CprE 288 – Introduction to Embedded Systems (Output Compare and PWM)

Instructors:

Dr. Phillip Jones

QUIZ Rules

- **Only 1-side of a page of notes**
- **Can use a calculator**
- **Can use a blank sheet to workout problems**
- **Using any additional material is a 0 for the Quiz**

Note: NEVER allowed to use Datasheet or Textbook for quizzes.

Exam Rules

- Exam Rules:
  - Can use textbook, datasheet, 1 page of notes (both sides) 10pt+ font, and calculator allowed
  - Electronic textbook and electronic Datasheet is fine.
  - Can use a blank sheet to workout problems
  - Nothing else can be used or **you will receive an F for CPRE 288**

Announcements

- **Class Project Related:**
  - **Project Demos:** During you lab session Prep-week
    - 5pts bouns + extra time if demoed the week before Prep-week
- **Exam 2:** Tuesday Nov 19 (11/19), **in class using Canvas**

Textbook & Data Sheet: Read and ask questions

- Exam 2 will predominantly consist of questions of the form
  - **Configure Registers to meet given specifications**
    - **UART, ADC, Input Capture, Output Compare, Timers, Interrupts**
    - **Each device has a section in the Datasheet and Textbook**
  - **Based on a given configuration, answer questions about how the program will behave**
    - **E.g. How long will something take to occur?**
    - **E.g. How many time a second with something occur?**
  - **Explain why a given configuration is incorrect for implementing a specified behavior**
  - **Assuming a given configuration, write a short program to implement a specific behavior**
  - **ADC calculation problem**

Overview of Today's Lecture

- Output Compare and Pulse Wave Modulation (PWM)
  - Datasheet: Chapter 11
  - Textbook reading: Section 9.1, 9.2

OUTPUT COMPARE

Output Compare

Output Compare: specify time at which to generate an event

Allows one to generate (i.e. output) a waveform.

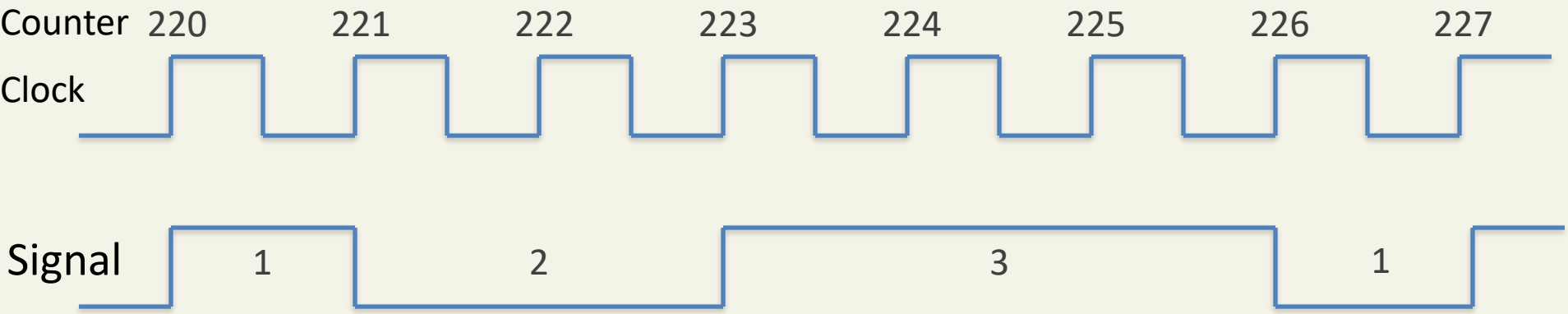
Many applications in microcontroller applications:

- Start analog devices
- Control speed of motors
- Control power output rate
- Communications
- Control servo (e.g. Servo Lab)

Recall, Input Capture: Capture the time of an event

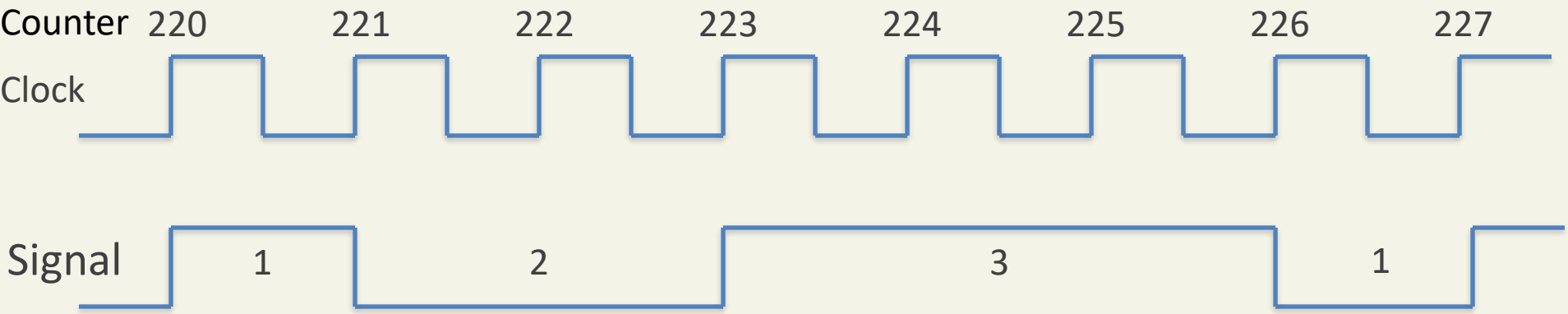
Output Compare

- Example: Generate a waveform that is 1-cycle high, 2-cycle low, 3-cyle high, 1-cycle low, and repeating



Output Compare

- Example: Generate a waveform that is 1-cycle high, 2-cycle low, 3-cyle high, 1-cycle low, and repeating
- Generate output events (transitions) at 220 (current time), 221, 223, 226, 227 and so on with initial state as low



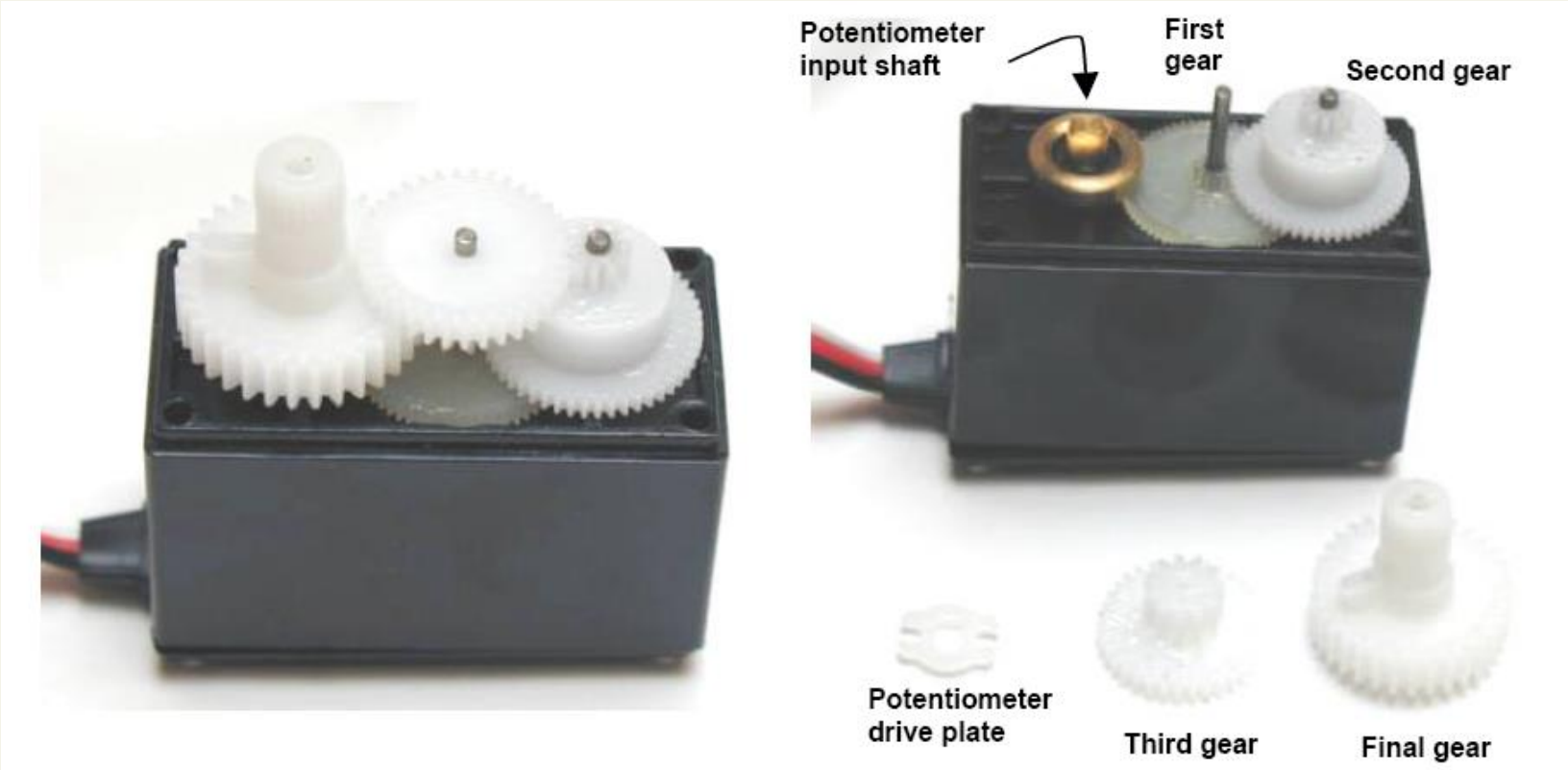
Servo Control

A **servo** is a special motor with built-in position feedback

- Can stop the shaft at a given position
- Relatively precise
- **Needs calibration**

Servo Control

The potentiometer plays as position sensor (see the next two slides)



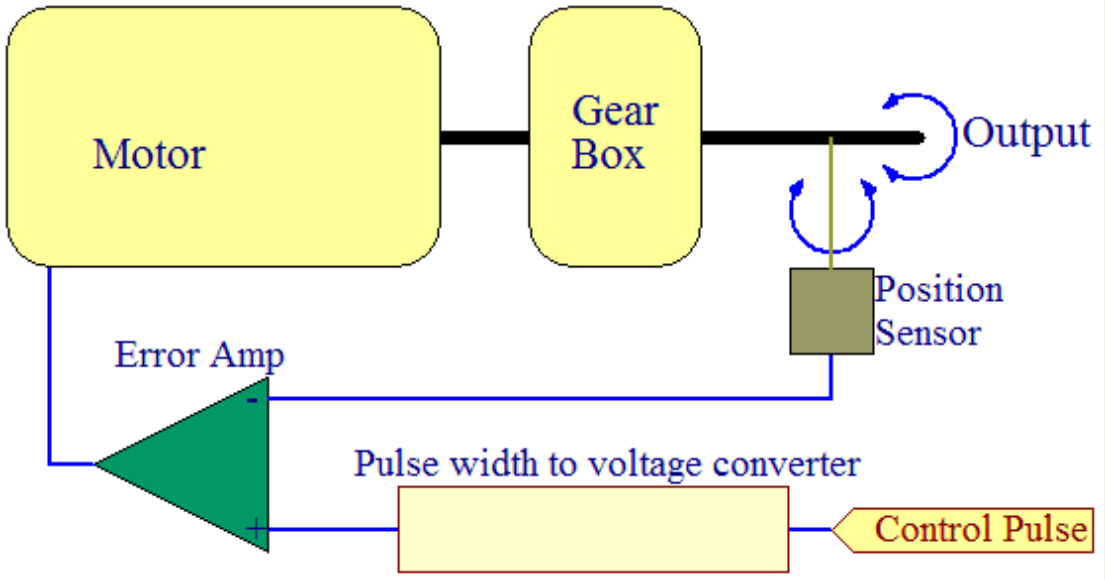
Source: Parallax Robotics Student guide, V1.4

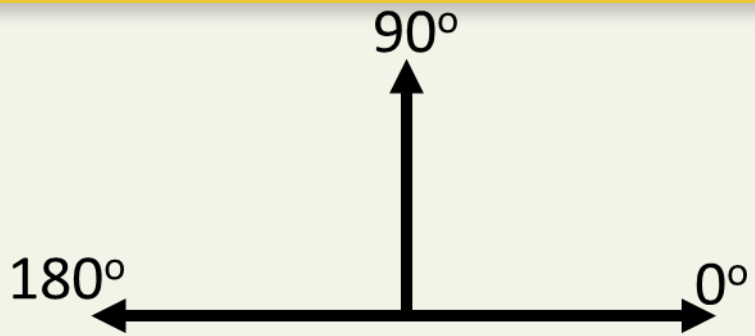
Servo Control

http://en.wikipedia.org/wiki/Potentiometer

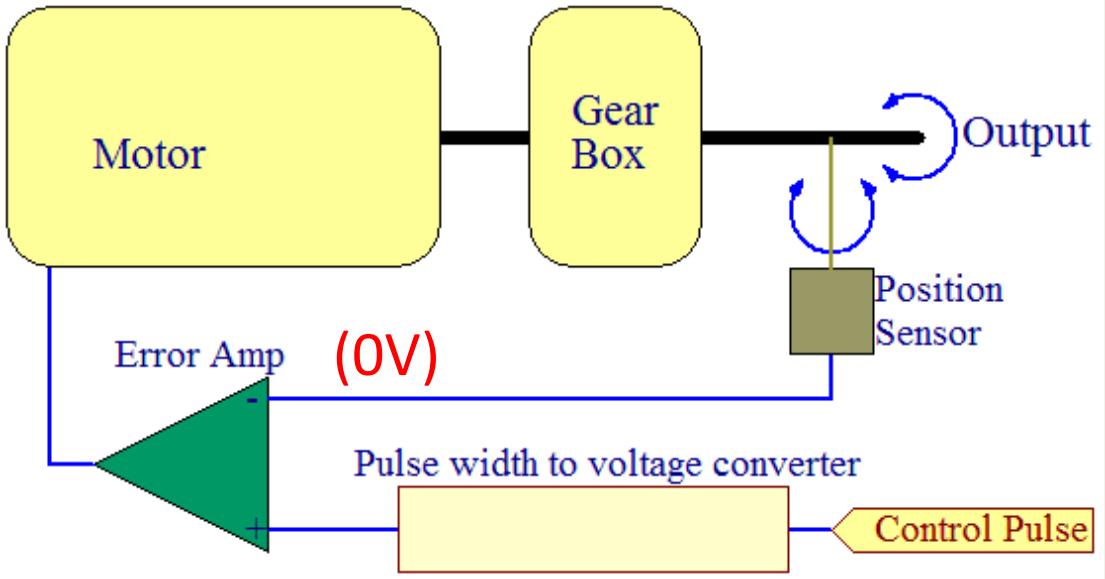
- Potentiometer: Three-terminal resistor with a sliding **mid contact**
- In the servo, the motor rotates the shaft that slides the mid contact
- The voltage at the mid contact provides feedback to the power circuits driving the motor

