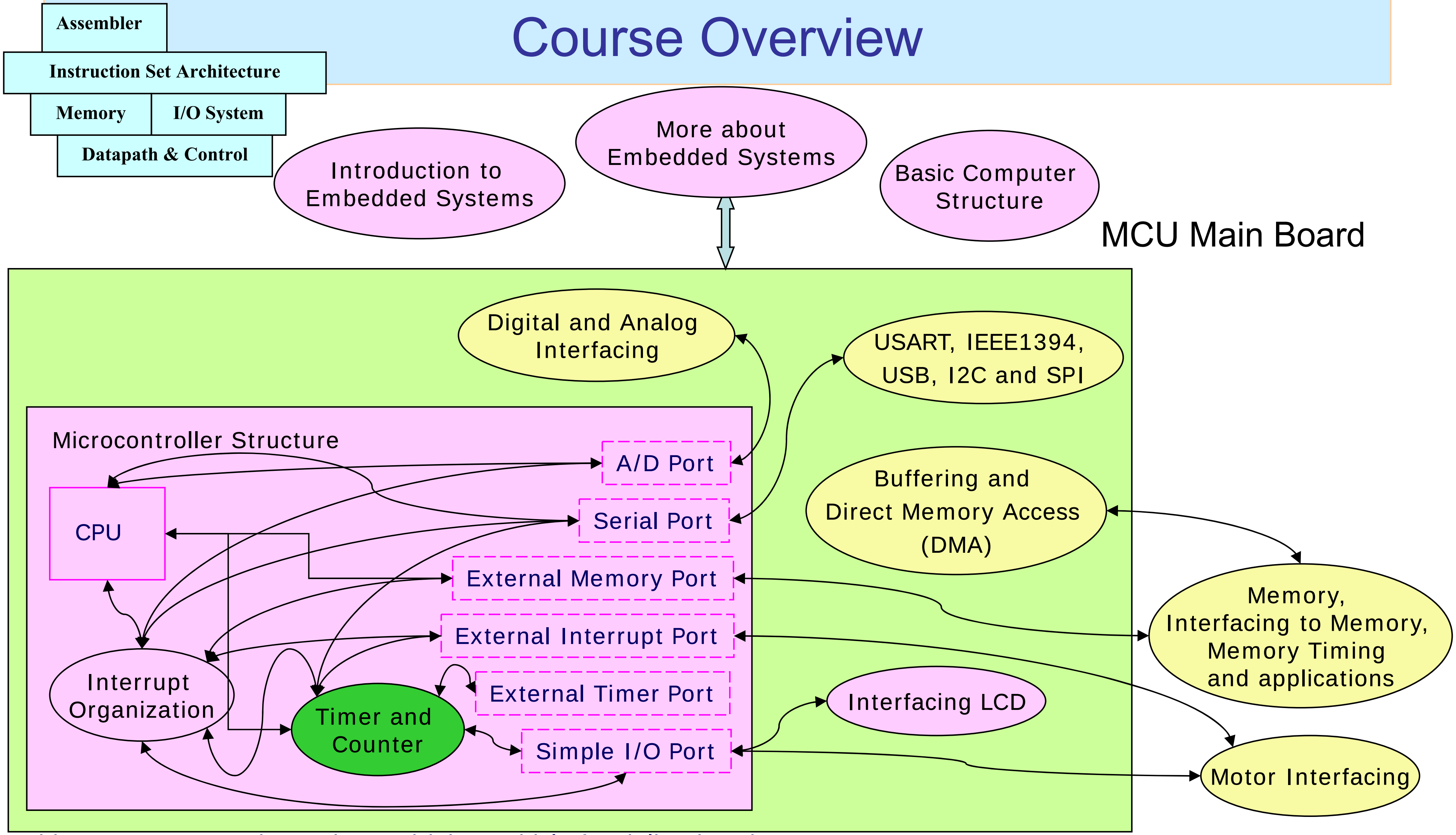
ELEC 3300
Introduction to Embedded Systems

