Embedded Systems Programming (mostly in C)

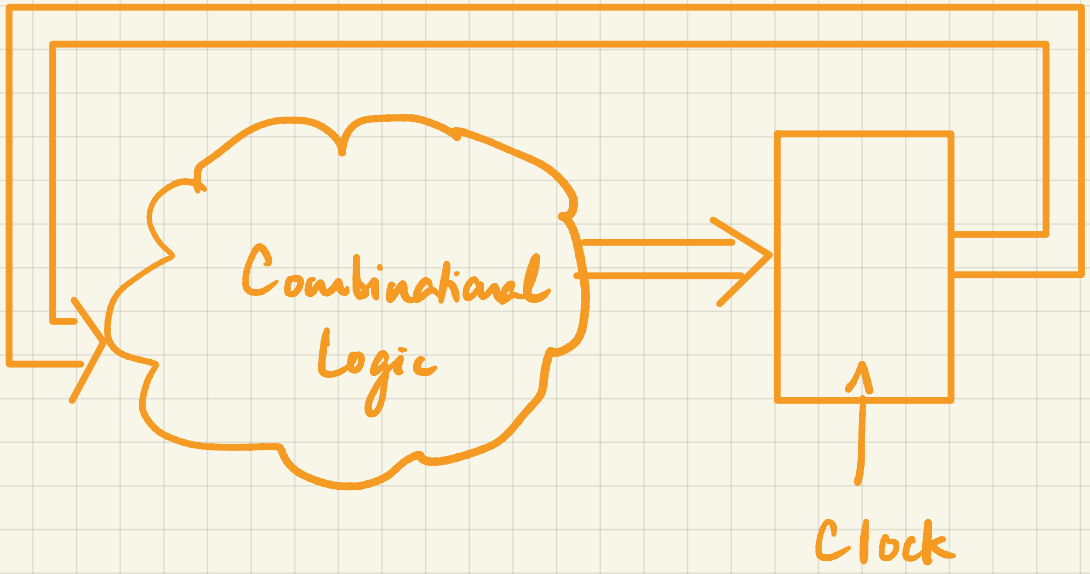
Unit 4: Clocks and Timing

Outline

- Clocking in Micro Controllers
- Sampling and ADCs
- Pulse Width Modulation
- Timers and Input Capture

Clocking

Digital Circuits



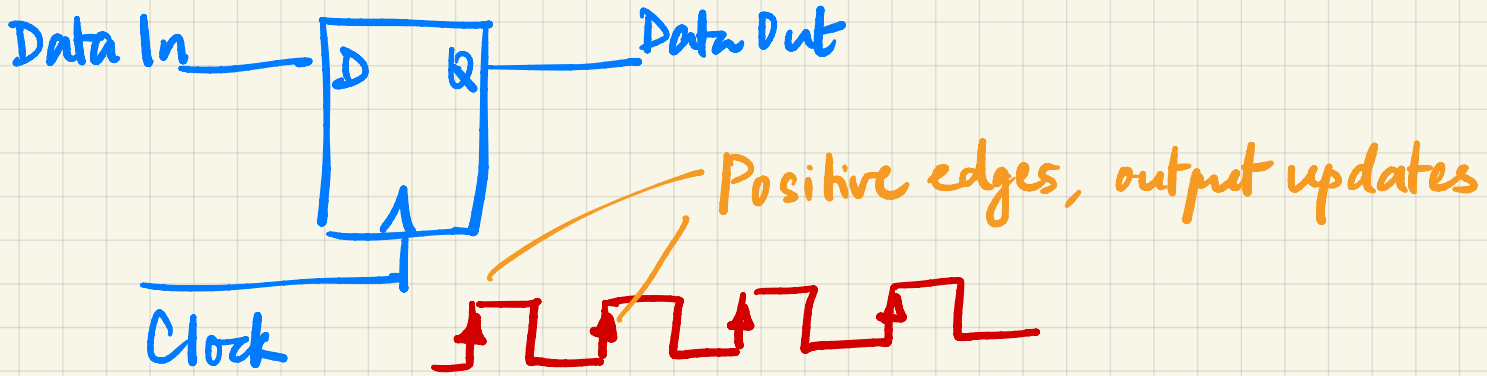
- Combinational elements – Computation / Logic
- Sequential elements – State / Storage

System State

- Retain previous history (memory)
- Compute with previous state + present inputs

Flip-Flop, Register, Latch ...

– Most common: edge triggered flip-flop



CPU

- Complex State Machine
  - Registers / Control hold present state
  - Instructions + Inputs decide computation

⇒ Clock needed to update registers

- Speed of clock ⇒ # cycles / second
  - Determined by electrical characteristics

Clock Speed

Desirable:

- High speed - more compute/second

Clock Speed

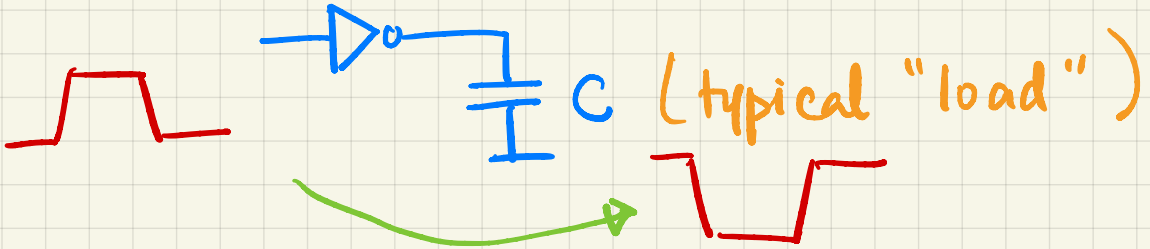
Desirable:

- High speed - more compute/second

Problems

- More switching/activity ⇒ Power consumed
- Slow protocols (UART)
- Peripherals with differing capabilities