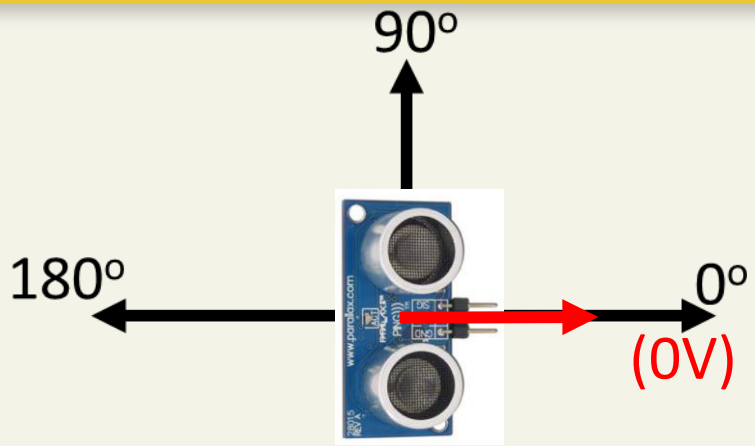
Servo Control



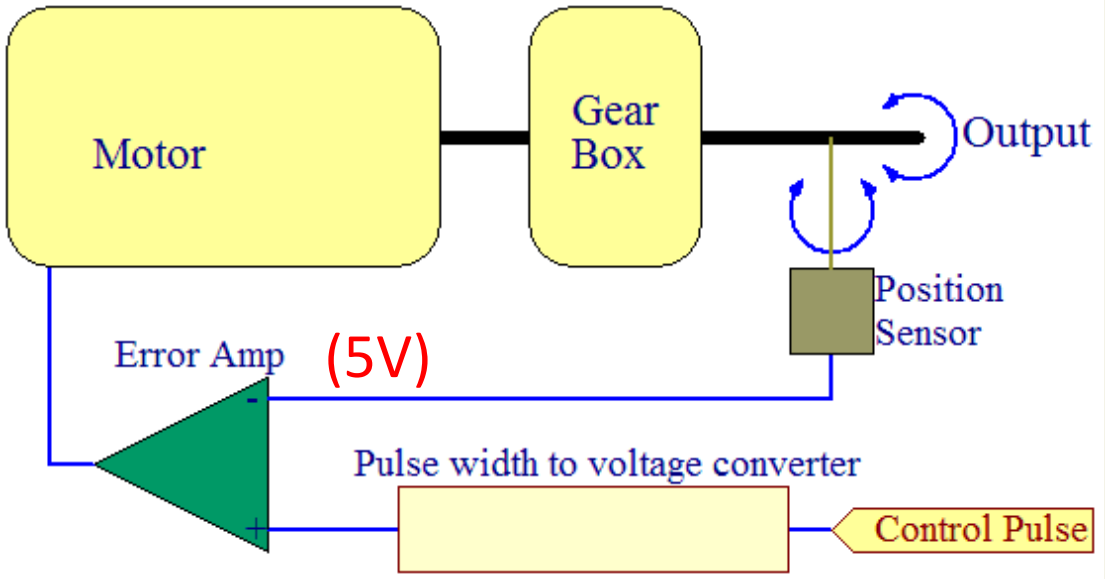


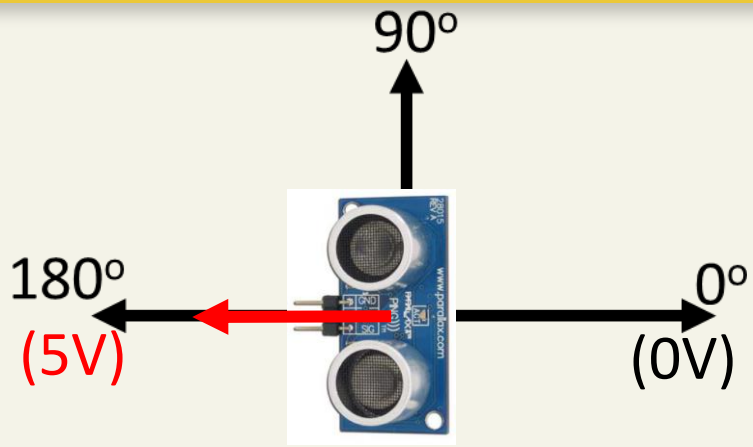
Servo Control



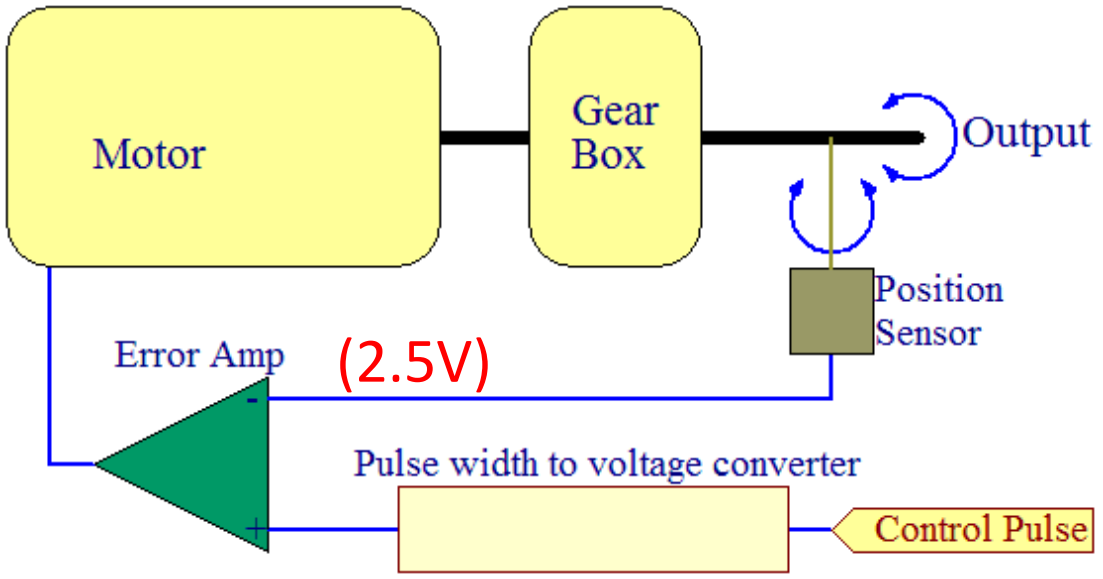


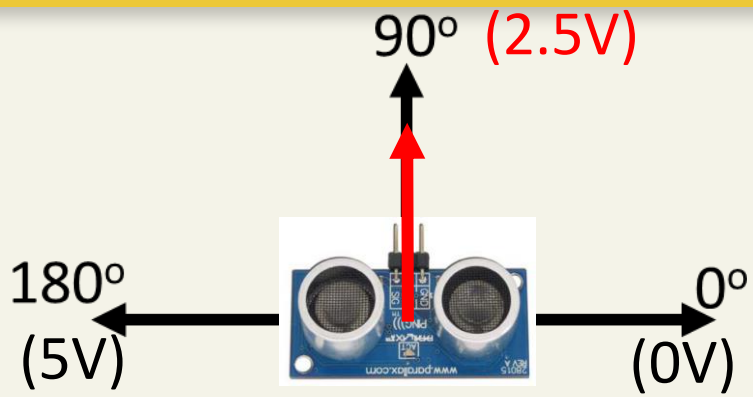
Servo Control



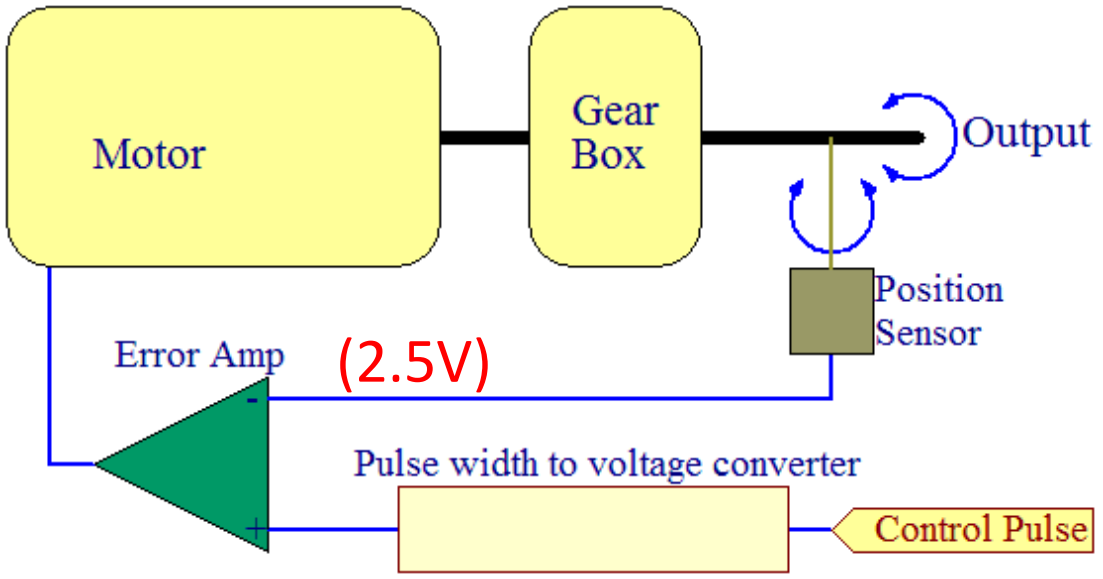


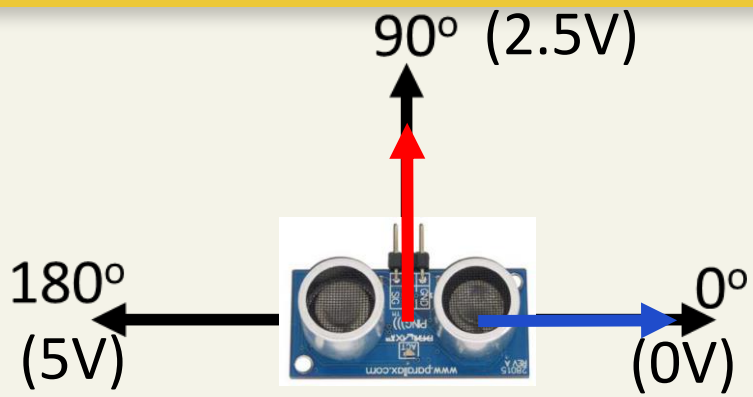
Servo Control



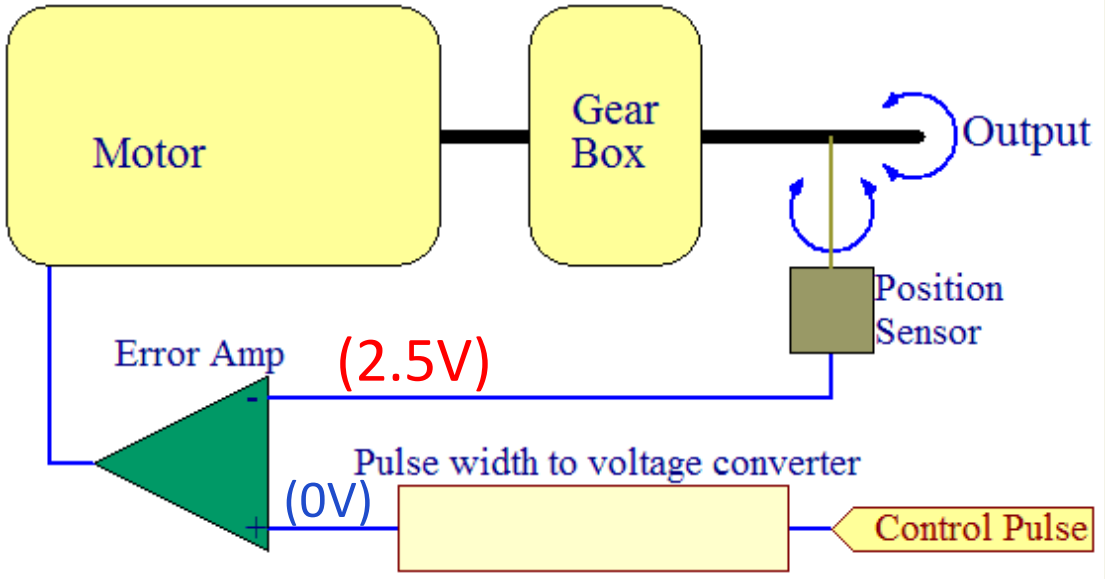


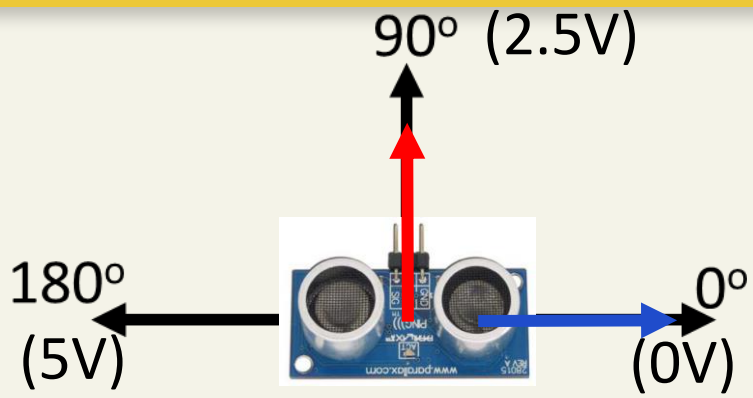
Servo Control





Servo Control





Control feedback loop:

1) A control pulse is converted to a target voltage
2) If the servo is not at the **target angular position**, there will be a **error** between the target voltage and the position sensor voltage
3) The voltage error is amplified and used to turn motor in the direction that reduces the voltage error toward zero

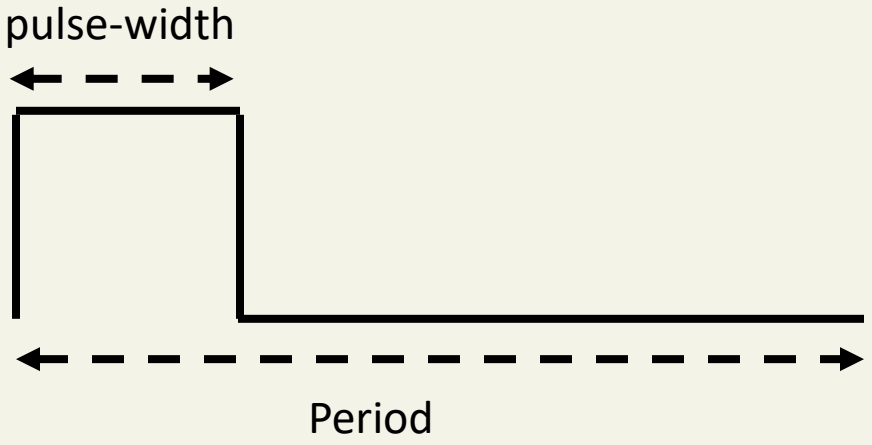
Pulse Width Modulation = PWM

Parameters: Period and Pulse Width

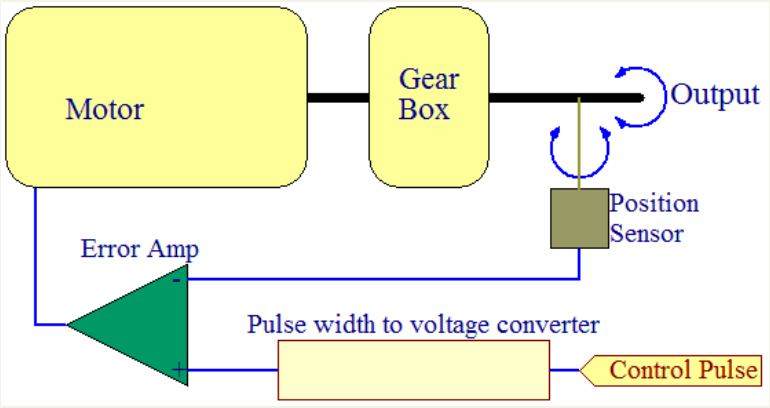
Duty Cycle = Pulse width / Period

Programming: How to set the two parameters?

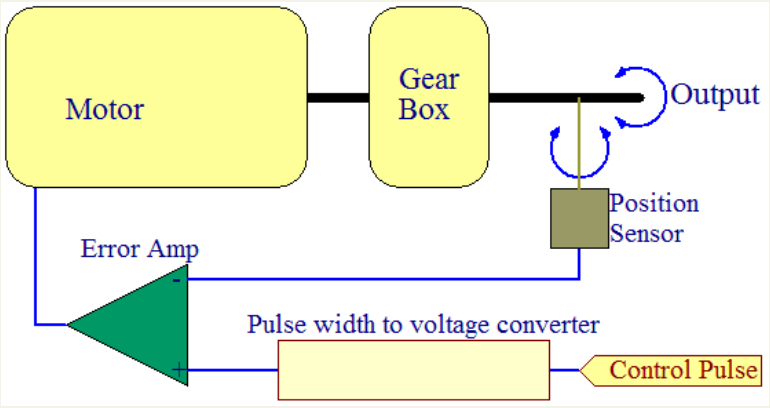
- Textbook: 9.2.3.7 PWM Mode: Steps 1-5



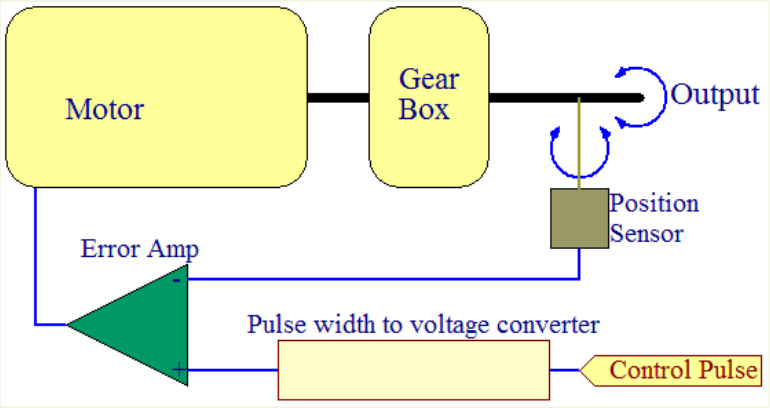
Using PWM to Generate a Voltage Level



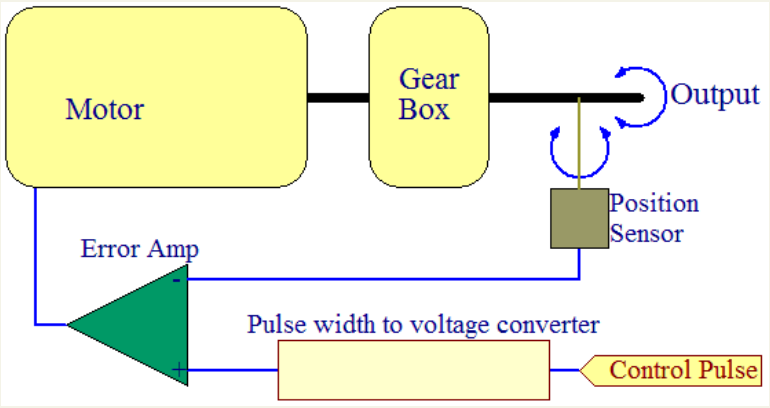
Using PWM to Generate a Voltage Level



Using PWM to Generate a Voltage Level



Using PWM to Generate a Voltage Level



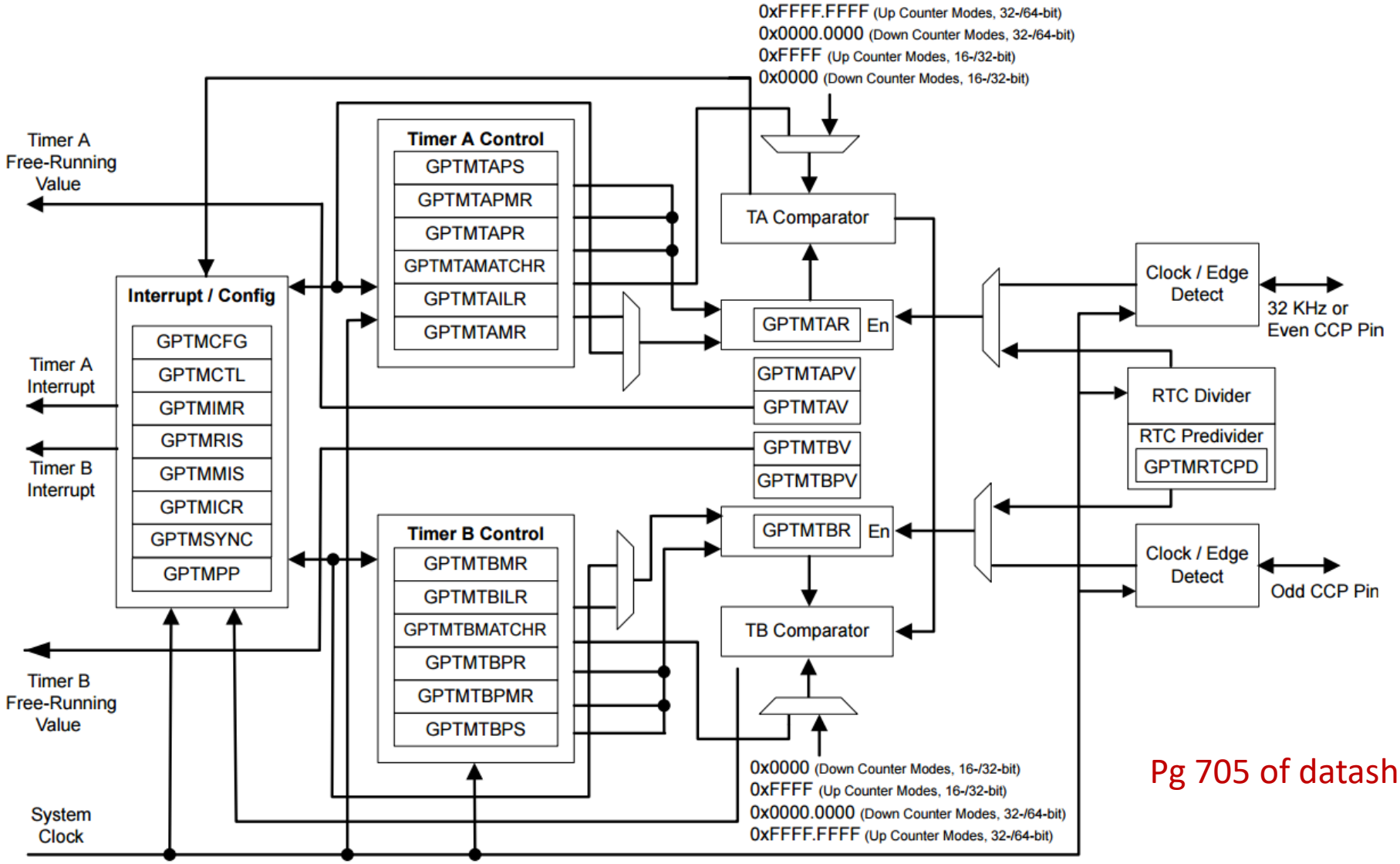
TM4C123G Timer module

6 general purpose 16/32-bit timer blocks

- 11 timer modes
- **2 independent matching units per block (A / B)**
- Pulse width modulation output
- Many other features

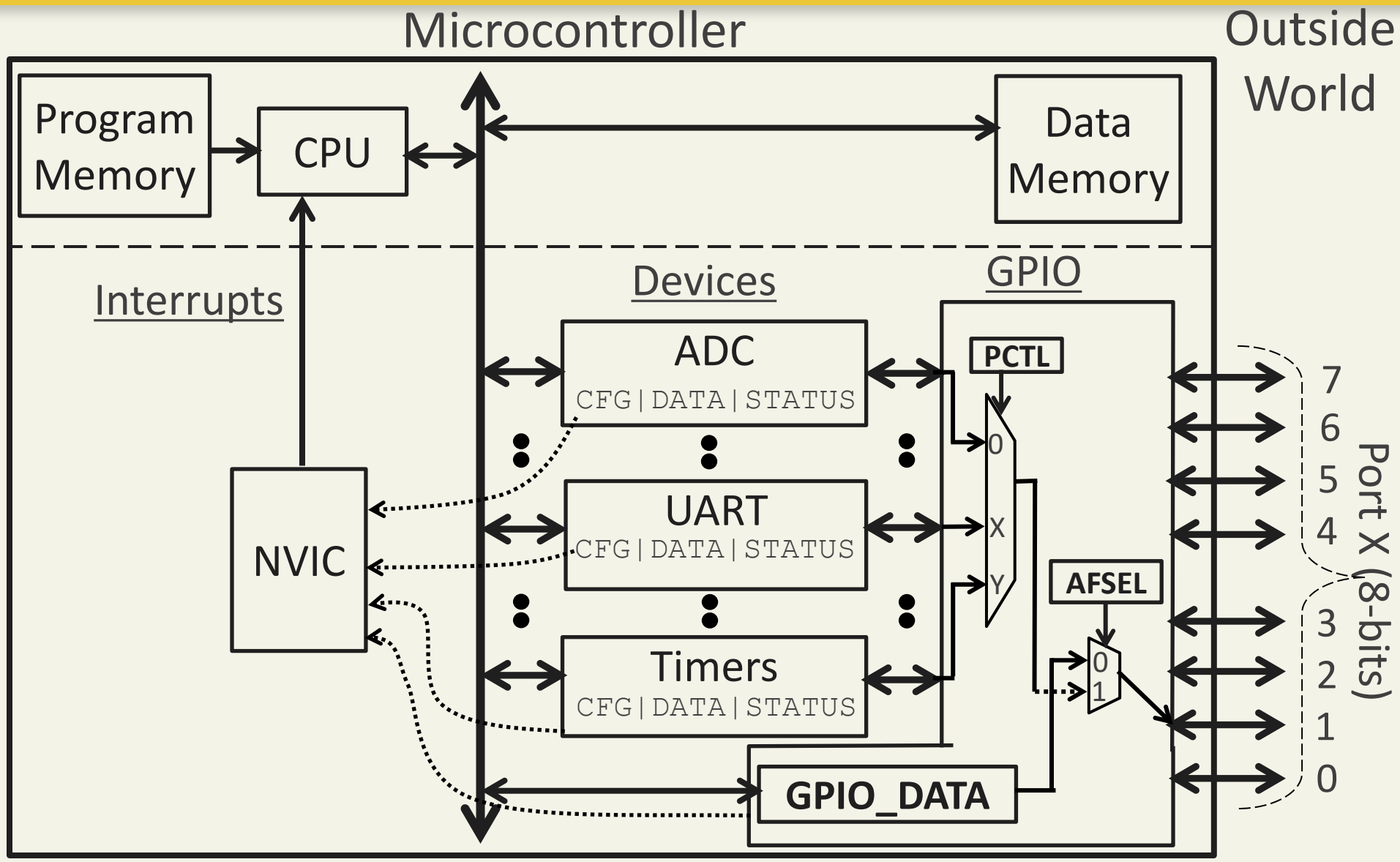
***Note that there is also a separate PWM module (Not used in CPRE 288)

