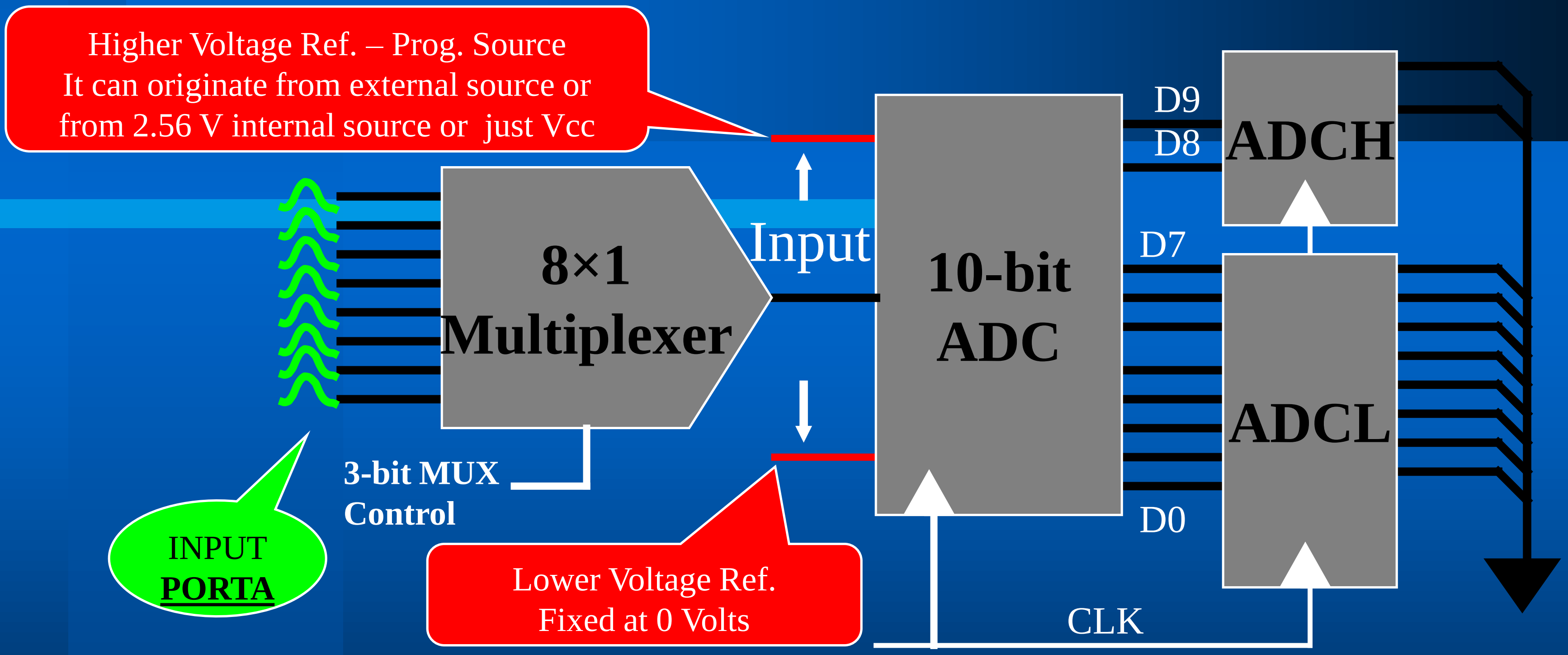
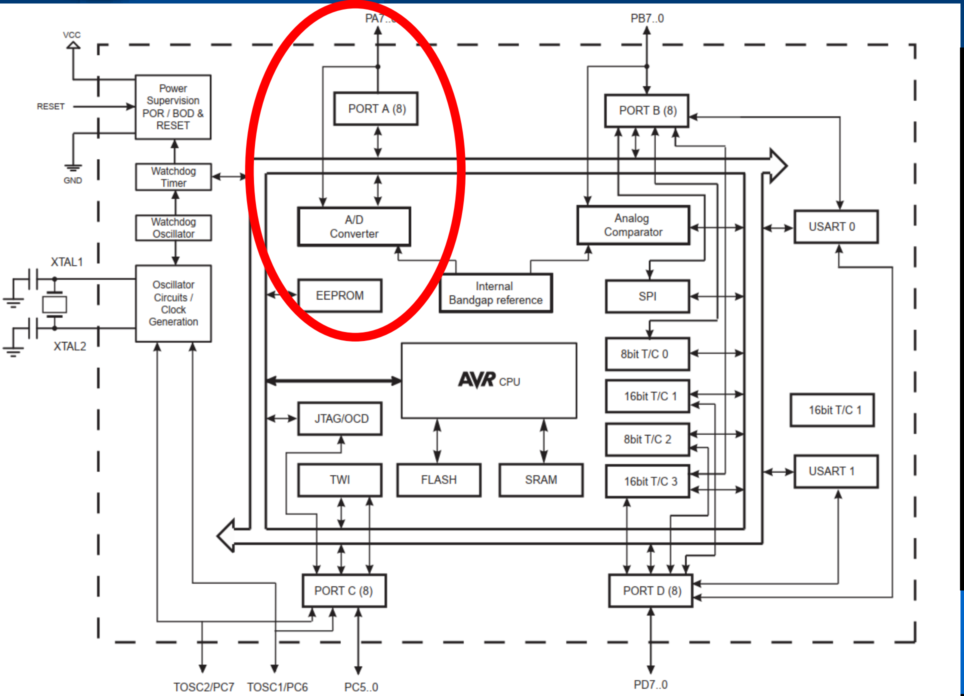
Device Drivers – Digital Voltmeter