Speed and Power



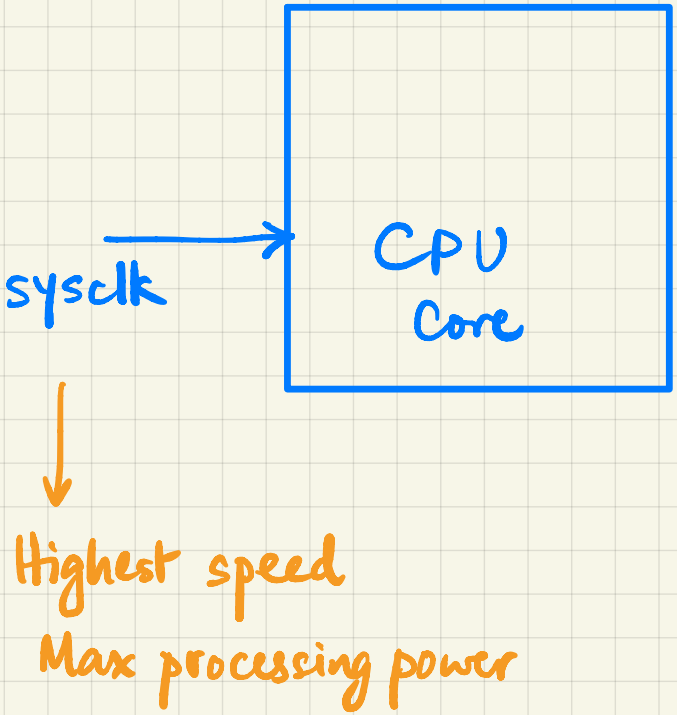
- Switching activity of gates ⇒
  - Energy pulled from supply to load
  - Energy dissipated from load to ground
- Typically: $E \propto C V^2 f$

  - f → Operating frequency
  - V → Supply voltage
  - C → Number of gates / Size of circuit

MCUs and Clocks

- Multiple clocks
  - Flexibility: higher performance vs lower power
- Partial shutdown capability
- Real-Time Clock (RTC)

