# CSCE 438/838 IoT Lab

# Lab 1

# Lab 2 - Clocks, Timers and Interrupts

# Content

- Clock
- Timer
- Interrupt

# Lab 1 - WDT

- Getting familiar with the Arduino environment and program
- Controlling the registers of a system peripheral
- We will apply the same approach in this lab

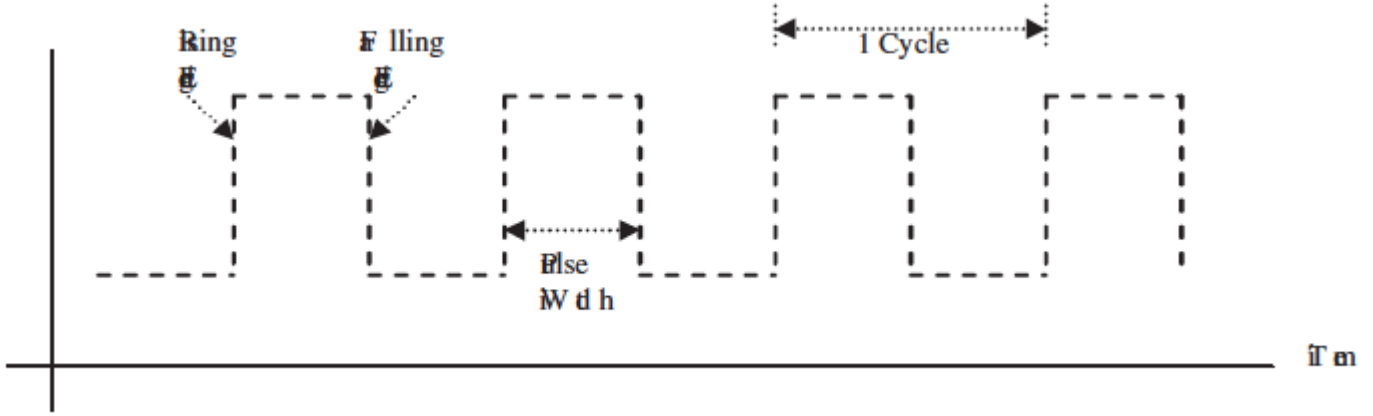
- WDT is a type of a timer

# Timer

- Timer: A counter that is updated based on a clock signal
  - Timer = Counter + Clock
- Clock: A periodic signal with a certain frequency (clock freq.)
  - A single period is called a tick
- Any timer
  - holds the number of ticks,
  - at the clock frequency,
  - since a particular instance in time
- ES Clock ≠ Wall clock
- ES Timer ≠ Wall clock (except RTC)

# Why Clocks?

- Computer Architecture
  - Embedded system architecture
    - MCU architecture
- CPU needs a clock source
- Peripherals need clock to be synchronized
- Clock signal



# Clock (Oscillator)

- MCU needs a clock source
- MCU spends most of its time in a low-power mode
- Need a precise clock source to wake up at certain (real) times
- Need a clock source for time stamping
- High frequency (HF) clock: Can be started and stopped rapidly, need not be very accurate
- Low frequency (LF) clock: Run continuously to track real time, low power, accurate