Timer Block Diagram

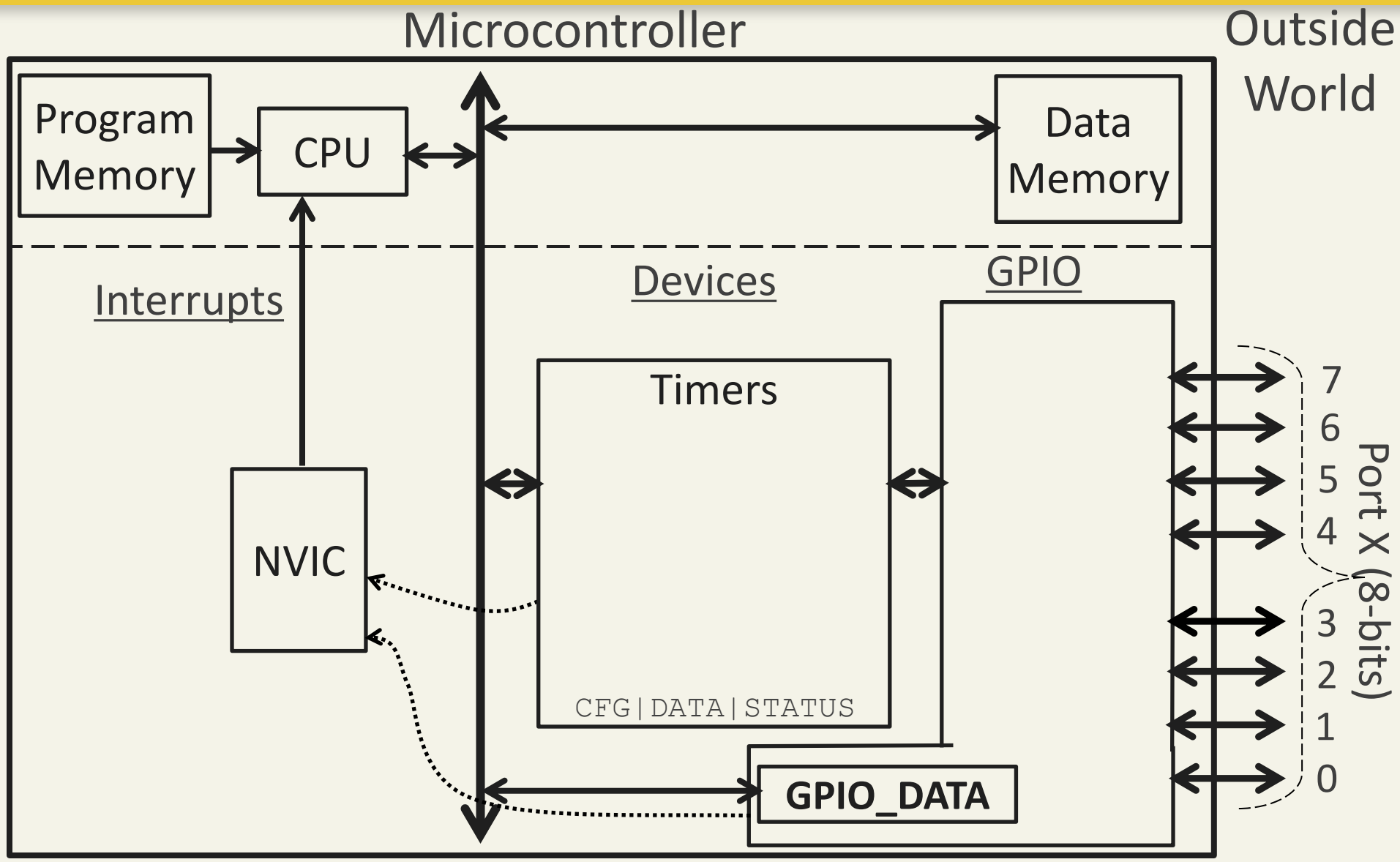


Pg 705 of datasheet

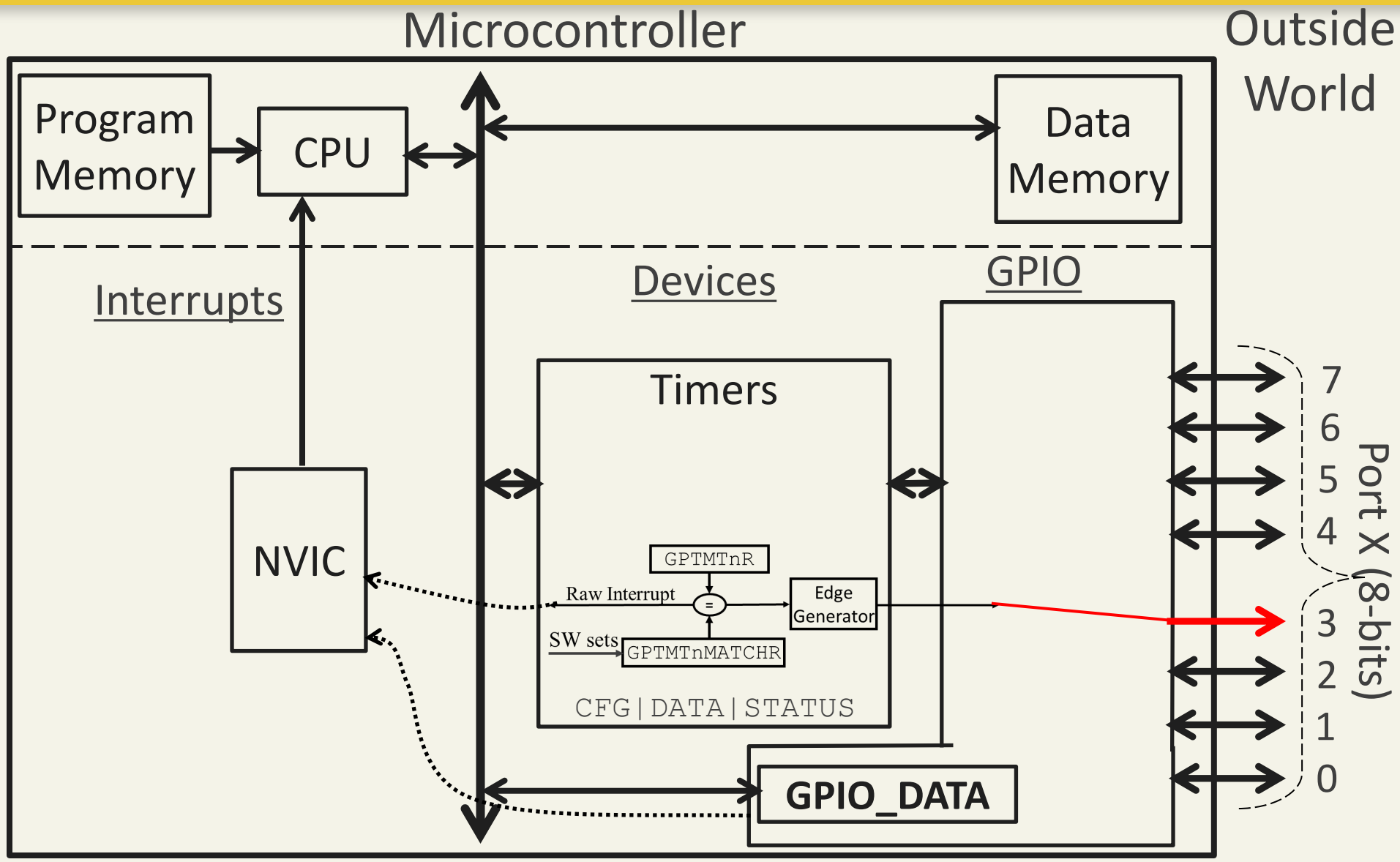
Microcontroller / System-on-Chip (SoC)



Microcontroller / System-on-Chip (SoC)



Microcontroller / System-on-Chip (SoC)



Output Compare: Design Principle

Time is important!

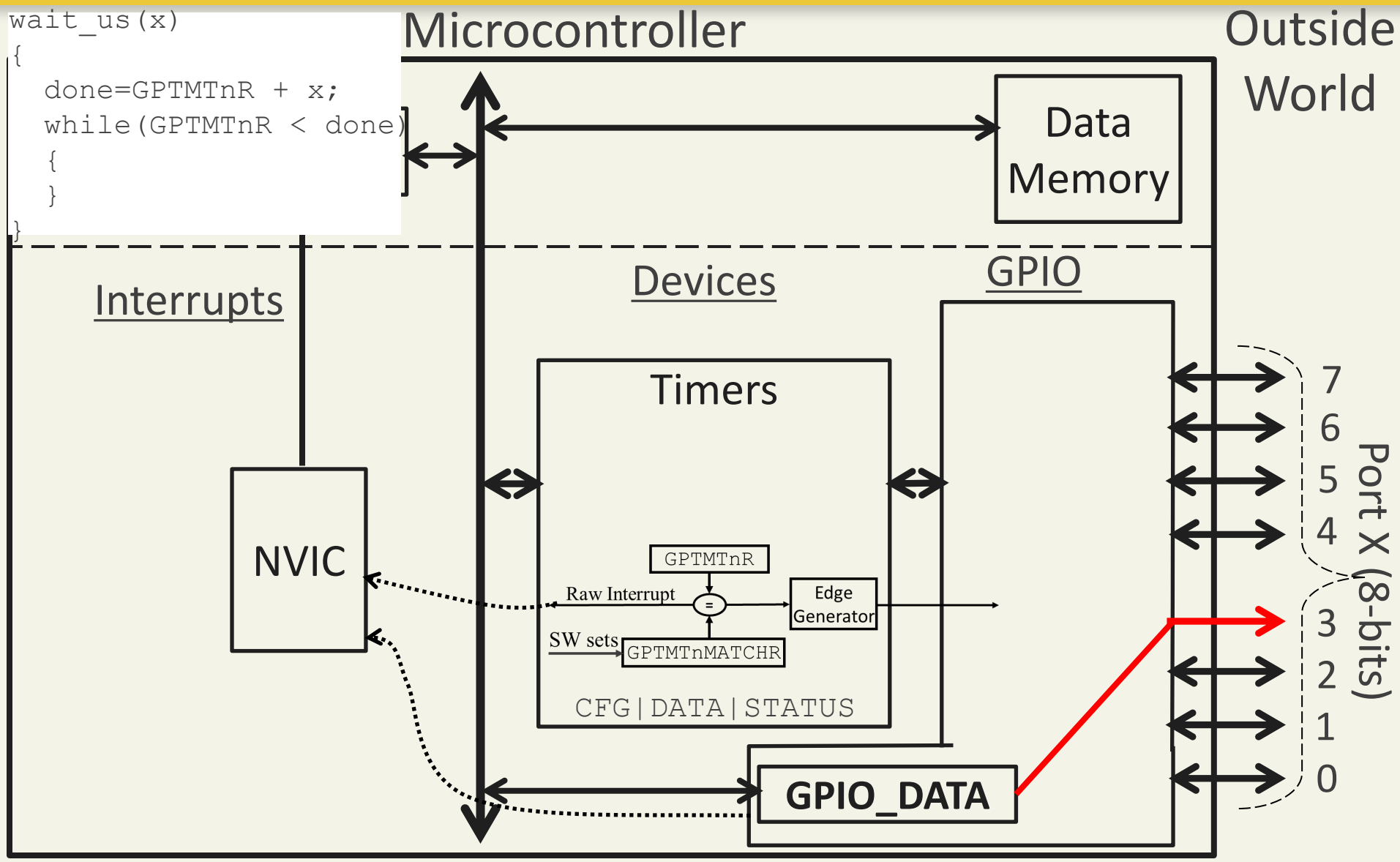
Microcontroller approaches for generate events at precise time intervals?

- Use time delay functions?