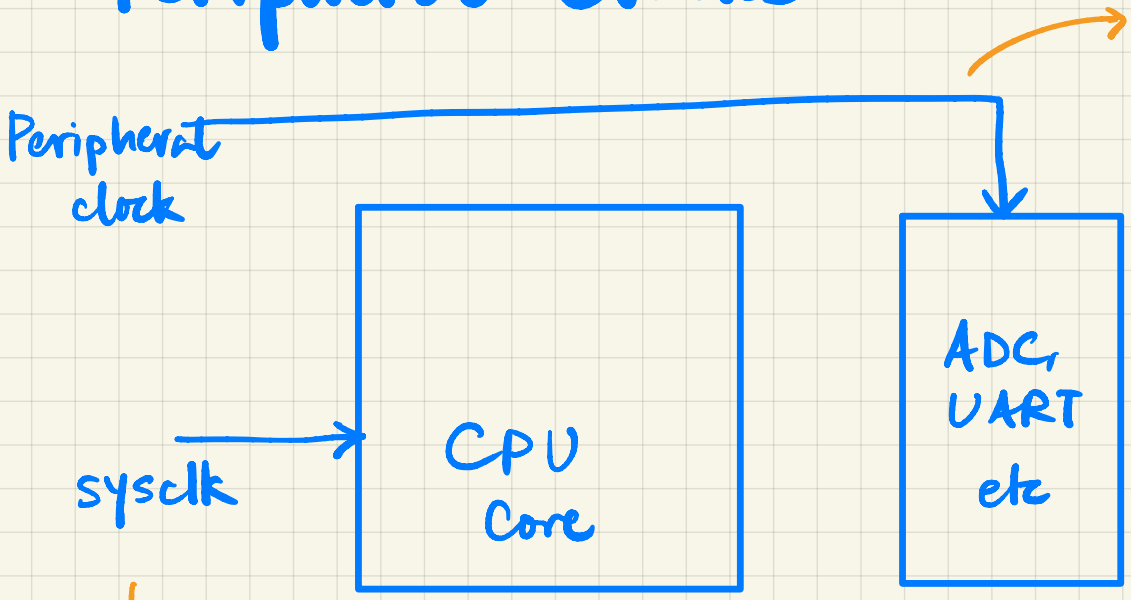
System Clock



Highest speed

- Max processing power

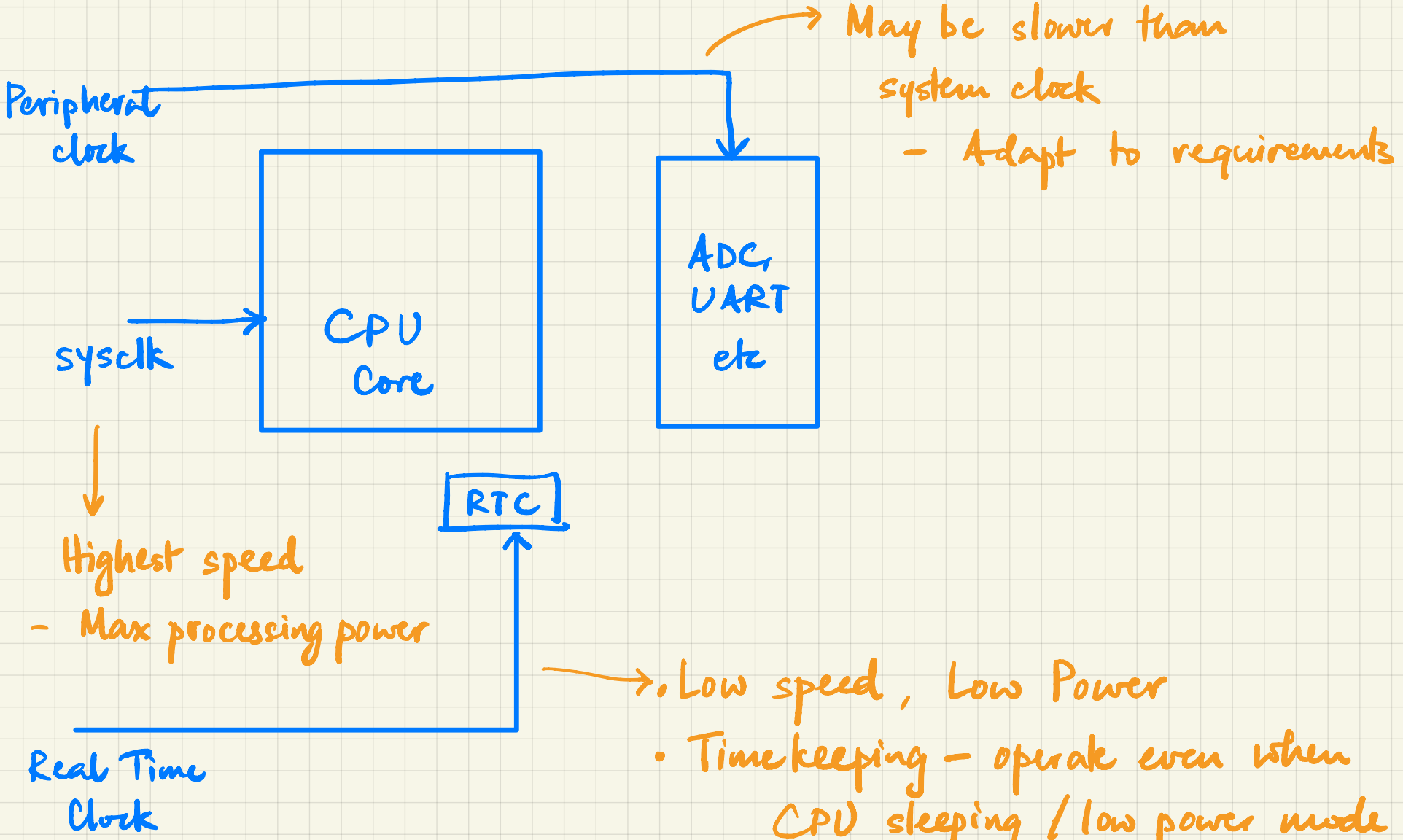
Peripheral Clocks



May be slower than system clock
- Adapt to requirements

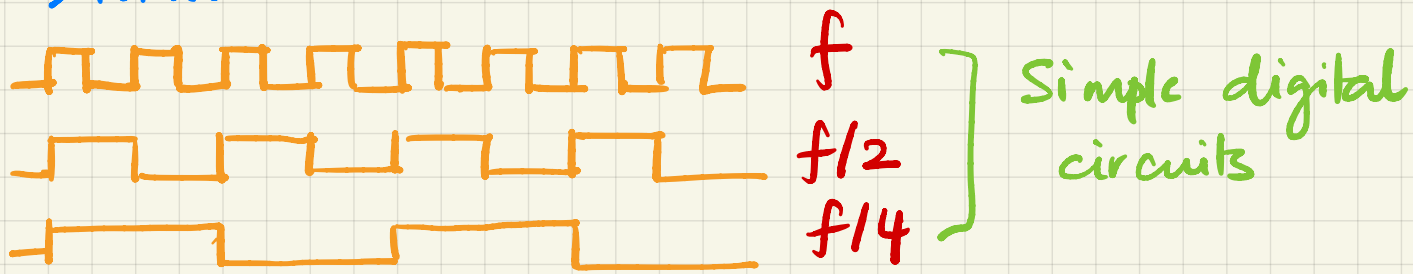
Highest speed
- Max processing power

Real Time Clock

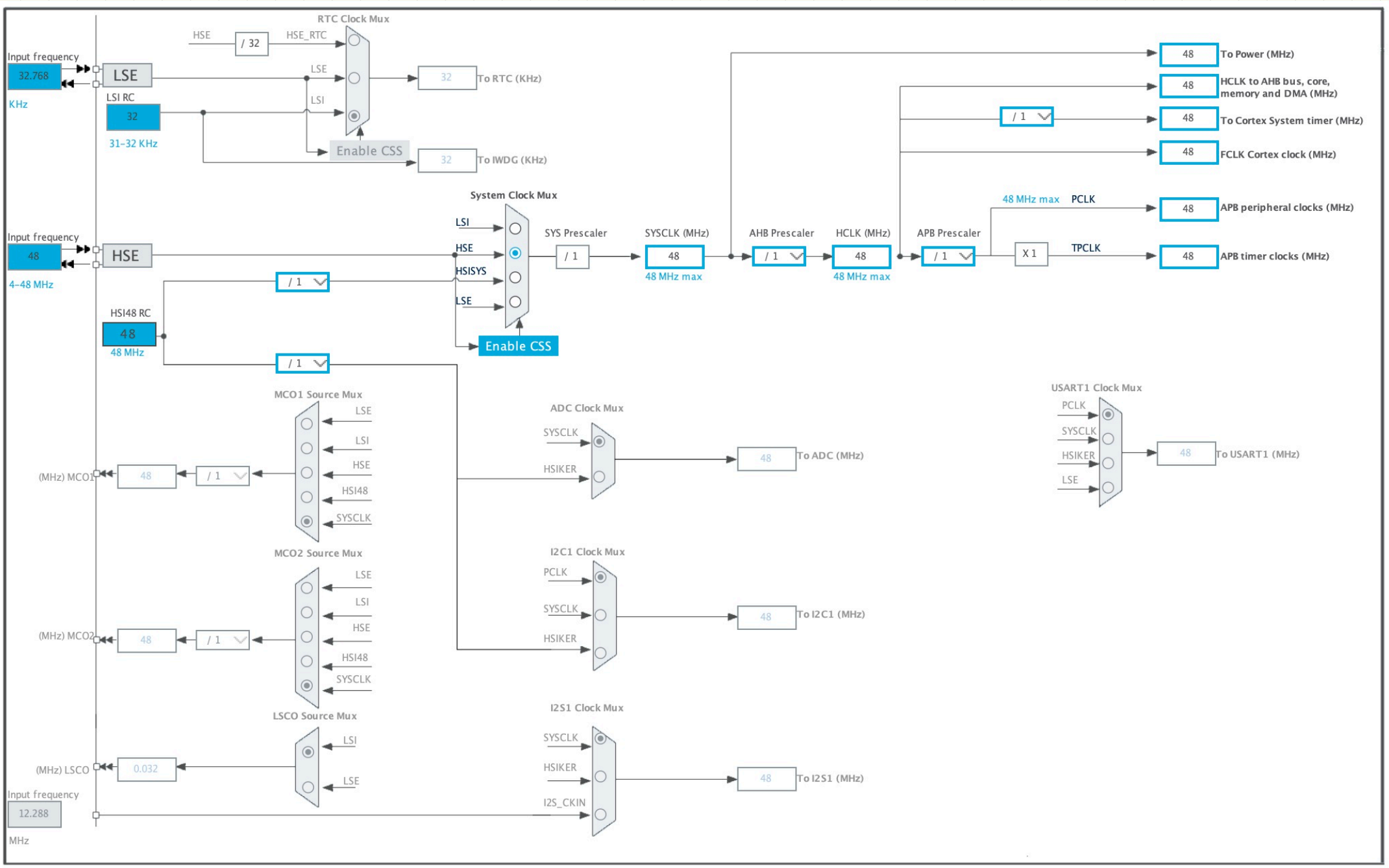


Clock Sources

- Internal RC oscillators
  - MCU can run without external source
  - Not very accurate
- External Crystal oscillators