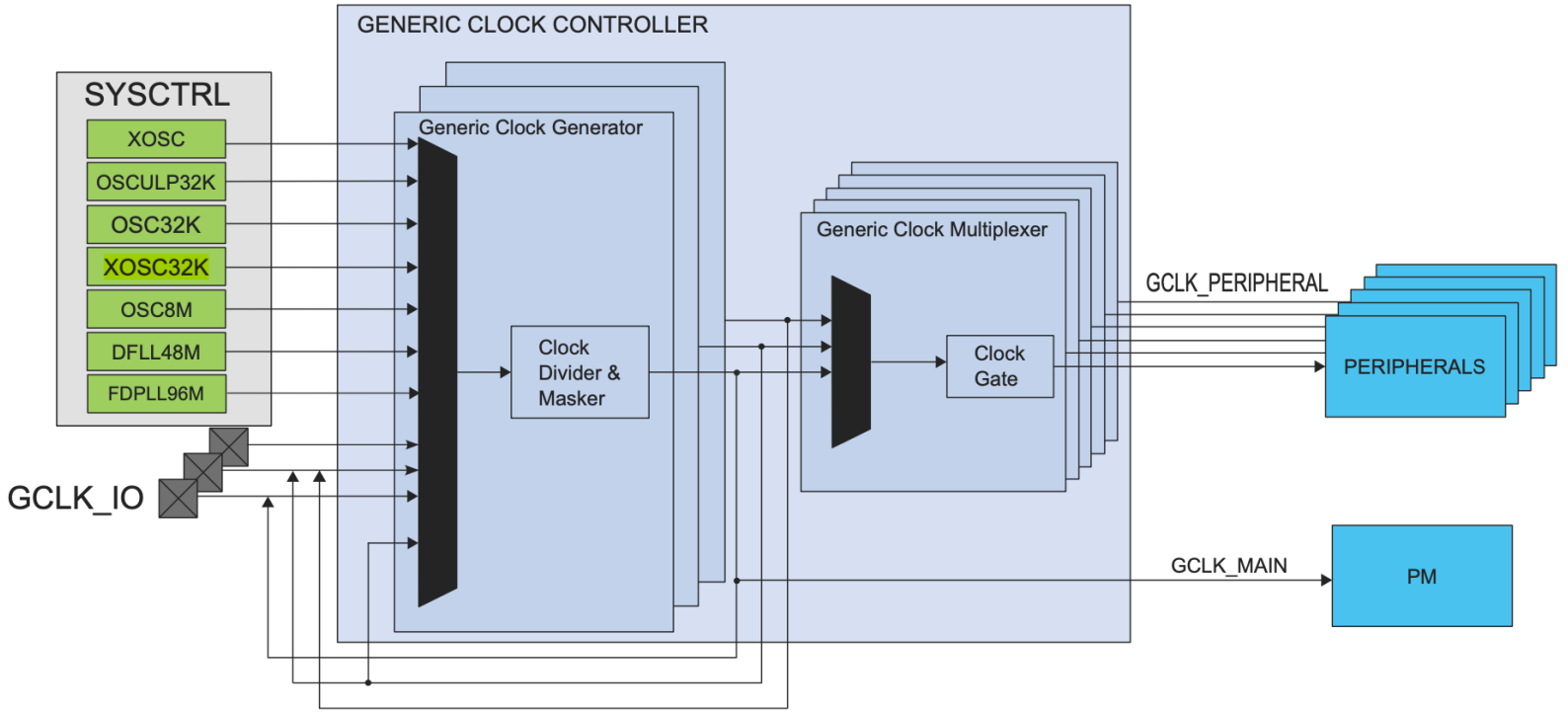
# Clock Sources

- Generally two types
  - Crystal: Accurate and stable but expensive and delicate.
  - Resistor and capacitor (RC): Cheap and quick to start but used to have poor accuracy and stability.
- Clock sources on SAMD21

| | |
|---|---|
| Oscillators | 32.768kHz crystal oscillator (XOSC32K)<br>0.4-32MHz crystal oscillator (XOSC)<br>32.768kHz internal oscillator (OSC32K)<br>32kHz ultra-low-power internal oscillator (OSCULP32K)<br>8MHz high-accuracy internal oscillator (OSC8M)<br>48MHz Digital Frequency Locked Loop (DFLL48M)<br>96MHz Fractional Digital Phased Locked Loop (FDPLL96M) |