The ATmega1284P ADC

The Atmega1284P has one 10-Bit successive approximation ADC (on-chip) connected to an 8 × 1 analog multiplexer:



The ATmega1284P ADC



- **Ideal for applications of voltage or sensor monitoring.**
- **The trigger to digitize/convert can be programmed to come either from program control, or from a voltage comparator or from an interrupt, or from a timer.**
- **In this course this feature will be used to make a digital voltmeter.**

- The 8 Pins of Port A have dual use. As we have already seen they can be used for data I/O.
- They are also connected to the ATmega1284P 8×1 analog multiplexer whose output is driven to a 10-bit ADC which is on-chip. So they can be used to digitize analog data from 8 different voltage sources.

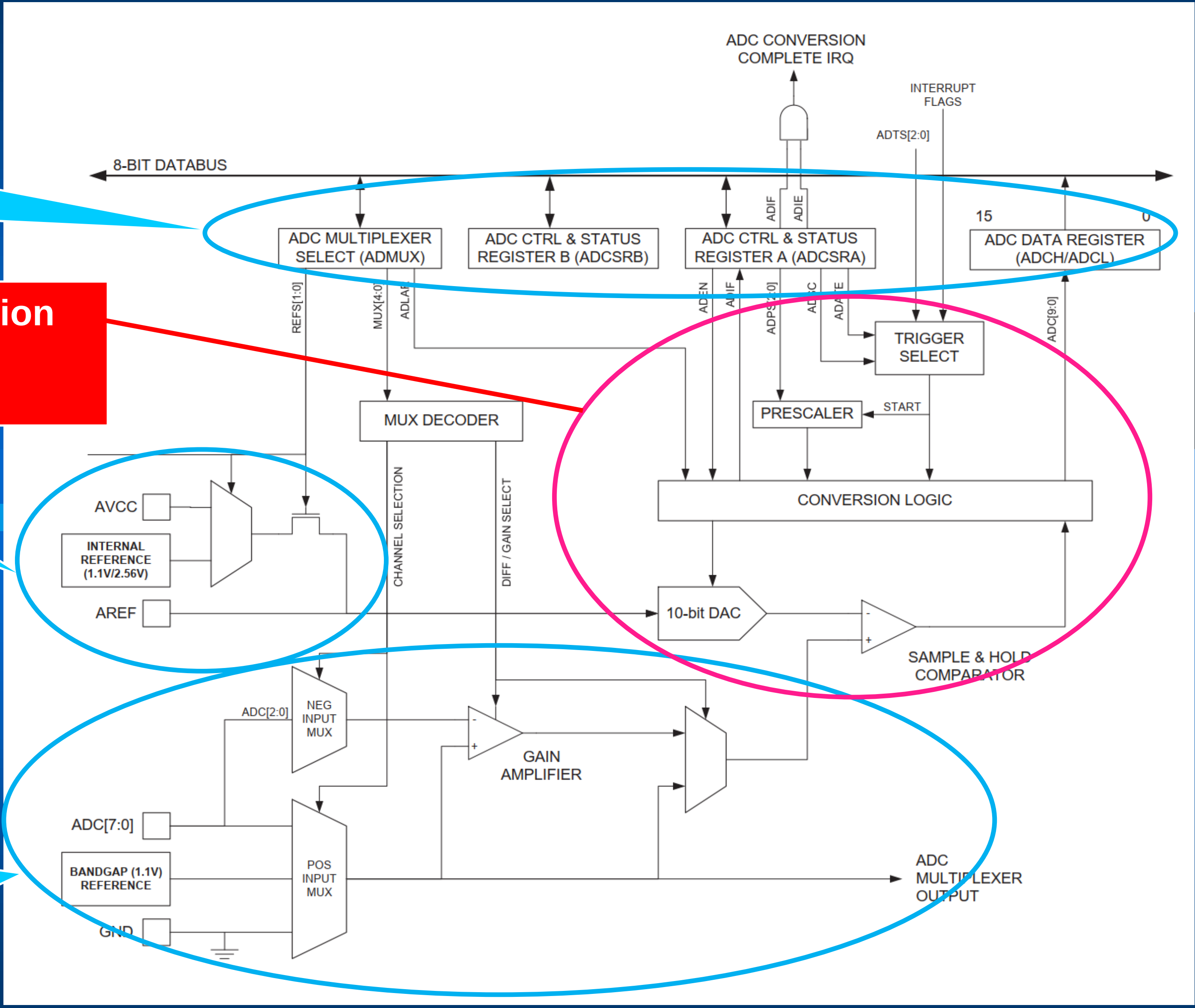
Successive approximation ADC

ADC Registers for configuration and output data

Successive Aproximation 10-bit ADC

Voltage Reference selection Logic