  - High accuracy, Low drift / temperature sensitivity
- Phase Locked Loops
  - Frequency Conversion – up or down (eg. 48MHz from 4MHz input)
- Clock Division



STM 32 clocks



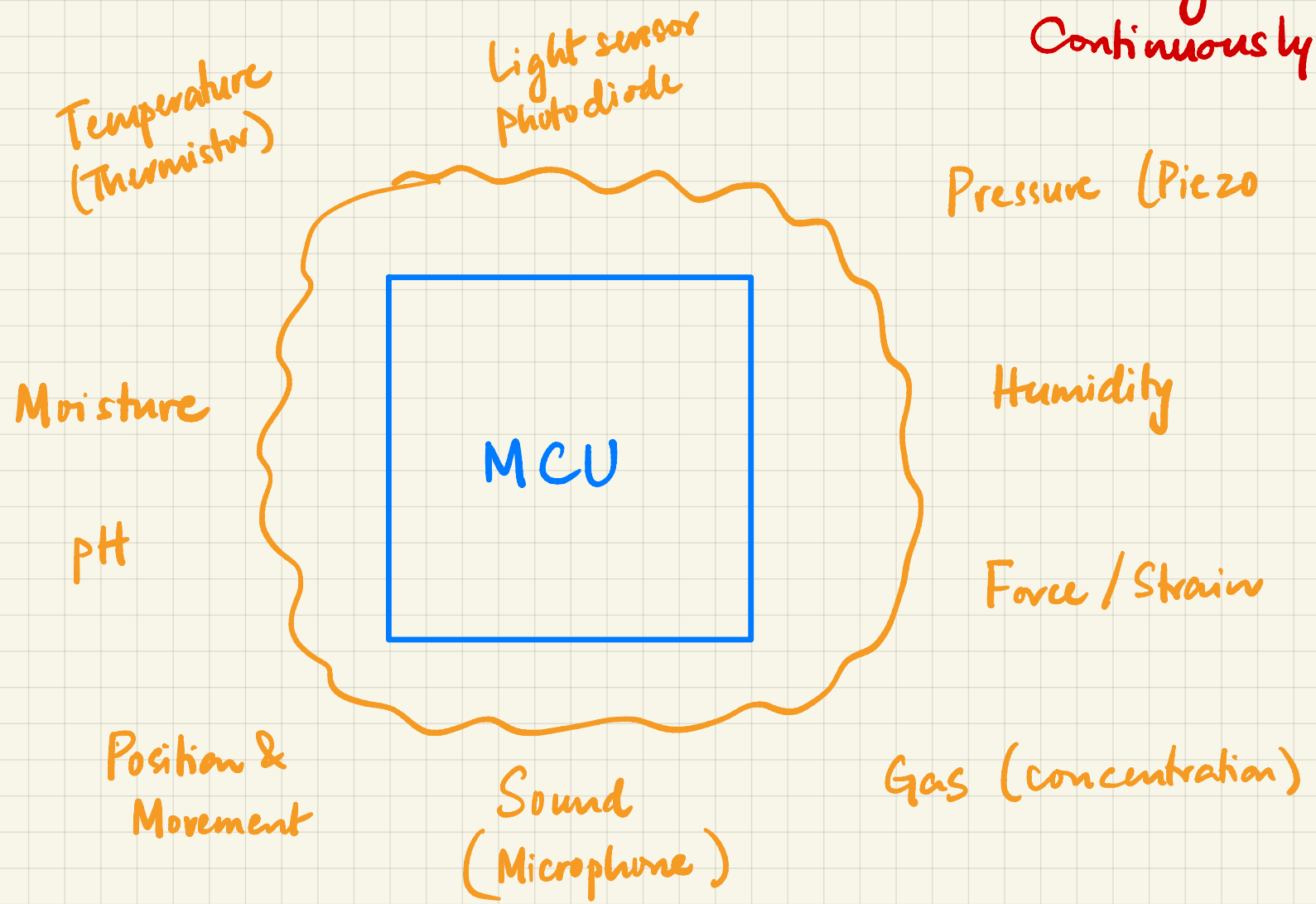
Implications

- Performance
  - High speed ⇒ More computation
  - But more Power
- Dynamic Frequency Scaling
  - PLLs can change output
  - Reduce/Increase based on CPU load
- Low-power modes
  - Typically multiple modes in MCU
  - Deep sleep - RTC wakeup.