# Clock System on SAMD21

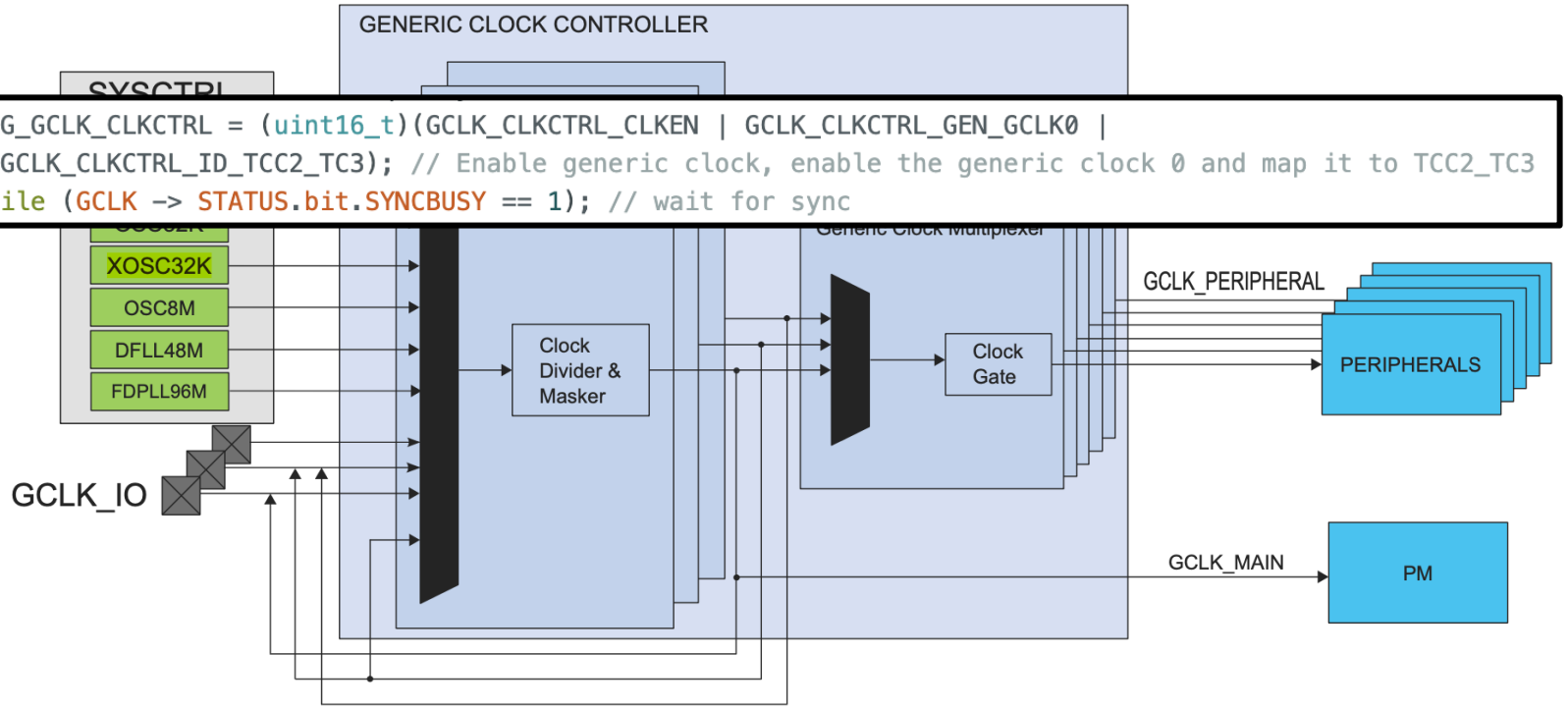
- Clock sources (SYSCTRL)
  - Pick a clock source
- Generic Clock Controller (GCLK):
  - Provide the clock input for the system and peripherals
  - Generic Clock Generators: Programmable prescalers to convert a clock source signal into a desired frequency
  - Generic Clock Generator 0 generates the clock signal GCLK_MAIN
  - Generic Clocks: Serve as clocks for the peripherals
- Power Manager (PM)
  - Generates and controls the synchronous clocks in the system



# Clock System on SAMD21

- Clock sources (SYSCTRL)
  - Pick a clock source
- Generic Clock Controller (GCLK):
  - Provide the clock input for the system and peripherals
  - Generic Clock Generators: Programmable prescalers to conve a clock source signal into a desired frequency
  - Generic Clock Generator 0 generat the clock signal GCLK_MAIN
  - Generic Clocks: Serve as clocks for the peripherals
- Power Manager (PM)
  - Generates and controls the synchronous clocks in the system

```
REG_GCLK_CLKCTRL = (uint16_t)(GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK0 |
  GCLK_CLKCTRL_ID_TCC2_TC3); // Enable generic clock, enable the generic clock 0 and map it to TCC2_TC3
while (GCLK -> STATUS.bit.SYNCBUSY == 1); // wait for sync
```