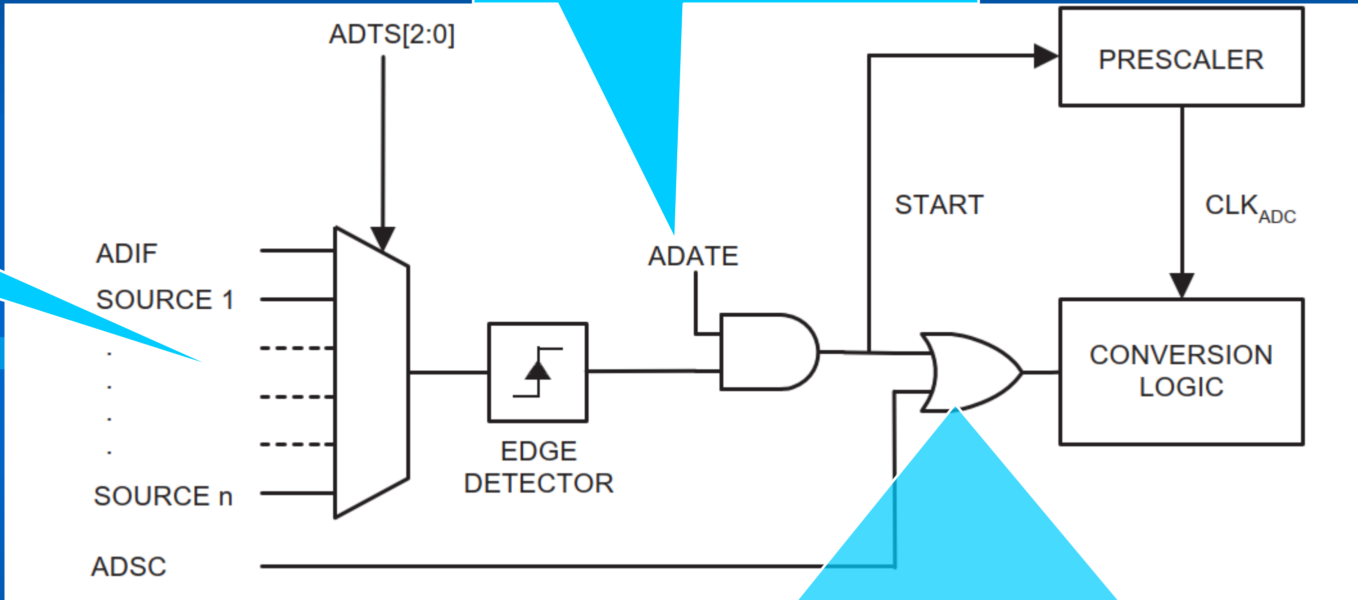
Analogue Multiplexor Logic



ADC Trigger Logic

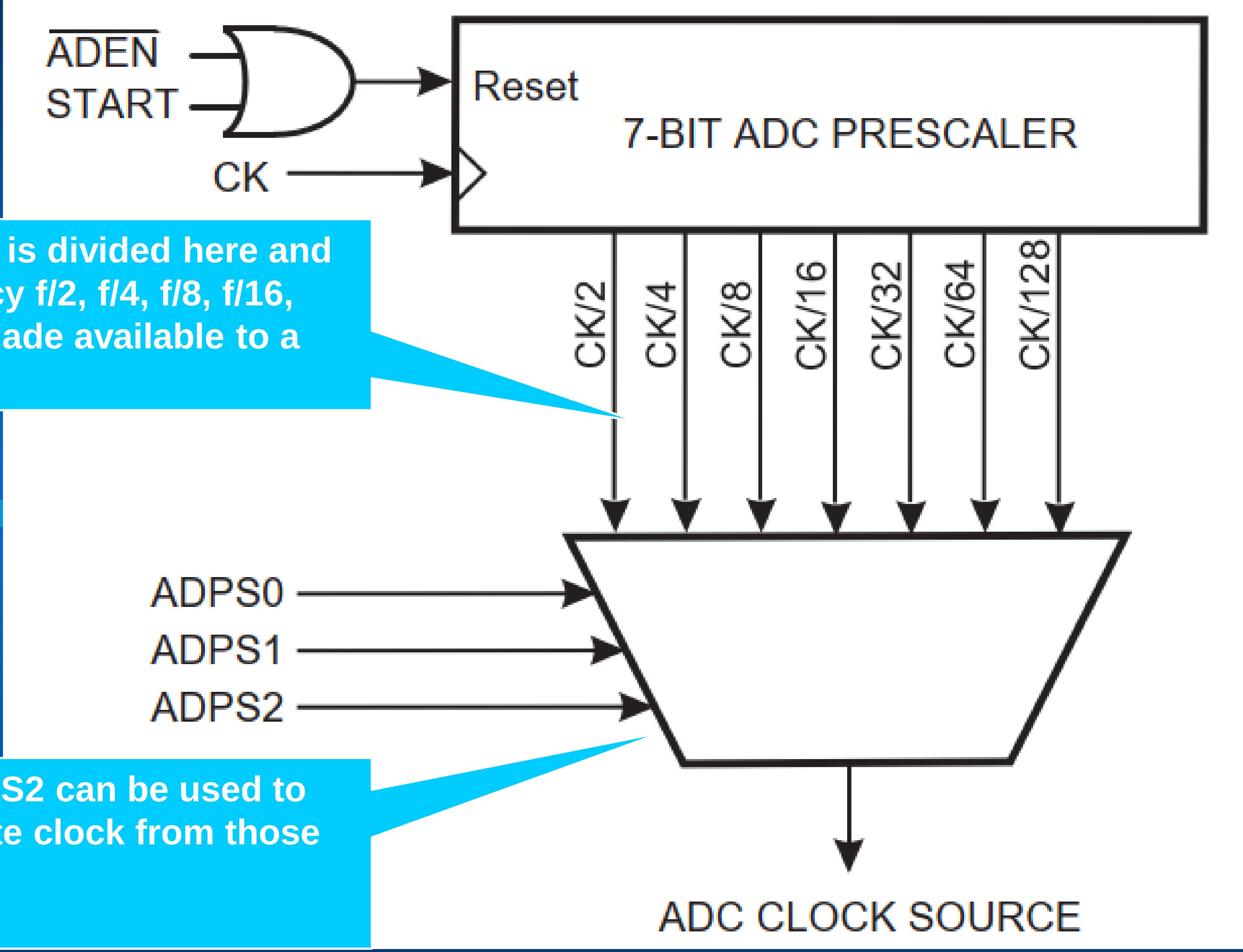
ADATE enables the auto-triggering mode

In Auto Trigger mode One can select via ADTS different trigger sources……



ADSC overwrites auto triggering and causes a single convert

Clock pre-scaling



The clock frequency is divided here and clocks with frequency f/2, f/4, f/8, f/16, f/32, f/64, f/128 are made available to a multiplexer.