Analog to Digital Conversion

Analog Sensors

Analog / Continuously Variable



Sampling / Digitization

- CPU only deals with numbers as binary

  ⇒ Convert Analog Values to Binary Numbers

Sampling / Digitization

- CPU only deals with numbers as binary

  ⇒ Convert Analog Values to Binary Numbers

- Discrete Time

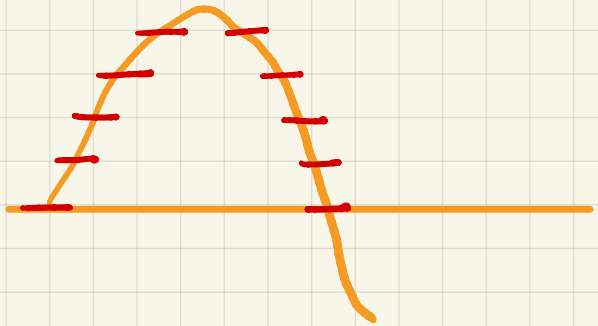
Sampling / Digitization

- CPU only deals with numbers as binary

  ⇒ Convert Analog Values to Binary Numbers

- Discrete Time

- Discrete Amplitude



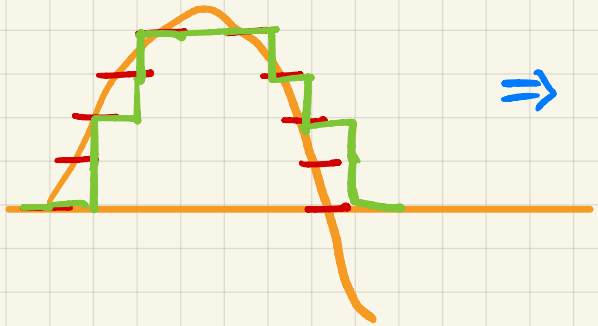
Sampling / Digitization

- CPU only deals with numbers as binary

  ⇒ Convert Analog Values to Binary Numbers

- Discrete Time

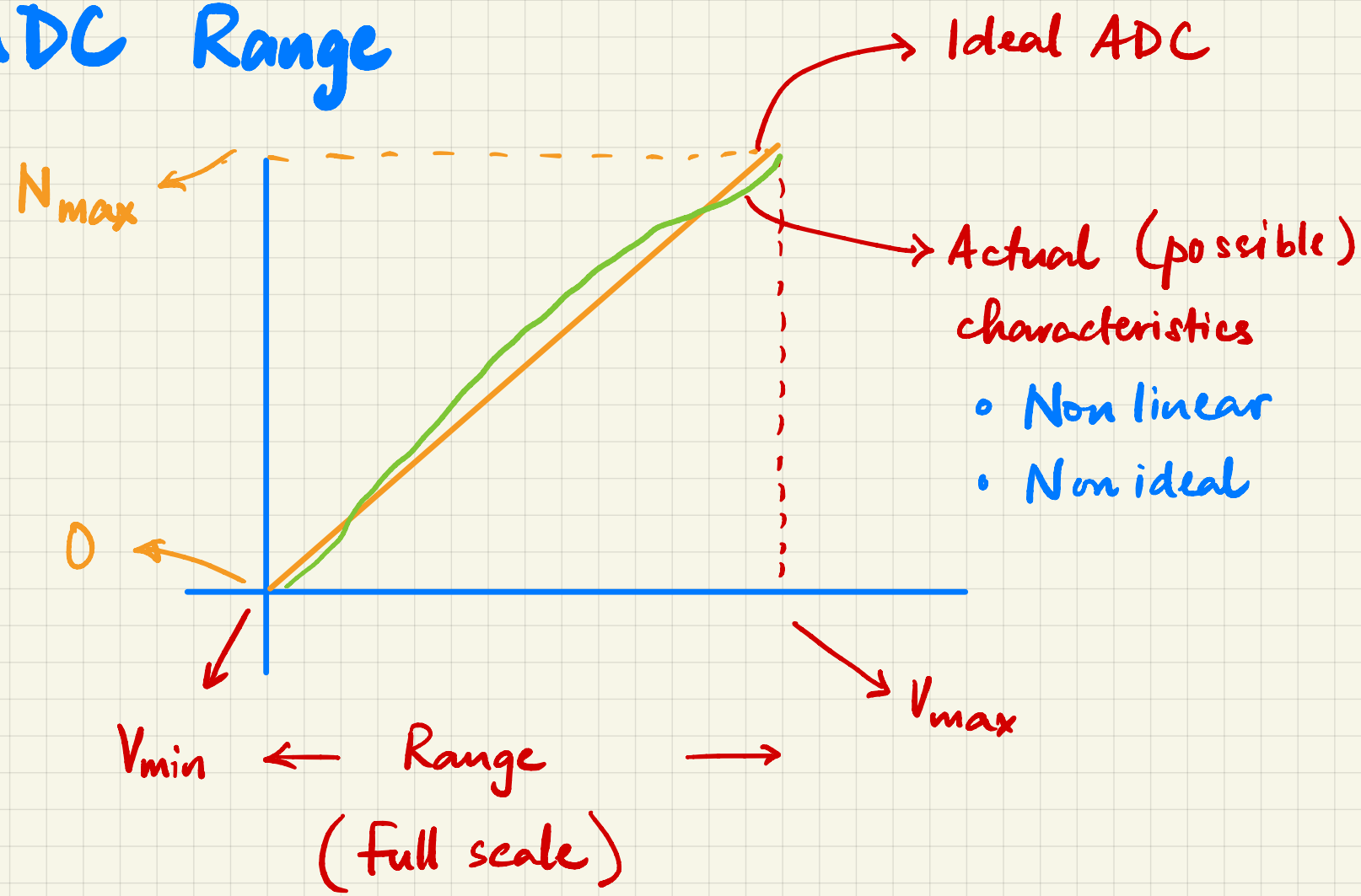
- Discrete Amplitude



⇒ Digital signal

Analog to Digital Convertor
(ADC)

ADC Range


Ideal ADC
Actual (possible) characteristics
• Non linear
• Non ideal
N_{max}
0
V_{min}
Range
(Full scale)
V_{max}

ADC Range

- N-bits $\Rightarrow$ $[0, 2^N - 1]$

- Ideally
  - $V_{min} \rightarrow 0$
  - $V_{max} \rightarrow 2^N - 1$
  - Step size $= \dfrac{V_{max} - V_{min}}{2^N}$