# Timers

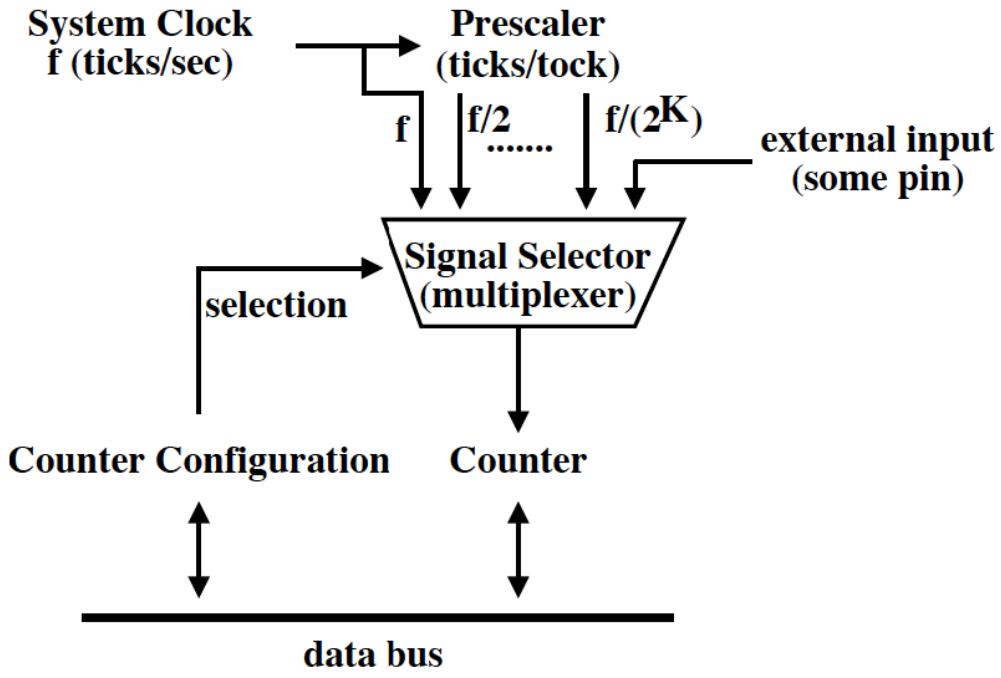
- Core of effective embedded system operation
- Know every detail of the available timers in your ES
- Generally, three types
  - Watchdog timer
  - Generic timer
  - Real time clock (RTC)

# Why timers?

- The system timing is fundamental to nearly every embedded application;
- The main applications of timers:
  - Generate events of fixed time-period (sampling, PWM)
  - Allow periodic wakeup from sleep of the device
  - Count transitional signal edges
  - Schedule tasks
  - Replacing delay loops with timer calls allows the CPU to sleep between operations, thus consuming less power

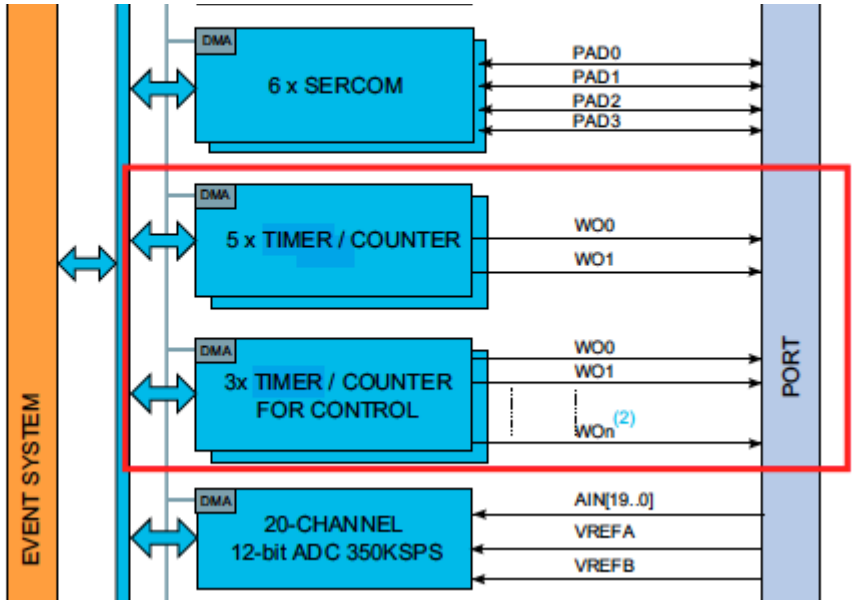
# Timer

- Clock Source
- Counter: A counter is a register that saves number of times that the detector signal transitions from high to low.
- Prescaler: We can further divide the input clock frequency by a range of different divisors.
- Signal Selector: For comparing and capturing external signal