- Use interrupts?
  - May be accurate enough depending on application needs

Microcontroller / System-on-Chip (SoC)

```
wait_us(x)
{
  done=GPTMTnR + x;
  while(GPTMTnR < done)
  {
  }
}
```



Output Compare: Design Principle

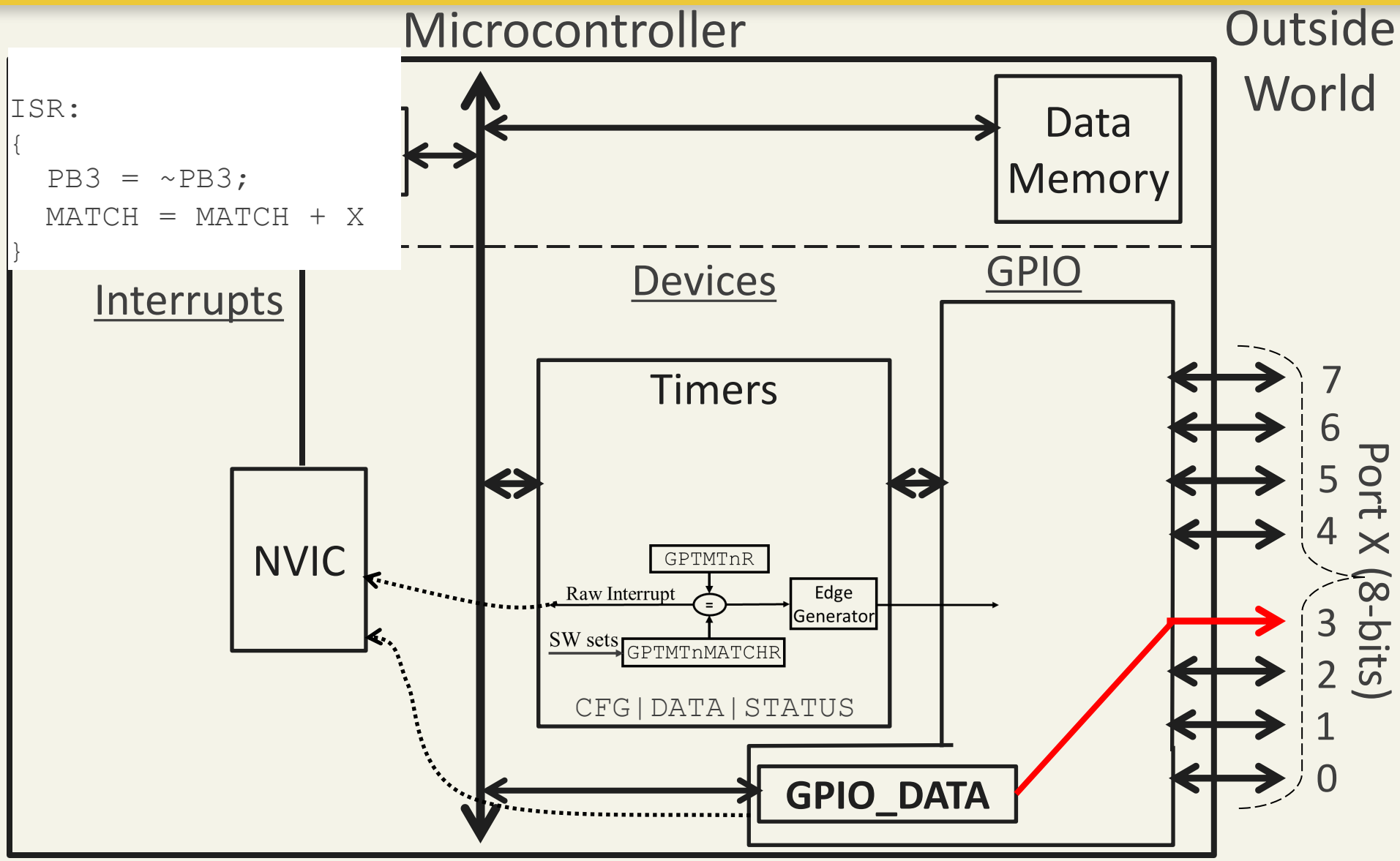
Time is important!

Microcontroller approaches for generate events at precise time intervals?

- Use time delay functions?
  - May be accurate enough depending on application needs
  - CPU cannot do anything else, interrupts may delay
- Use interrupts?

Microcontroller / System-on-Chip (SoC)

```
ISR:
{
   PB3 = ~PB3;
   MATCH = MATCH + X
}
```



Output Compare: Design Principle

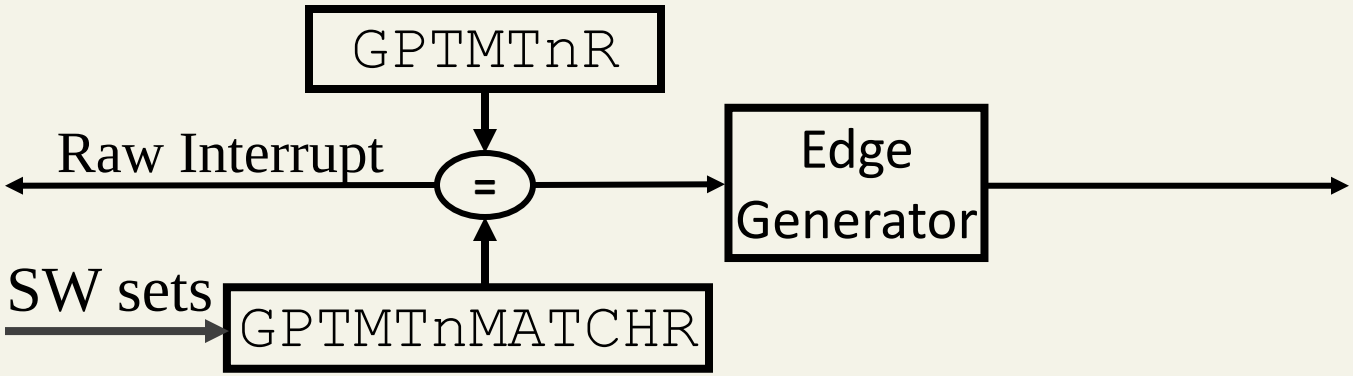
Time is important!

Microcontroller approaches for generate events at precise time intervals?

- Use time delay functions?
  - May be accurate enough depending on application needs
  - CPU cannot do anything else, interrupts may delay
- Use interrupts?
  - May be accurate enough depending on application needs
    - Potential sources of issues: i) interrupt overhead, ii) possible delays from other interrupts.

Output Compare: Design Principle

When the Timer equals the user-defined Match value the output compare unit can cause an action (e.g. interrupt, and/or output event).



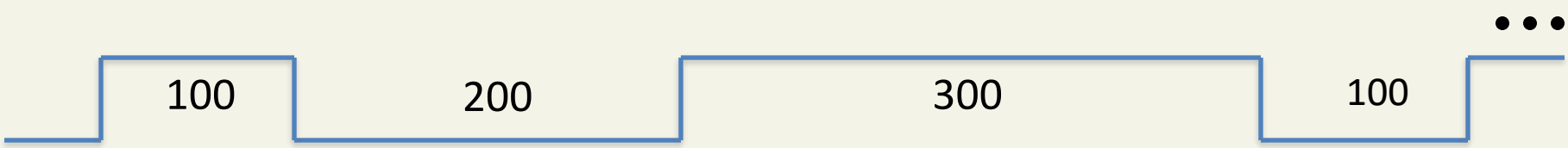
GPTMTnRn: Timer/Counter

GPTMTnMATCHR : Output Compare Match Register

Output Compare: General Purpose Waveform

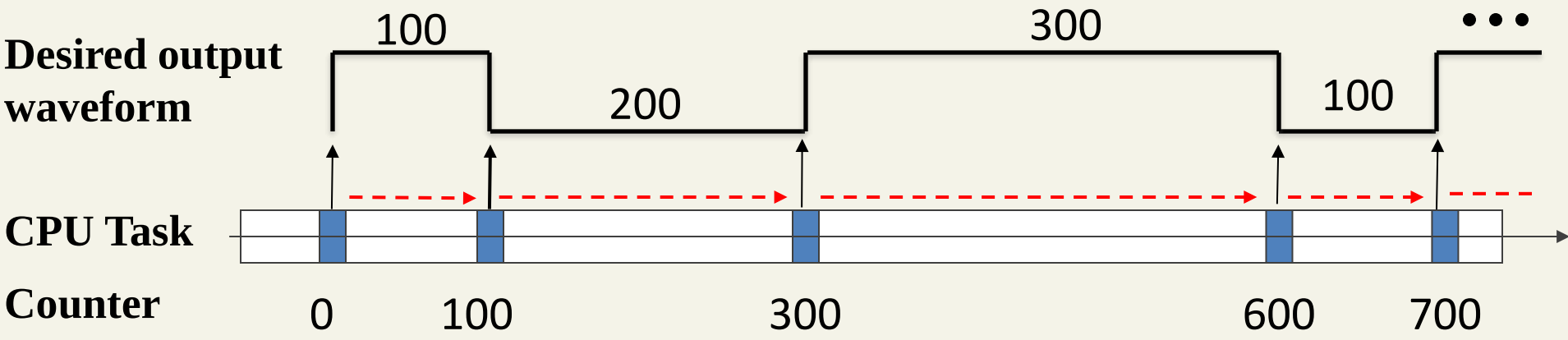
How to generate an output **waveform of arbitrary shape**?

Example: Generate a waveform that is high for 100 cycles, low for 200 cycles, high for 300 cycles, low for 100 cycles. Then repeats.



Output Compare: General Purpose Waveform

Approach: Preset the time of each event



CPU Foreground Task:

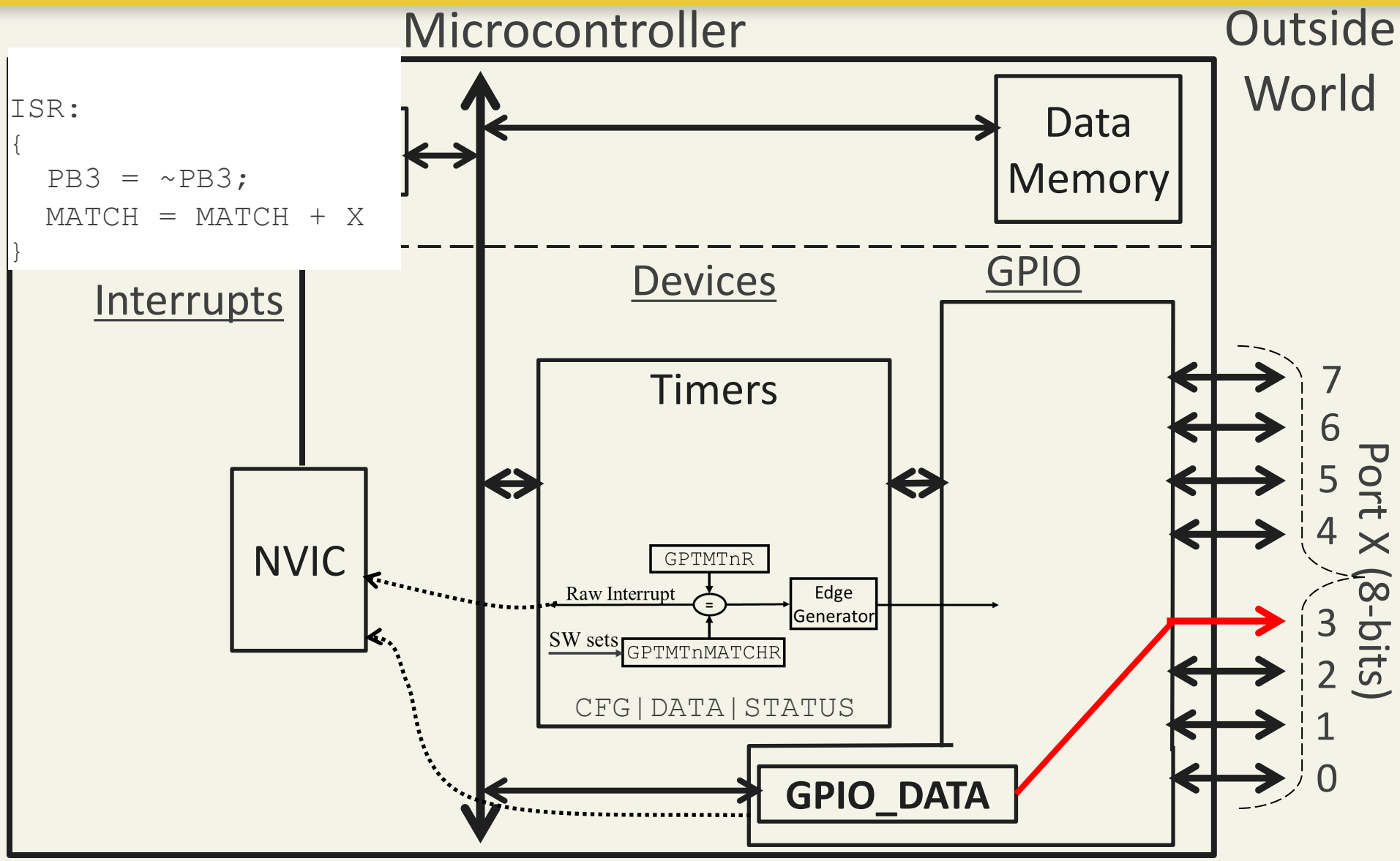
CPU Interrupt processing:

CPU toggle output: ⟶

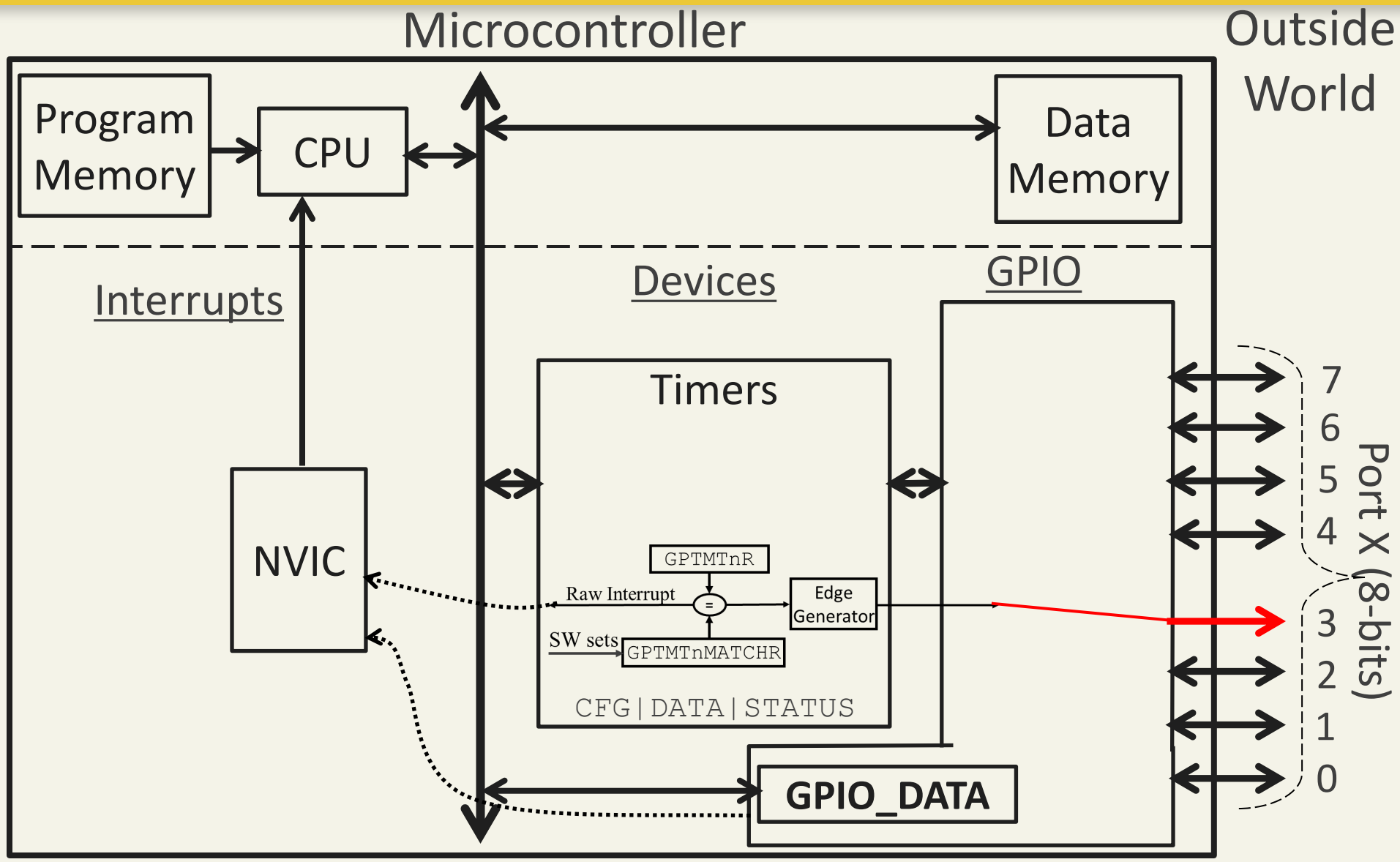
Set time for next match interrupt: - - ⟶

Microcontroller / System-on-Chip (SoC)

```
ISR:
{
   PB3 = ~PB3;
   MATCH = MATCH + X
}
```



Microcontroller / System-on-Chip (SoC)



TM4C123G 16/32-bit Timer/Counter

- Timer 16/32 bit
  - two 16bit timers (A & B) or single 32bit (A)
  - 6 Channels (0 – 5)
    - 11 modes
      - One shot
      - Periodic
      - Periodic Snapshot
      - Wait-for-Trigger
      - Real-Time Clock
      - Input Edge Count
      - Input Edge Time
      - PWM (one shot or periodic)
      - DMA
      - Synchronizing GP-Timer Blocks
      - Concatenated Modes

Servo Lab: Important Timer Modes

- **Periodic mode**
  - **Count down** to zero from a preset value, or **Count up** from zero to a value
    - Periodic means Timer resets to a preset value after reaching zero (Count down), or resets to zero after reaching a preset value (Count up)
    - One-shot means Timer stops when reaches 0 (Count down), or when reaches preset value (Count up)
  - Can set interrupt to fire when timer reaches end value and/or match value

This mode can generate generic waveforms at the cost of interrupt overhead

Servo Lab: Important Timer Modes

- **PWM mode (see datasheet table for details)**
  - 24 bit count-down counter
    - Prescale register used to increase size and is **not** used as prescale divisor
    - Timer size = 16-bit main counter + 8-bit extension using the prescale reg
  - Allows for creation of PWM waveform
  - Timer hardware handles switching output (off/on) for the developer (**no need for interrupts to generate the PWM waveform**)

Options for Generating a Waveform

- PWM waveform generation (**Used for Servo Lab**)
  - Place Timer in **PWM Mode:** the Output Compare (OC) hardware generates a PWM waveform without CPU involvement
  - can only easily generate a PWM waveform
  - no CPU overhead, since ISRs are not required
  - **See datasheet (1 pg.) for PWM operation mode details & example**
- Generic waveform generation
  - use Timer in **Periodic Mode**,
  - ISR sets the next event time in the MATCH, and the output value in the GPIO DATA register
  - can generate any arbitrary digital waveform
  - There is CPU overhead for executing the ISRs

PWM

Two parameters in PWM programming

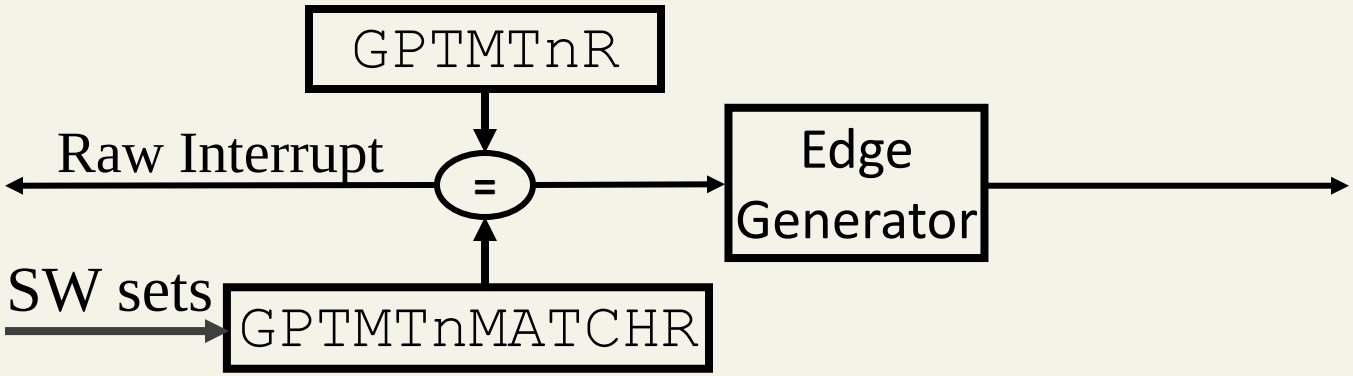
- Period: by writing a TOP value: lower 16 bits to the Interval Load Register (ILR), and the upper 8 bits to the Timer Prescale register.
- Pulse width: by writing to the Match Register

How does the Timer hardware work in PWM mode

- GPTMT*n*R (i.e., Counter) decrements every cycle
- First **event** occurs when GPTMT*n*R = GPTMT*n*MATCHR
- Second event occurs when GPTMT*n*R is reset (after it reaches 0 and is reset to TOP (i.e. ILR)).

Output Compare: Design Principle

When the Timer equals the user-defined Match value the output compare unit can cause an action (e.g. interrupt, and/or output event).



GPTMTnRn: Timer/Counter

GPTMTnMATCHR : Output Compare Match Register

Pulse Width Modulation = PWM

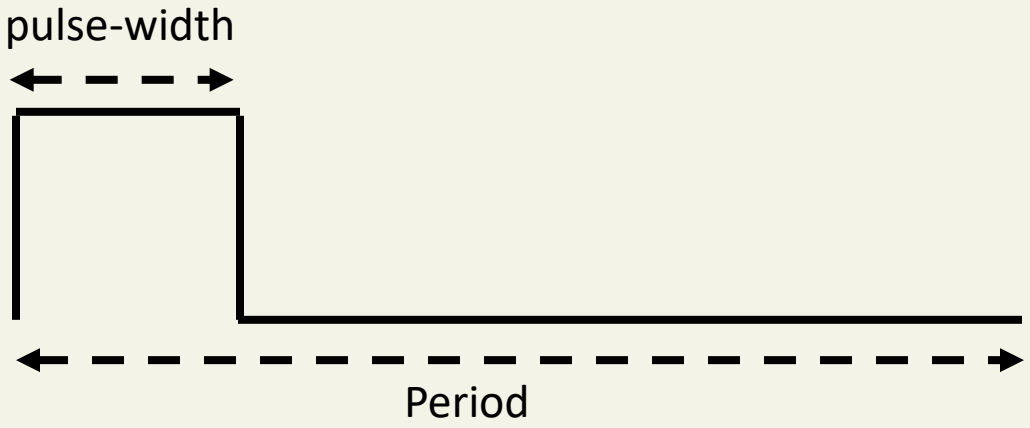
Parameters: Period and Pulse Width

Duty Cycle = Pulse width / Period

Programming: How to set the two parameters?

- Textbook: 9.2.3.7 PWM Mode: Steps 1-5

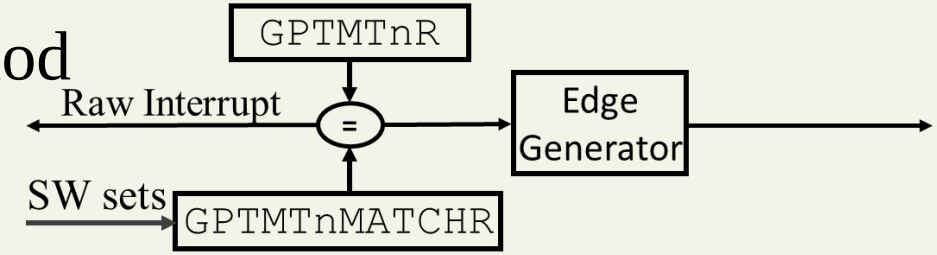
Servo Lab: PB5



Pulse Width Modulation = PWM

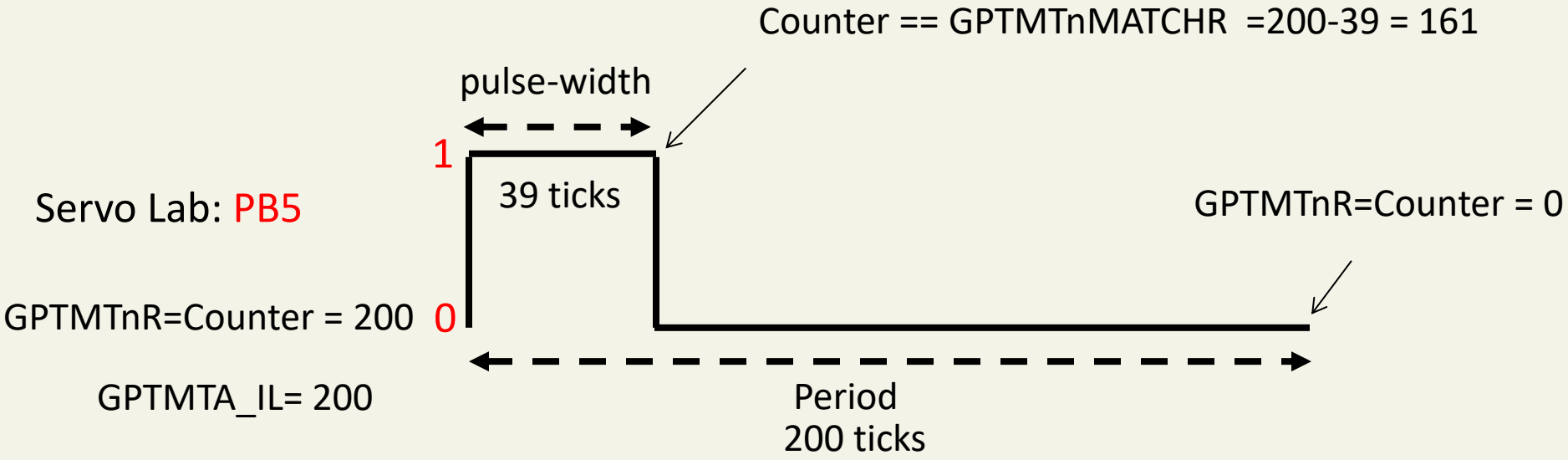
Parameters: Period and Pulse Width

Duty Cycle = Pulse width / Period



Programming: How to set the two parameters?