ADC Range

- N-bits $\Rightarrow$ $[0,\ 2^N - 1]$

  eg $N = 10 \Rightarrow [0, 1023]$

- Ideally
  - $V_{min} \longrightarrow 0$ $\longrightarrow$ 0 V
  - $V_{max} \longrightarrow 2^N - 1$ $\longrightarrow$ 3.3 V
  - Step size $= \dfrac{V_{max} - V_{min}}{2^N}$ $\longrightarrow$ $\dfrac{3.3}{1024} = 0.0033$

    $\simeq 3$ mV

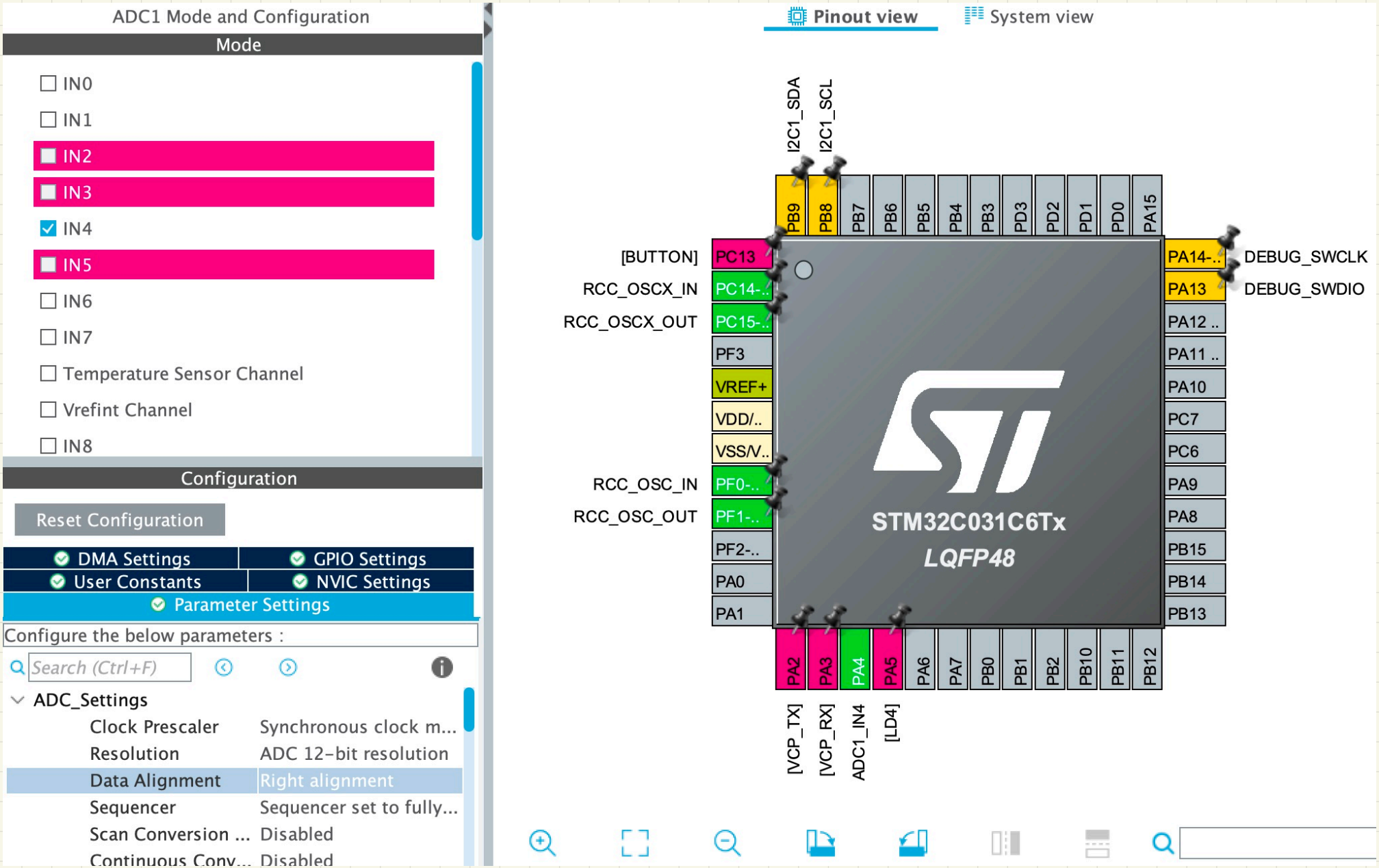
    Resolution

Sampling Rate

= sampling interval

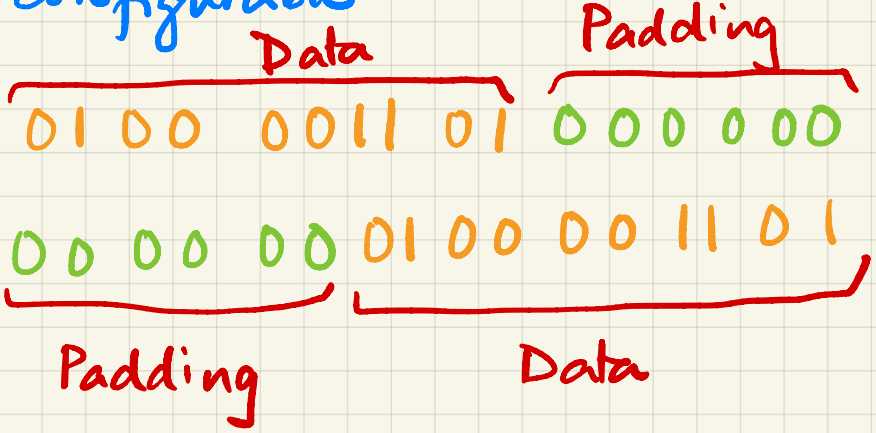
$f_s = \frac{1}{T}$ = sampling frequency

- $f_s \uparrow \Rightarrow$ better capture, more detail
  - more data to process
- Optimum / Minimum f_s : determined by signal characteristics
  - Nyquist criteria – out of scope.