# Timer/Counter (TC) on SAMD21

- Features of TC on SAMD21
  - Selectable configuration
  - 8-, 16- or 32-bit TC, with compare/capture channels
  - Waveform generation
    - Frequency generation
    - Single-slope pulse-width modulation
  - Input capture
    - Event capture
    - Frequency capture
    - Pulse-width capture
  - One input event
  - **Interrupts/output events on:**
    - **Counter overflow/underflow**
    - **Compare match or capture**
  - Internal prescaler



# What is an interrupt?

- An interrupt is an input signal to the processor indicating an event that needs immediate attention
  - Low level programming concept
  - Extremely important – used extensively in modern computer programs
- The processor checks whether events have occurred, gives suitable responses and performs periodic tasks
- Interrupts “interrupt” the normal flow of a program
- The processor stops the normal program, handles the interrupt, and then resumes its normal work
- Interrupt handler (Interrupt service routine – ISR): Needs to be short, quick, to the point

# Why interrupt?

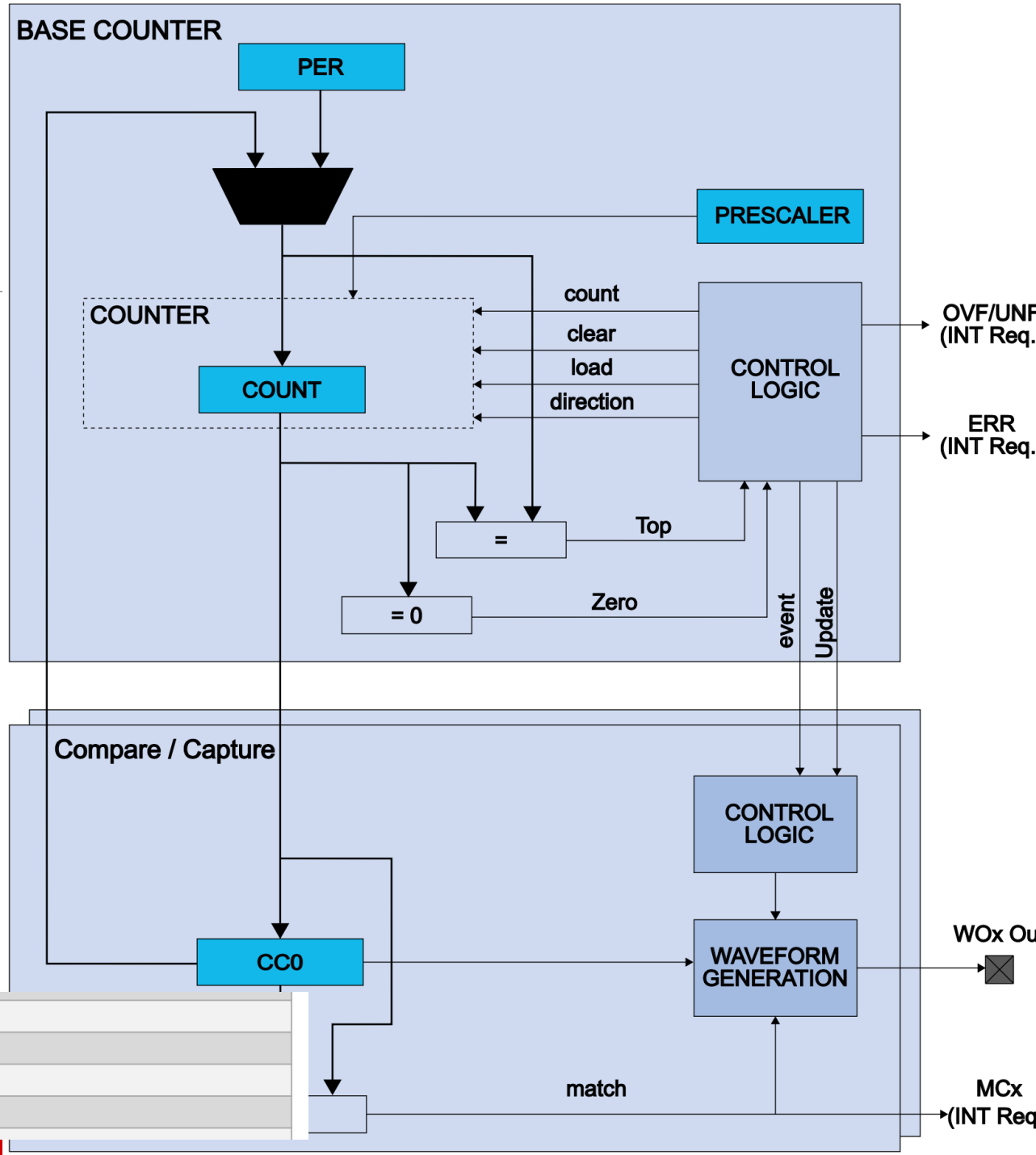
- Consider the main loop() in Arduino: A sequence of instructions executed serially, jumps are allowed
- Programs for embedded systems usually service real-life demands
- Some events occur and we cannot wait