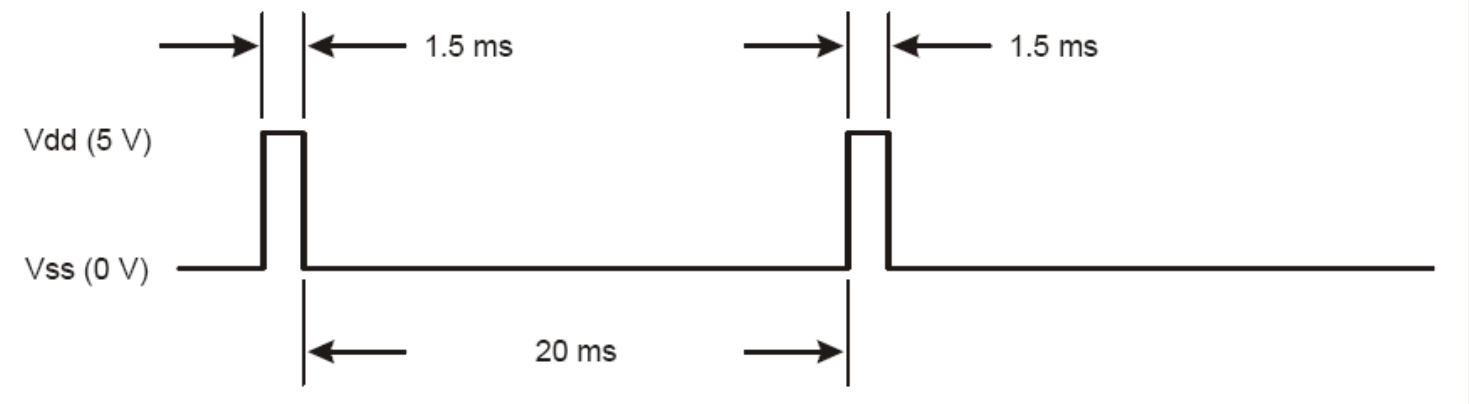
- Textbook: 9.2.3.7 PWM Mode: Steps 1-5



Servo Control

The Servo input: Periodic digital waveform of pulses

- The pulse width decides the target voltage (a property of the circuit inside the servo)
- The pulses must be separated by a time between 10ms and 40ms
- How does your program generate the waveform?



- This is a form of periodic PWM waveform: Pulse width = 1.5ms, pulse period = 21.5ms

Servo Control

In the Servo Lab, your program should be able to

1. Send pulses to the servo to make it move
2. Make the servo stop at the center position
3. Make the servo stop at different angles:

   0, 45, 90 (center), 135, 180

Servo Control

Use digital waveform to command servo of the target position

If the servo is **ideal**:

~1ms pulse – clockwise far end

~1.5ms pulse – center position

~2ms pulse – counterclockwise far end

10-40ms PWM period (does not have to be precise)

*The **actual servos requires calibration***

Servo Control

Programming tasks: Generate a periodic waveform with a certain pulse width and a fixed period

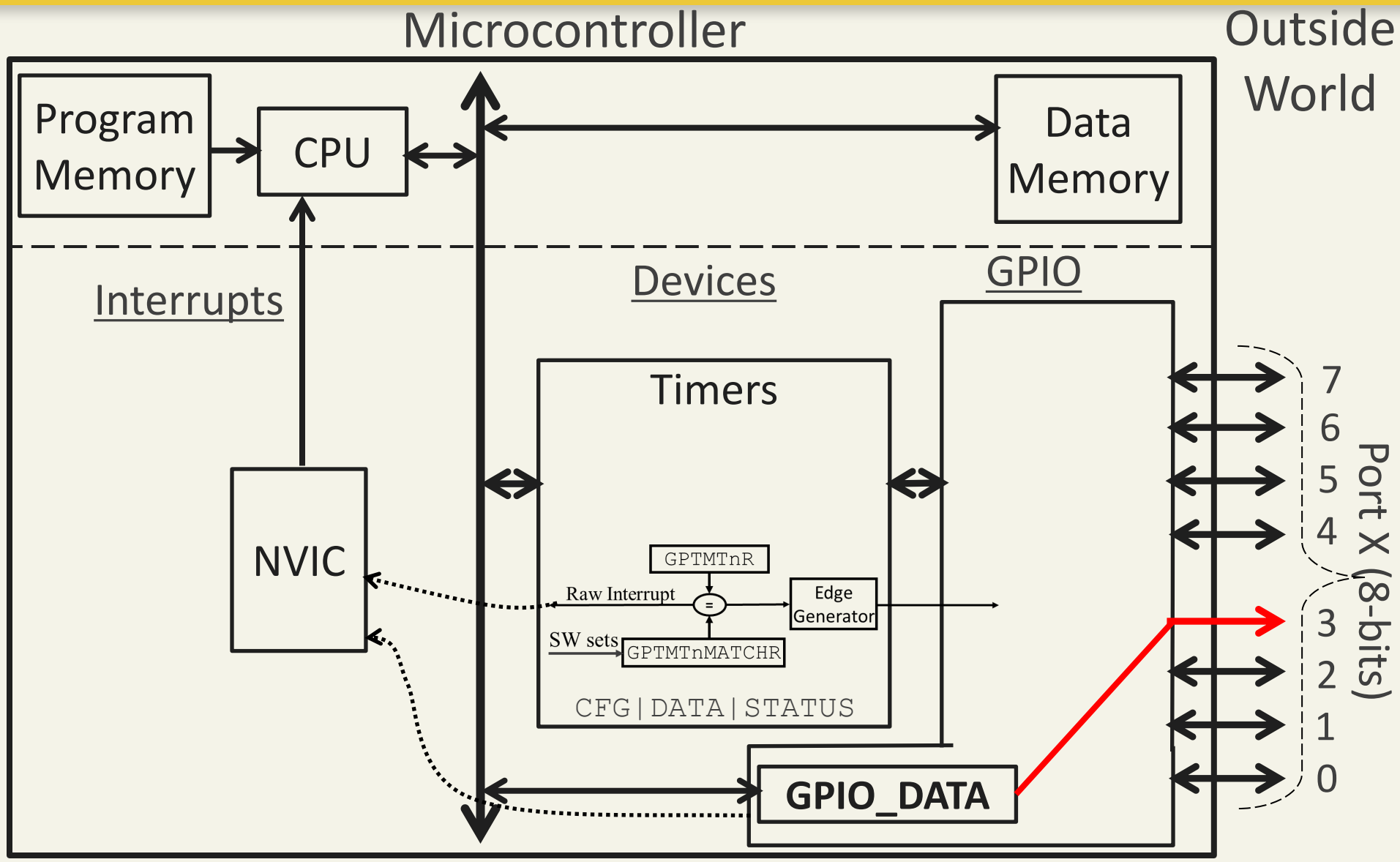
1.0~2.0ms corresponds to 0~180 degree counterclockwise

Suggested by servo's document. Again, ***calibration IS necessary***

Options for Generating a Waveform

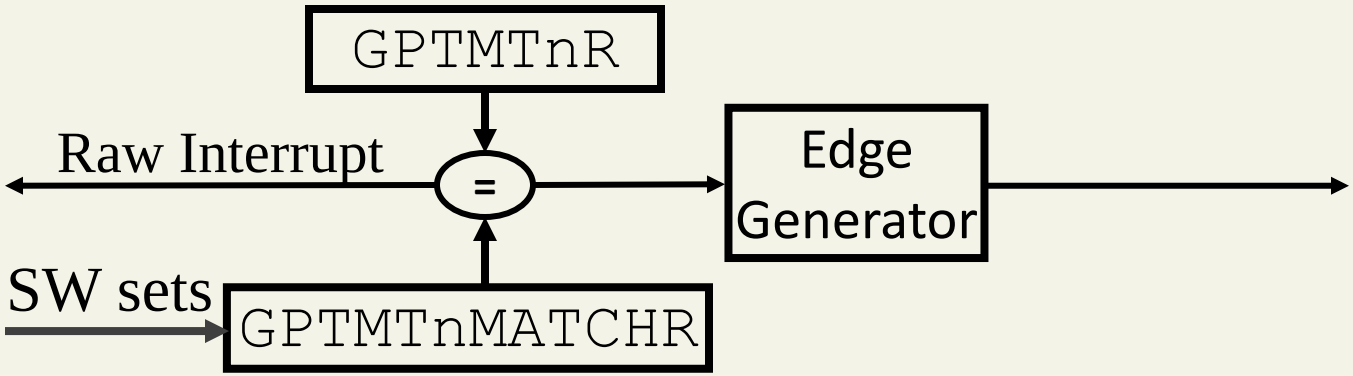
- PWM waveform generation (**Used for Servo Lab**)
  - Place Timer in **PWM Mode:** the Output Compare (OC) hardware generates a PWM waveform without CPU involvement
  - can only easily generate a PWM waveform
  - no CPU overhead, since ISRs are not required
- Generic waveform generation
  - use Timer in **Periodic Mode**,
  - ISR sets the next event time in the MATCH, and the output value in the GPIO DATA register
  - can generate any arbitrary digital waveform
  - There is CPU overhead for executing the ISRs

Microcontroller / System-on-Chip (SoC)



Output Compare: Design Principle

When the Timer equals the user-defined Match value the output compare unit can cause an action (e.g. interrupt, and/or output event).



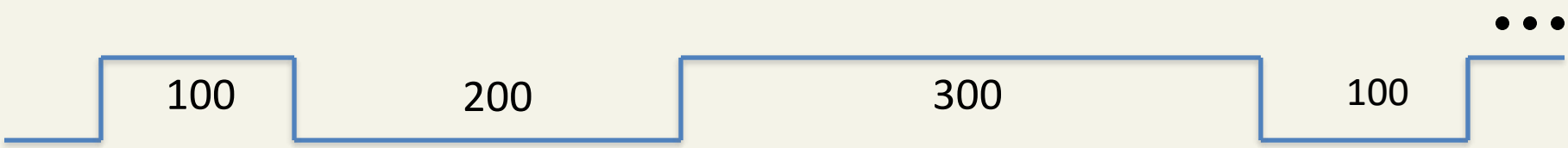
GPTMTnRn: Timer/Counter

GPTMTnMATCHR : Output Compare Match Register

Output Compare: General Purpose Waveform

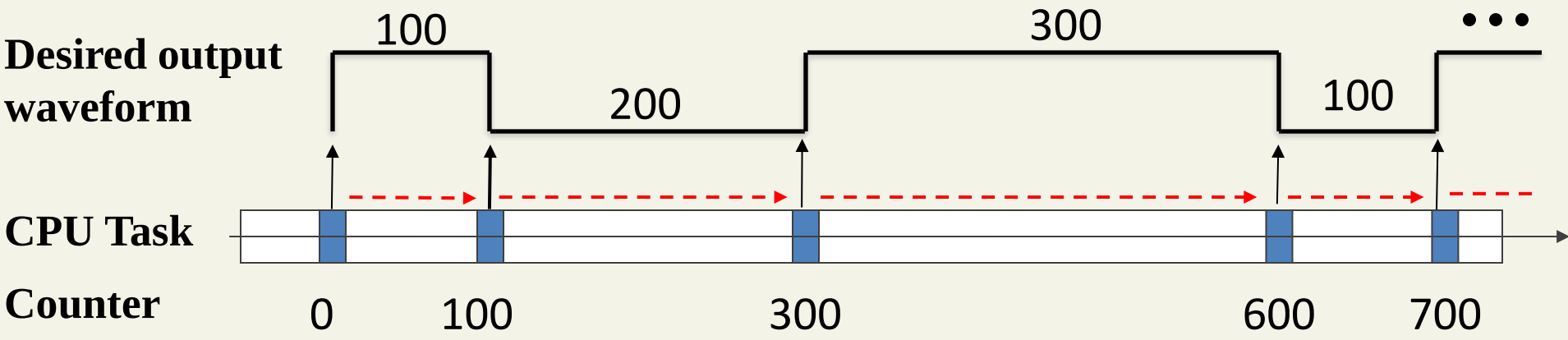
How to generate an output **waveform of arbitrary shape**?

Example: Generate a waveform that is high for 100 cycles, low for 200 cycles, high for 300 cycles, low for 100 cycles. Then repeats.



Output Compare: General Purpose Waveform

Approach: Preset the time of each event



CPU Foreground Task:

CPU Interrupt processing:

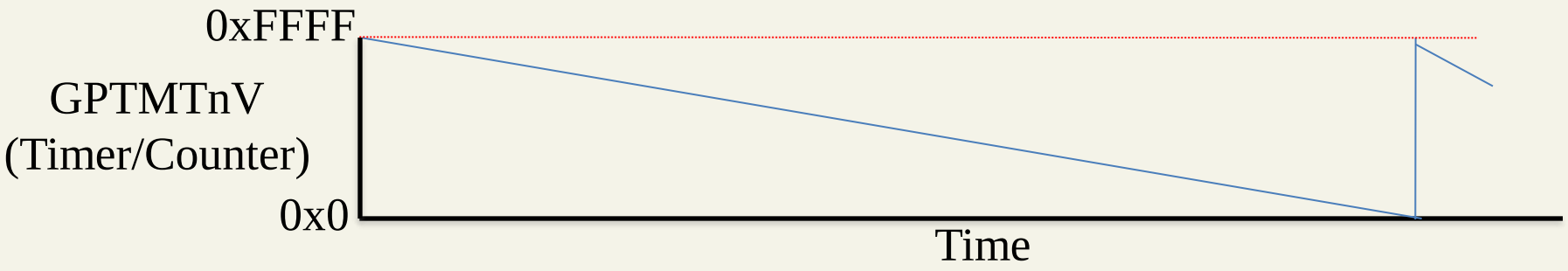
CPU toggle output: ⟶

Set time for next match interrupt: - - ⟶

General Purpose Waveform

Example: Use general purpose waveform generation to make a waveform that is high for M cycles, and low for M cycles