ADC1 Mode and Configuration
Mode
IN0
IN1
IN2
IN3
IN4
IN5
IN6
IN7
Temperature Sensor Channel
Vrefint Channel
IN8
Configuration
Reset Configuration
DMA Settings
GPIO Settings
User Constants
NVIC Settings
Parameter Settings
Configure the below parameters :
Search (Ctrl+F)
ADC_Settings
Clock Prescaler
Synchronous clock m...
Resolution
ADC 12-bit resolution
Data Alignment
Right alignment
Sequencer
Sequencer set to fully...
Scan Conversion ...
Disabled
Continuous Conv...
Disabled
Pinout view
System view
I2C1_SDA
I2C1_SCL
PB9
PB8
PB7
PB6
PB5
PB4
PB3
PD3
PD2
PD1
PD0
PA15
[BUTTON]
PC13
RCC_OSCX_IN
PC14-..
RCC_OSCX_OUT
PC15-..
PF3
VREF+
VDD/..
VSS/V..
RCC_OSC_IN
PF0-..
RCC_OSC_OUT
PF1-..
PF2-..
PA0
PA1
PA14-..
DEBUG_SWCLK
PA13
DEBUG_SWDIO
PA12 ..
PA11 ..
PA10
PC7
PC6
PA9
PA8
PB15
PB14
PB13
STM32C031C6Tx
LQFP48
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB10
PB11
PB12
[VCP_TX]
[VCP_RX]
ADC1_IN4
[LD4]

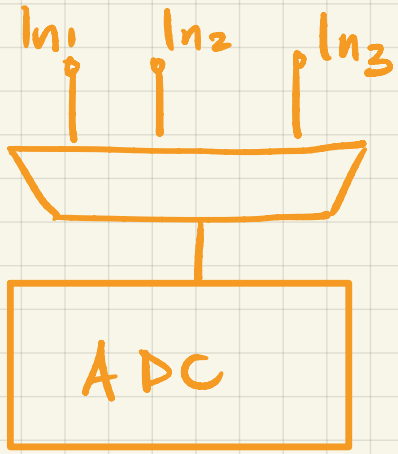
Settings

- Prescaler – Derive ADC clock from a higher clock
  – Divide down
- Resolution – 12 bit, 10 bit etc – configurable
- Alignment – Left align ⇒
  – Right align ⇒



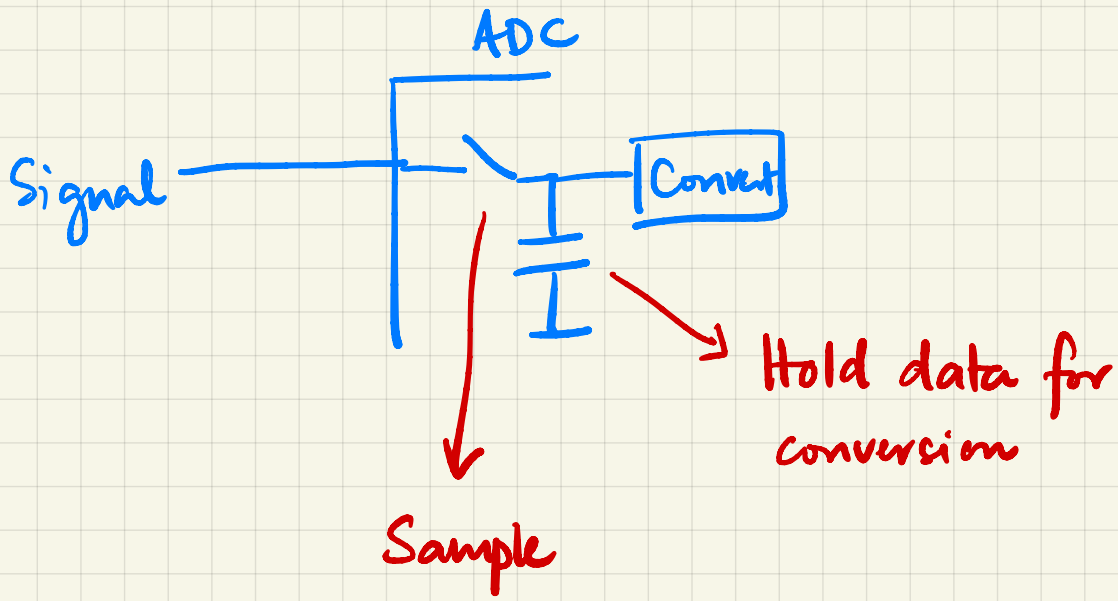
Channels

- Single ADC – Multiple Inputs



  - Scan - After converting one, auto switch to next
  - Continuous - After one conversion on all channels, automatically start next
  - Sequencer - Specify a sequence of input channels to convert

Sample-and-Hold



- Longer sample-hold times → better conversion
  → slower conversion

Trigger

- When to start ?

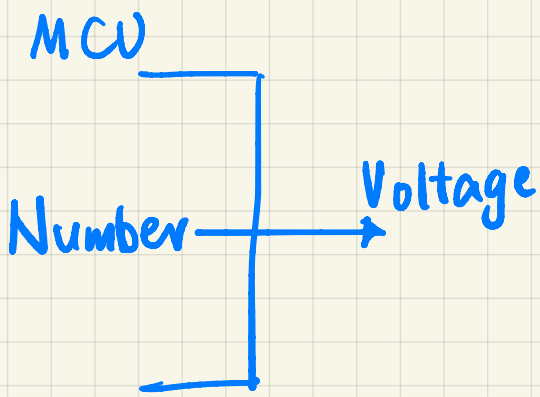
  - Automatic - sample, convert, repeat
  - Triggered - request from software
    - event / timer etc.