# Timers and Interrupts

- Timers can generate interrupts:
  - **Counter overflow/underflow**
  - **Compare match or capture**
  - Generate periodical events/waveforms
- *How?*

# SAMD21 Timers/Counters

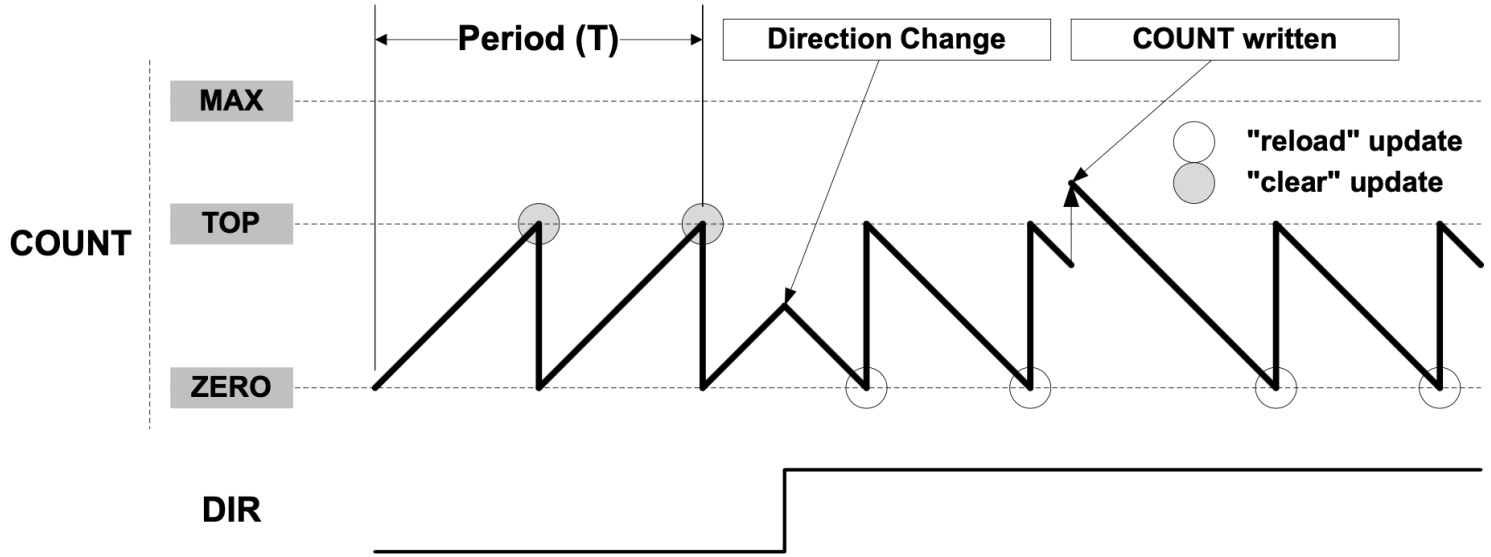
- TC = counter, prescaler, compare/capture channel (CC), control logic
- Functionalities
  - Waveform generation
  - Input capture
    - Event capture
    - Frequency capture
    - Pulse-width capture
  - Interrupts
    - Counter overflow
    - Compare match
    - Capture
- TC instances are paired, even and odd, starting from TC3, and use the same generic clock GCLK_TCx




| | | |
|---|---|---|
| 0x1A | GCLK_TCC0, GCLK_TCC1 | TCC0,TCC1 |
| 0x1B | GCLK_TCC2, GCLK_TC3 | TCC2,TC3 |
| 0x1C | GCLK_TC4, GCLK_TC5 | TC4,TC5 |
| 0x1D | GCLK_TC6, GCLK_TC7 | TC6,TC7 |