ADPS0, ADPS1, ADPS2 can be used to select the appropriate clock from those made available.

The ADC Registers I

- The 10-bit ADC on Atmega1284P receives its inputs from PORTA.

- The reference voltage can be chosen under program control to be either external, or internal 2.65 Volt, or the Analog Vcc.

- The device can be controlled by 4 on board registers:
  - ADCCSRA: ADC Status Register A
  - ADCCSRB: ADC Status Register B (Enabled by ADATE)
  - ADMUX : ADC Multiplexer Selection Register
  - ADCH : Data Register for bits D9, D8 (right aligned)
  - ADCL : Data Register for bits D0 - D7 (right aligned)
  - DIDR0 : Digital Input Disable Register 0

The ADC Registers II

The ADC Status Register ADCSRA :
(Page 256 in ATmega1284P data sheet)

D7	D6	D5	D4	D3	D2	D1	D0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

ADEN : It turns on the ADC clock (You must do it first)

ADSC : Start conversion

ADATE : ADC Auto trigger enable (set it zero here since we use single converts)

ADIF : Interrupt flag *(gets set during a conversion and gets cleared if you write '1' to it AFTER you have read BOTH data registers)*

ADIE : Interrupt enable *(in the software I have not used interrupts so I have kept it '0')*

ADPS0-2 : Three bits that determine the ADC clock prescale i.e. the number which one can divide the CPU clock to produce the ADC clock (e.q. 3 = 1/8)

The ADC Registers III

The ADC Multiplexer Register ADMUX is used to select:

(1) The reference voltage V_{REF}

(2) How to present the digital results (left/right aligned)

(3) Which channel to digitize and the type of input signal (single ended or differential) and also allows to select 1.1 or 0 Volts as input

D7	D6	D5	D4	D3	D2	D1	D0
REFS1	REFS0	ADLAR	MUX3	MUX3	MUX2	MUX1	MUX0

- **Bits D7 and D6 are used to select $V_{REF\ K}$**
  - **00 : Internal V_{REF} turned off**
  - **01 : V_{REF} AVCC with external capacitor at AREF**
  - **10 : V_{REF} Internal 1.1 Volts with external capacitor at AREF**
  - **11 : V_{REF} 2.56 Volts with external capacitor at AREF**
- **Bit D5 controls if the result is left adjusted (D5=1) or right adjusted (D5=0)**
- **Bits D0-D4 Control the input configuration (see page 256 of ATmega1284 Data sheet)**

The ADC Registers IV

The ADC Data Register ADCL (in right adjust mode) is used for the 8 lower data bits (D7 - D0) :

D7	D6	D5	D4	D3	D2	D1	D0
D7	D6	D5	D4	D3	D2	D1	D0

The ADC Data Register ADCH (in right adjust mode) is used for the two highest data bits (D9,D8) :

D7	D6	D5	D4	D3	D2	D1	D0
						D9	D8

By setting bit D5 of the ADMUX register one can setup the two registers in left adjust mode. In that case the 8 highest bis will be in ADCH and the 2 lowest buts in ADCL. If for the given application the 8 highest bits are sufficient.

The ADC Registers V

Digital Input Disable Register 0 – DIDR0 (ATmega1284 Data sheet page 259)

D7	D6	D5	D4	D3	D2	D1	D0
ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D

- **Whenever an input if port A is used for digitizing analog signals it is advisable to turn of the data interface connected to that pin to avoind unnecessary power consumption.**
- **Setting any of the above bits to 1 results in turning off the digital interface connected to that pin.**

Task Plan

Design and construct a Digital Voltmeter:

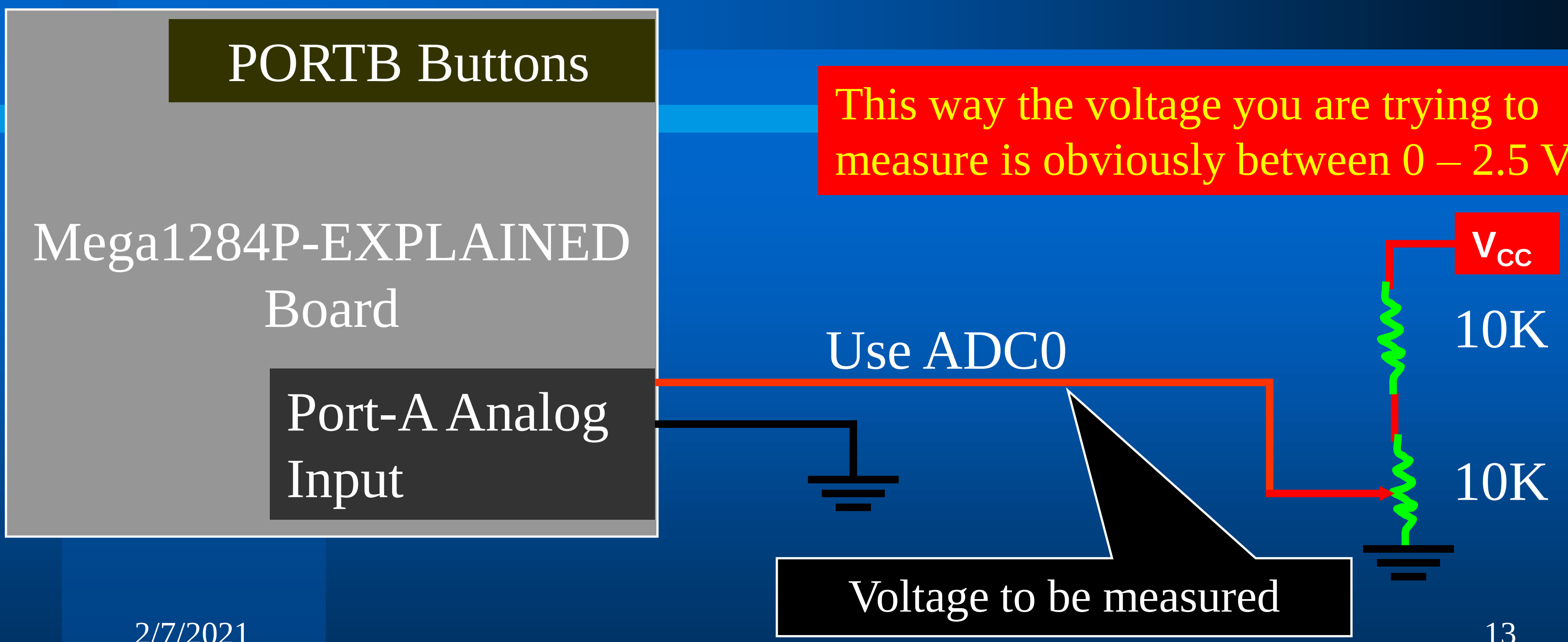
- The Atmega1284P on-chip ADCs should be used to digitize the input analog voltages.
- Use one of the ADC inputs on PORTA to inject the voltage to be measured, which should be less than 2.5 Volts.
- One could use one of the PORTB switches to instruct the Micro Controller to measure voltage.

Connect your voltmeter to the potentiometer and measure the voltage. Calibrate it against a Voltmeter in the Lab. Demonstrate that your device works !!!!