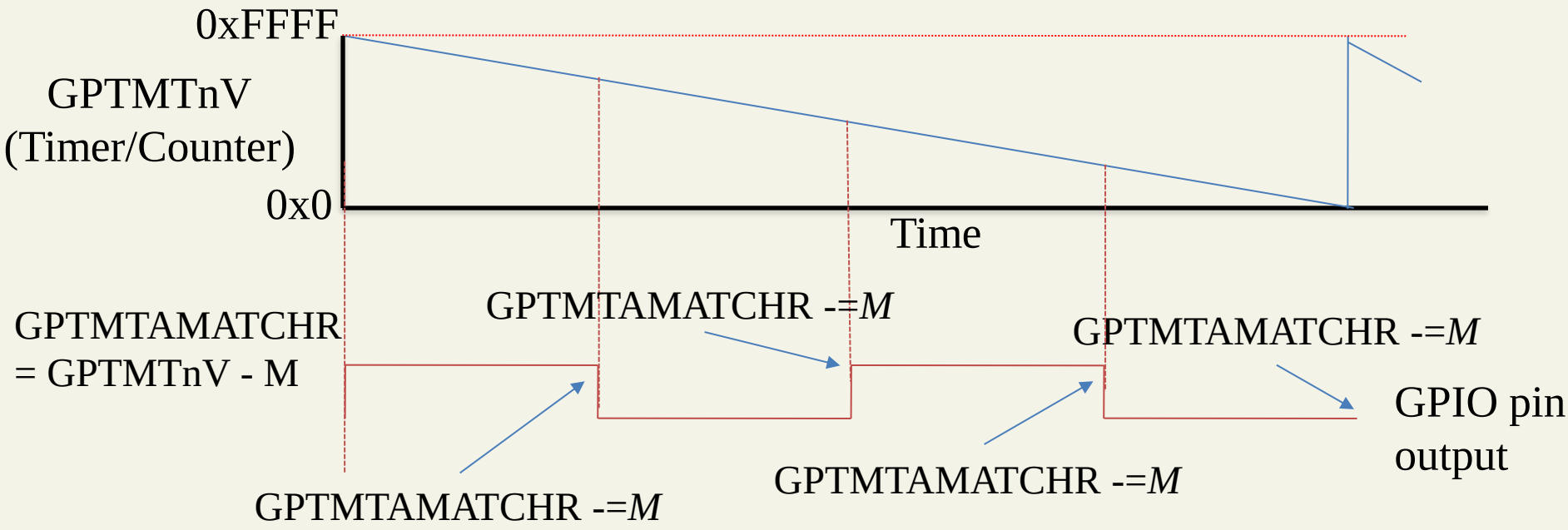
Waveform Period = $2 \times M$



General Purpose Waveform

Example: Use general purpose waveform generation to make a waveform that is high for M cycles, and low for M cycles

Waveform Period = $2 \times M$



Programming Example: General Purpose Waveform

Use Periodic Timer Mode to generate a 50% duty cycle waveform, with a Period of 2*M timer cycles, using Timer0A in count down mode. Assume the Timer, GPIO (PF0), and NVIC initialized already

```
TIMER0A_Handler(void)
{
     // 1) Check that a Match interrupt occurred
   if(TIMER0_MIS_R & TIMER_MIS_TAMMIS)
   {
     // 2) Clear interrupt flag
     TIMER0_ICR = TIMER0_ICR | TIMER_MIS_TAMMIS;

     // 3) Set next match time
      TIMER0_TAMATCHR_R = TIMER0_TAMATCHR_R - M;

     // 4) Toggle output wire
     if(GPIO_PORTF_DATA_R & 0x01)
     {
        GPIO_PORTF_DATA_R &= ~0x01;//set low
      }
     else
     {
        GPIO_PORTF_DATA_R |= 0x01;//set high
      }
   }
}
```

Programming Example: General Purpose Waveform

Use Periodic Timer Mode to generate a 50% duty cycle waveform, with a Period of 2*M timer cycles, using Timer0A in count down mode. Assume the Timer, GPIO (PF0), and NVIC initialized already

```
TIMER0A_Handler(void)
{
     // 1) Check that a Match interrupt occurred
   if(TIMER0_MIS_R & TIMER_MIS_TAMMIS)
   {
     // 2) Clear interrupt flag
     TIMER0_ICR = TIMER0_ICR | TIMER_MIS_TAMMIS;

     // 3) Set next match time
     TIMER0_TAMATCHR_R = TIMER0_TAMATCHR_R - M;

     // 4) Toggle output wire
     GPIO_PORTF_DATA_R = GPIO_PORTF_DATA_R ^ 0x01;




   }
}
```

Programming Example: General Purpose Waveform

Generate a periodic waveform repeating the following:

100-cycle low, 100 high, 200 low, 200 high, 300 low, 300 high.

- Assume: 1) Timer already configured in count-down periodic mode, 2) Assumer Port F wire 0 will be the output and is already properly configured, 3) Assume MATCH interrupts have already been enabled, and the NVIC has been configured.

- **Give code to place in the Timer ISR**

Programming Example: General Purpose Waveform

Use Periodic Timer Mode to generate the specified waveform. Using Timer0A in count down mode. Assume the Timer, GPIO(PF0), and NVIC initialized already.


```
volatile unsigned int count[6]={100, 100, 200,200, 300, 300};
int pos = 0;

//Assume output is initially high
TIMER0A_Handler(void)
{
    // 1) Check that a Match interrupt occurred
   if(TIMER0_MIS_R & TIMER_MIS_TAMMIS)
   {
     // 2) Clear interrupt flag
     TIMER0_ICR = TIMER0_ICR | TIMER_MIS_TAMMIS;

     // 3) Set next match time
     TIMER0_TAMATCHR_R = TIMER0_TAMATCHR_R - count[pos];
     pos =(pos+1) % 6;

     // 4) Toggle output wire
     GPIO_PORTF_DATA_R = GPIO_PORTF_DATA_R ^ 0x01;
   }
}
```

Programing Example: General Purpose Waveform

Initialize Timer/Counter 0A's OC unit as periodic for general purpose waveform gen

```
timer_init(){
//init GPIO, and enable Timer clock, and count-down
...
TIMER0_CTL_R &= ~TIMER_CTL_TAEN; //disable timer0A
TIMER0_CFG_R |= TIMER_CFG_16_BIT; //set to 16bit
TIMER0_TAMR_R = TIMER_TAMR_PERIOD; //set to periodic
TIMER0_TAPR_R = 0; //set timer prescaler
TIMER0_TAILR_R = 0xFFFF; //set period
TIMER0_TAPMR_R = 0; // set match prescaler
TIMER0_TAMATCHR_R = first_match; // set value for initial intpt
TIMER0_ICR_R |= TIMER_IMR_TAMIM; //clear interrupts
TIMER0_IMR_R |= TIMER_IMR_TAMIM; //enable match interrupts
IntRegister(INT_TIMER0A, TIMER0A_Handler); //Bind intrupt handle

// NVIC setup
...
IntMasterEnable(); //enable global interrupts
TIMER0_CTL_R |= TIMER_CTL_TAEN; //enable timer0A
}
```

Review of OC Programming Interface

- GPTMCTL – Control
- GPTMCFG – Configuration
- GPTMTnMR – Timer n mode

- GPTMTnPR – Timer n prescale / 8 bits PWM
- GPTMTnILR – Timer n interval load
- GPTMTnPMR – Timer n prescale match
- GPTMTnMATCHR – Timer n match

- GPTMIMR – Interrupt mask
- GPTMRIS – Raw interrupt status
- GPTMICR – Interrupt clear

See page 726 of data sheet for more info

Summary of OC General Purpose Waveform

Good for generating **waveforms of any shape**

Programming: Use Interrupt to pre-set the timing of the next event

Cons:

- Interrupt overhead can be high
- Cannot generate high-frequency waveforms
- CPU cannot sleep into deep power-saving modes

Summary of PWM

Good for generating **Pulse Width Modulation waveforms** and **Clock waveforms**

Two parameters: **Pulse Width** and **Period Length**

(they decide the timing of two events)

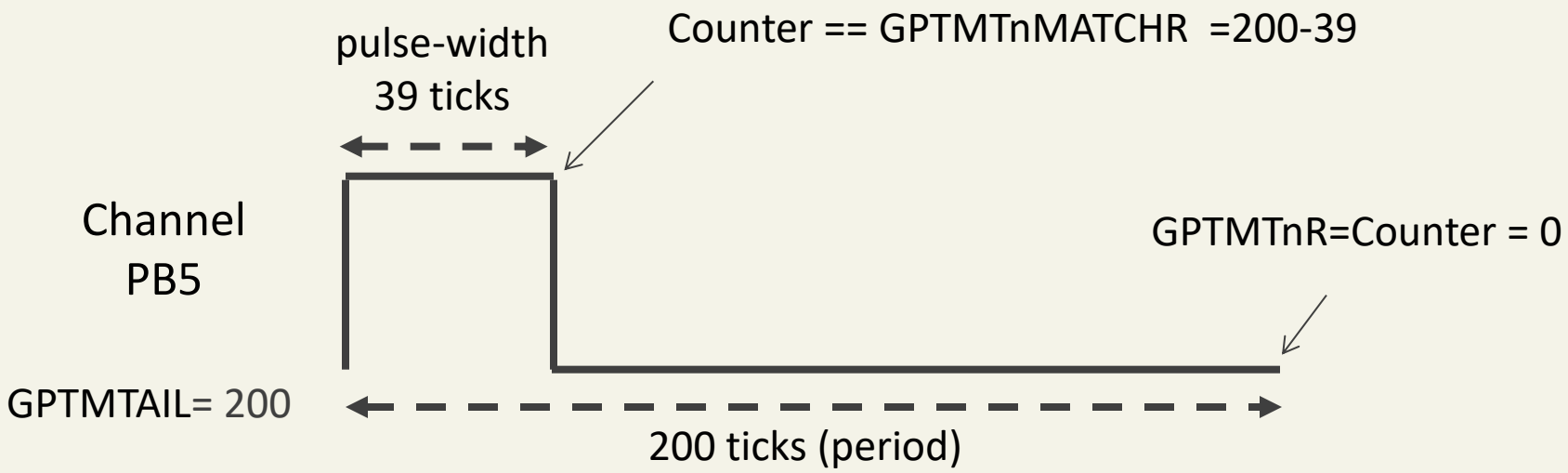
Programming

- Lower 16bits (15:0) go in the Timers Interval Load register
- Higher 8bits (23:16) go in the Timer Prescale Register
- Match stores (Top - Pulse_Width) for down counter mode

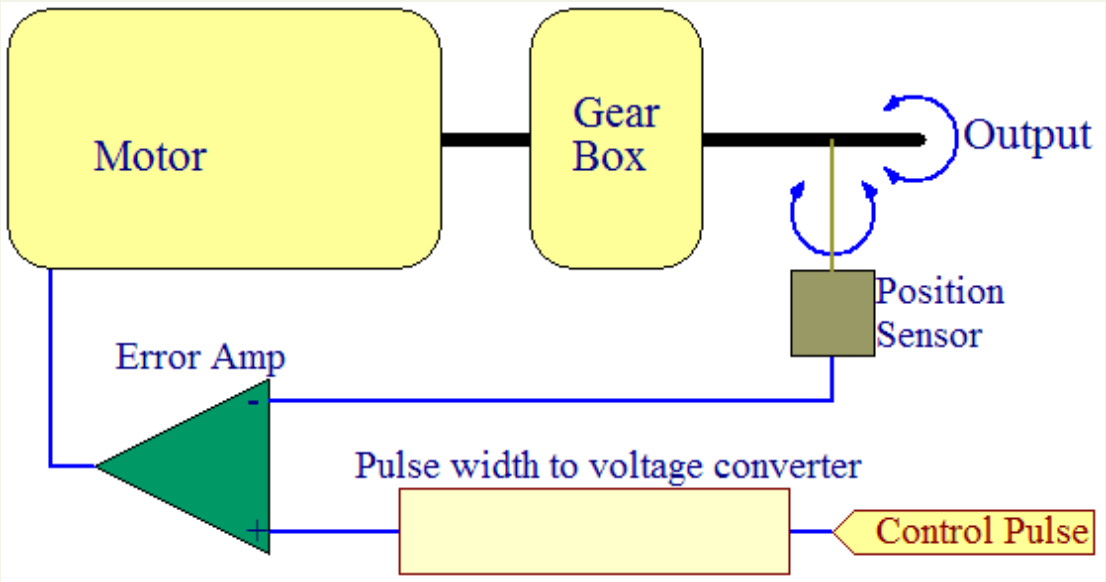
PWM: Summary example

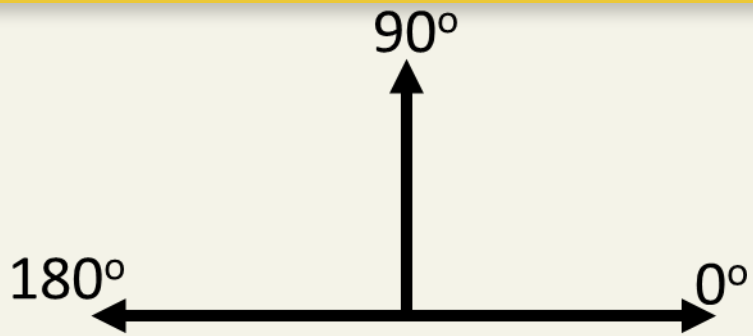
Objectives:

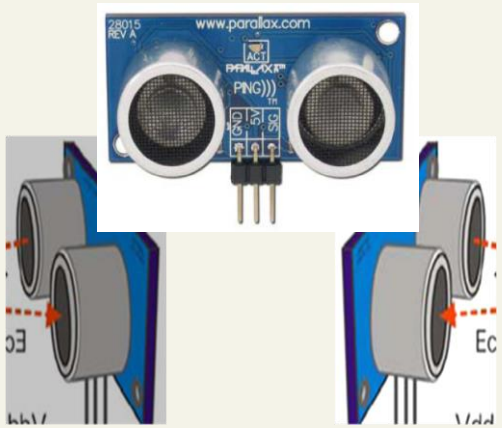
- Generate a PWM wave that has a pulse width of 39 ticks and a period of 200 ticks
- The PWM wave should be generated on PB5 in lab



Servo Control







Control feedback loop:

1) A control pulse is converted to a target voltage
2) If the servo is not at the **target angular position**, there will be an **error** between the target voltage and the position sensor voltage
3) The voltage error is amplified and used to turn motor in the direction that reduces the voltage error toward zero