ADC Summary

- Interface with the Physical World - Analog Sensors
- Parameters
  - Resolution, Full range/scale, max. frequency
- MCU applications
  - Multi-channel, sequencing
  - Trigger on events, timers

Pulse Width Modulation

Digital to Analog Conversion



eg. Range $\in [0, 3.3V]$

Numeric $\in [0, 1023]$

Number = 400

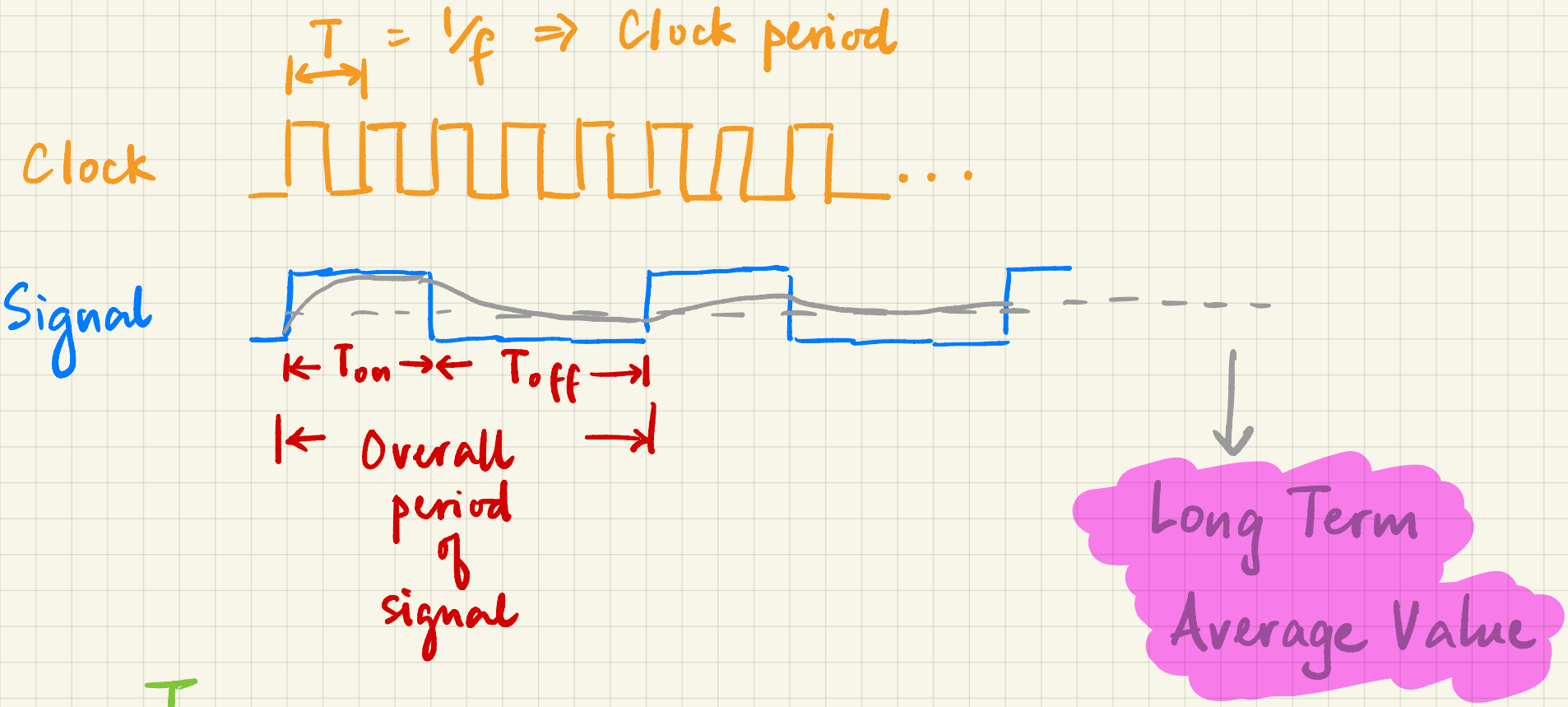
$\Rightarrow$ 1.289 V (ideal)

Quantization Noise

Random Noise

Non-idealities

Alternative



$$\frac{T_{on}}{T_{on}+T_{off}} = \text{Duty cycle}$$

Pulse Width Modulation

- High frequency clock
- Timer to count T_{on}, then T_{off}

  $\Rightarrow$ any duty cycle

  - Resolution = T (high frequency clock period)
- Average signal value
  - Low pass filtering / implicit averaging

Low complexity Digital to Analog Conversion

Typical setup process

```
void setupPWM() {
    // Set the timer to generate a 1 kHz PWM signal
    // Prescaler = 16 (assuming a 16-bit timer)
    // Timer count for 1 kHz = 16000 (16 MHz / 1000 Hz)
    TIMER->PRESCALER = 16;
    TIMER->PERIOD = 1000; // Period for 1 kHz frequency

    // Set duty cycle to 50%
    TIMER->COMPARE = 500; // 50% of 1000 is 500

    // Enable PWM mode on the timer
    TIMER->MODE = PWM_MODE;

    // Start the timer
    TIMER->START();
}
```

Applications

Analog Output Control

- LED brightness
- Motor speed
- Power Delivery (switched mode power supply - SMPS)
- Communication - waveform generation
- Temperature Control
- Generic D/A conversion

Pros:

- Simplicity
- Efficiency (on/off)
- Precision (clock controlled)

Cons

- Resolution/Response Time
- Harmonics, EMI, audio
- Interfere with analog

Timers

Timers

Versatile Peripherals

- Delay generation, Pulse Width Modulation
- Event Counting (frequency measurement etc.)
- Input Capture
- Generate output events
- Timekeeping, Watchdog

STM32 Timers

TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 15.3.26: Timer synchronization*.

TIM3 introduction

The general-purpose timer TIM3 consists of a 16-bit auto-reload counter driven by a programmable prescaler.

The timer may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

- Advanced Operation Modes
- Multiple Timer Circuits - Independent Operation
- CPU only initializes and responds
  - separate hardware

Configuration

- Clock Source
- Mode (up, down, PWM)
- Period & Prescaler
- Interrupts

Watchdog

"Hanging" microcontroller

Stuck in a loop - bad code or hardware error

Solution: Timeout → reset system if not responding

- Watchdog Timer - runs independently, counts down
- Main program must "pat" the watchdog before expiry
  - else WDT resets system
- Timeout - set long enough to avoid false reset

Watchdog Uses

- Embedded systems – esp. safety critical
  - unattended operation

Pros

- Reliability
- Error state detection
- Automatic Recovery

Cons

- False resets
- Limited Debugging
- Ultra-low power - resources