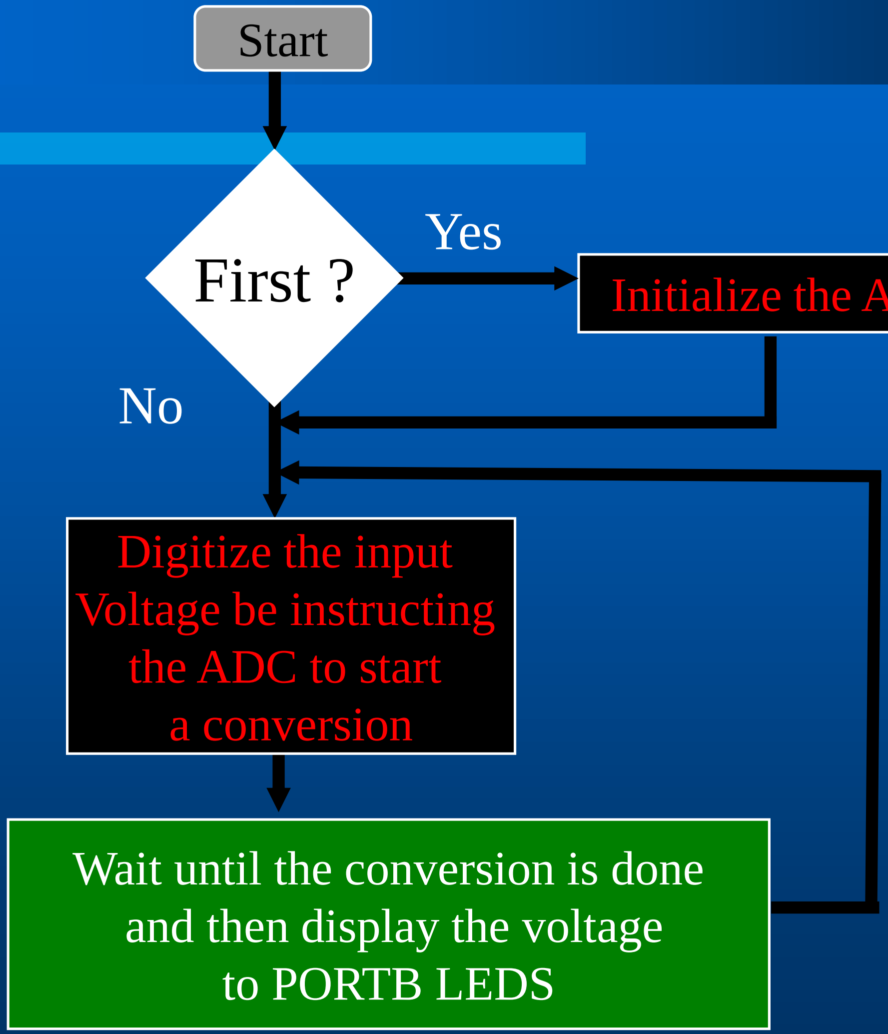
Conceptual Design

The reference voltage of the ADC should not be lower than the voltage you are trying to measure!

This way the voltage you are trying to measure is obviously between 0 – 2.5 Volts



Software Design of the Voltmeter



Costas Foudas, Imperial College, Rm: 508, x47590

ADC Driver Routines I

```
InitADC:
; Turn off all Digital traffic on Port A where the ADC inputs are. DIDR0 - Digital Input Disable Register 0
; Writing '1' on each of the 8 bits of DIDR0 you disable the corresponding Digital input because the ADC
; input are there. So we disable all 8 digital input pins.
;
LDI R16, $FF
STS DIDR0, R16    ; DIDR0 has SRAM ADDRESS 7E
;
; Next setup the VREF, RIGHT/LEFT ADJUST, ADC INPUT SETUP. The ADMUX Register handles this
; D7=1, D6=1 -> VREF = 2.56 VOLTS
; D5=0 -> RIGHT Aligned, D3=D2=D1=D0=0 --> ADC0 Single Ended
;
; LDI R16, $C0 ; Setup to read ADC0 single ended
LDI R16, $DE  ; Setup to inject 1.1 Volt for testing with no external voltage
STS ADMUX, R16    ; ADMUX has SRAM ADDRESS  7C
;
; Next enable the ADC, Disable Auto-Triggering, Disable ADC interrupts and set the prescale to 128 (no fast clock required
; if you want to measure DC voltages). Enable ADC -> D7=1, Don't start conversion yet -> D6=0, Disable Auto Triggering ->
; D5=0. Clear  Interrupt Flag by writting D4=1, Disable ADC interrupts so D3 = 0. Set the clock prescale 128 so ;
D0=D1=D2=1
;
LDI R16, $97
STS ADCSRA, R16    ; ADCSRA has SRAM ADDRESS 7A
RET
```

ADC Driver routines II

```
ConvertADC:
;
;  $D7 is the same as $97 in the InitADC
;  except that now we turn on D6 to cause a conversion
;
LDI R16, $C7
STS ADCSRA, R16    ; ADCSRA has SRAM ADDRESS 7A
;
; read the interrupt flag and make sure it is 1
Polling:
LDS R16, ADCSRA
LDI R17, $10
AND R16, R17
CP R16, R17
BREQ Polling
; write one to the interrupt flag to clear it
LDI R16, $97
STS ADCSRA, R16
;
RET
```

```
ReadADC:
;
LDS R16, ADCL
LDS R17, ADCH
OUT PORTB, R17
; given that we inject 1.1 volts with 2.56 Volts VREF
; we expect that R17 will have an '1' and it does
;
RET
```