# TC

- Counter Mode
  - Can either count events or clock ticks
  - Can count up or down
  - Wraps around to zero (cnt up) or to TOP (cnt down) and can be configured to generate an interrupt then
  - COUNT register can be read and written
- CCx – Compare/capture registers
  - Counter value is passed to CCx to compare to values or captured




| Name | Description |
|---|---|
| TOP | The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be the same as Period (PER) or the Compare Channel 0 (CC0) register value depending on the waveform generator mode in Waveform Output Operations. |
| ZERO | The counter is ZERO when it contains all zeroes |
| MAX | The counter reaches MAX when it contains all ones |
| UPDATE | The timer/counter signals an update when it reaches ZERO or TOP, depending on the direction settings. |
| Timer | The timer/counter clock control is handled by an internal source |
| Counter | The clock control is handled externally (e.g. counting external events) |
| CC | For compare operations, the CC are referred to as “compare channels”<br>For capture operations, the CC are referred to as “capture channels.” |

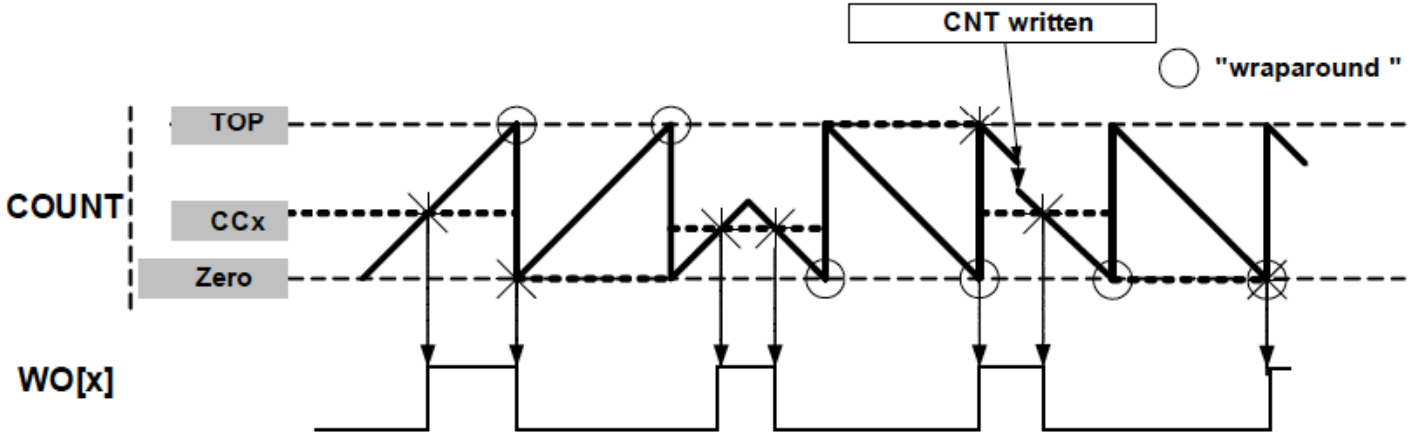
# Timer mode

- **Timer Waveform Generation Operation/Frequency Operation Mode**

**1. Normal Frequency Operation (**NFRQ**)**

- When NFRQ is used, the waveform output (WO[x]) toggles every time **CCx and the counter are equal**, and the interrupt flag corresponding to that channel will be set. The top value is the maximum value that the timer allows.

**Figure 29-4. Normal Frequency Operation**




For example, top value of the 16-bit mode is 2^16 = 65536.

You can write to change the COUNT value

# Timer mode

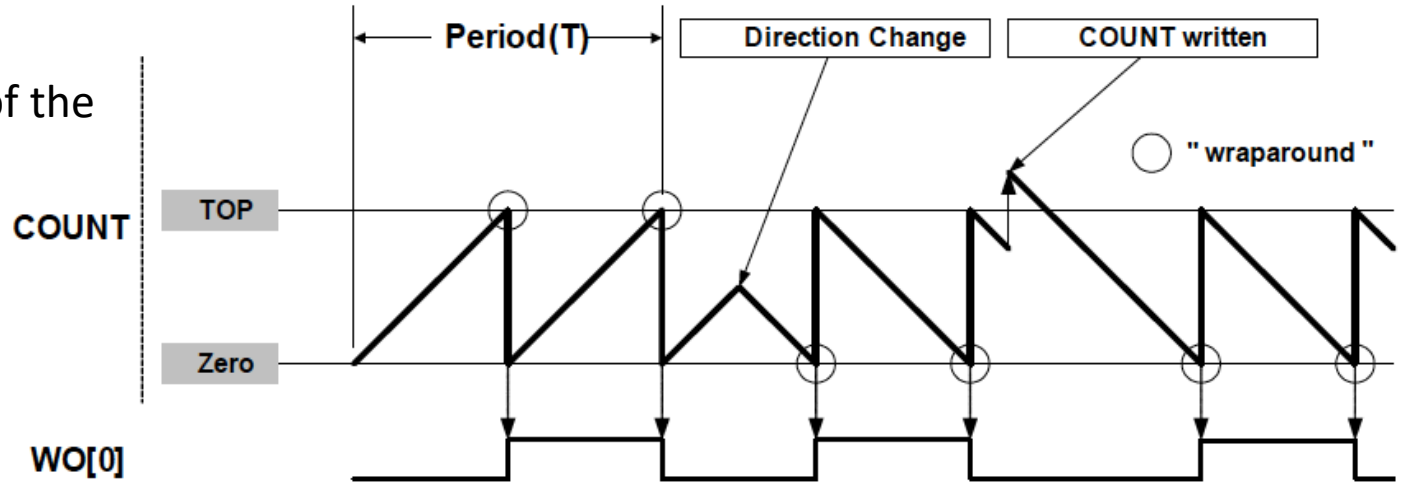
- **Timer Waveform Generation Operation/Frequency Operation Mode**

**2. Match Frequency Operation (**MFRQ**)**

- When MFRQ is used, the value in **CC0 will be used as the top value** and WO[0] will toggle on every overflow/underflow.

For example, if you write 35000 to CC0 of the 16-bit timer, the top value is 35000.

**Figure 29-5. Match Frequency Operation**



# Interrupt Priorities

- Ranked by emergency
  - Non-maskable interrupt (NMI)
  - Maskable interrupt: can be overwritten

| Peripheral Source | NVIC Line |
|---|---|
| EIC NMI – External Interrupt Controller | NMI |
| PM – Power Manager | 0 |
| SYSCTRL – System Control | 1 |
| WDT – Watchdog Timer | 2 |
| RTC – Real Time Counter | 3 |
| EIC – External Interrupt Controller | 4 |
| NVMCTRL – Non-Volatile Memory Controller | 5 |
| DMAC - Direct Memory Access Controller | 6 |
| USB - Universal Serial Bus | 7 |
| EVSYS – Event System | 8 |
| SERCOM0 – Serial Communication Interface 0 | 9 |
| SERCOM1 – Serial Communication Interface 1 | 10 |
| SERCOM2 – Serial Communication Interface 2 | 11 |
| SERCOM3 – Serial Communication Interface 3 | 12 |
| SERCOM4 – Serial Communication Interface 4 | 13 |
| SERCOM5 – Serial Communication Interface 5 | 14 |
| TCC0 – Timer Counter for Control 0 | 15 |
| TCC1 – Timer Counter for Control 1 | 16 |
| TCC2 – Timer Counter for Control 2 | 17 |
| TC3 – Timer Counter 3 | 18 |
| TC4 – Timer Counter 4 | 19 |
| TC5 – Timer Counter 5 | 20 |
| TC6 – Timer Counter 6 | 21 |
| TC7 – Timer Counter 7 | 22 |
| ADC – Analog-to-Digital Converter | 23 |
| AC – Analog Comparator | 24 |
| DAC – Digital-to-Analog Converter | 25 |
| PTC – Peripheral Touch Controller | 26 |
| I2S - Inter IC Sound | 27 |

# SAMD21 Interrupt Controller

- **SAMD21 Nested Vector Interrupt Controller (NVIC)**
- **For timers**
  - Built-in functions:
    - NVIC_DisableIRQ(TCx_IRQn);
    - NVIC_ClearPendingIRQ(TCx_IRQn);
    - NVIC_SetPriority(TCx_IRQn, 0);
    - NVIC_EnableIRQ(TCx_IRQn);
  - Interrupt service handler:
    - void TCx_Handler()

# Timer registers

- Functions: Section 29.6 on datasheet
- Registers: Section 29.7-29.8 on datasheet

# Example code

- See example code
- Run it
- Understand the function of those registers
- Let’s go through the instruction PDF

# Assignment

- Let’s go through the instruction PDF
- Report Format