Topic 7

Timer and Counter

Prof. Tim WOO

Course Overview



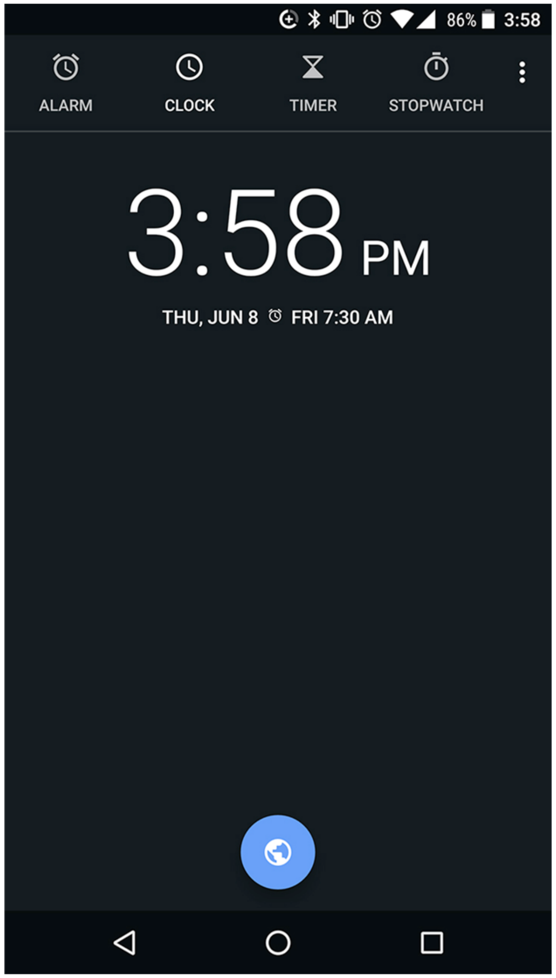
In this course, STM32 is used as a driving vehicle for delivering the concepts.

To be covered	In progress	Done

Expected Outcomes

- On successful completion of this topic, you will be able to
  - Summarize the importance of the timer in microprocessor
  - Distinguish the features of timer and counter
  - Demonstrate some applications of timers
  - Understand the timing diagram of the general-purpose timers in STM32.

Why TIMERS?



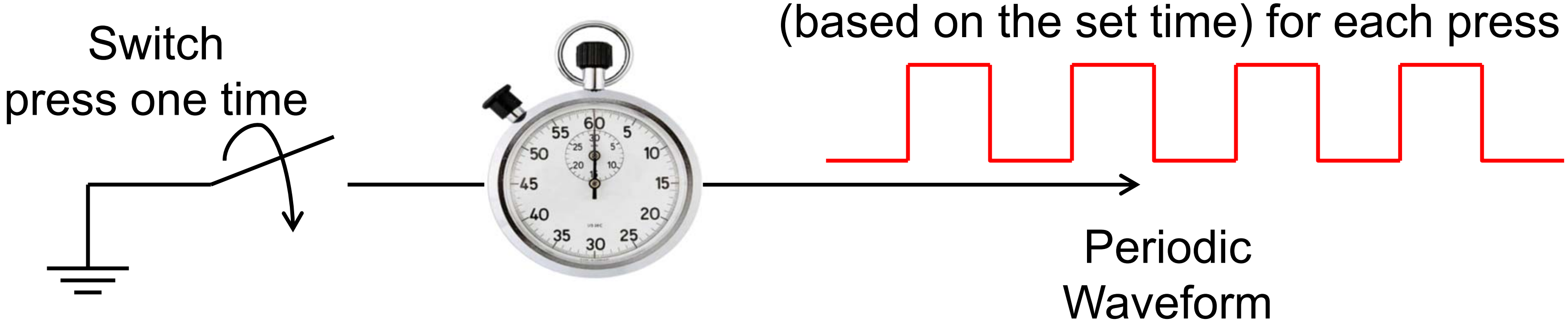
- CLOCK – display real time in multiple time zones
- ALARM – alarm at certain (later) time(s)
- STOPWATCH – measure elapsed time of an event
- TIMER – count down time and notify when count becomes zero (introduce a delay, count an event)

Timer-mode and counter-mode operation

- **Two modes of operation: TIMER and COUNTER modes**

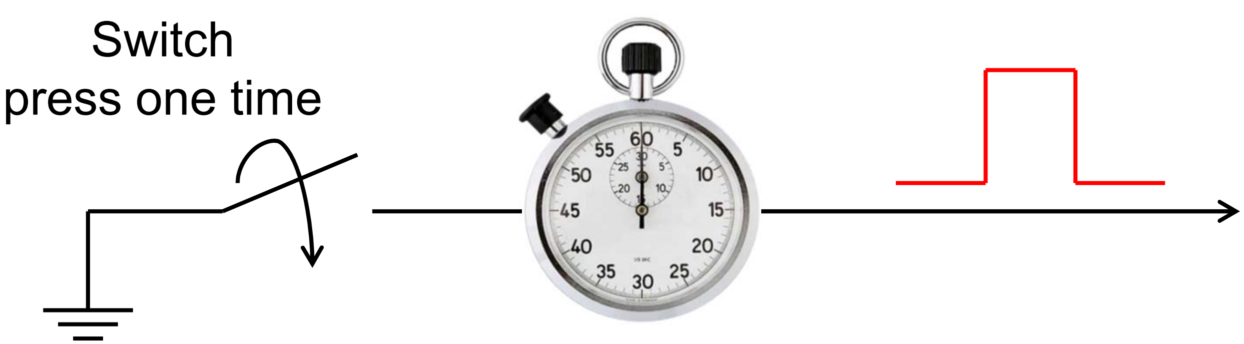
Start as timer:

Generates predefined number of pulses (based on the set time) for each press



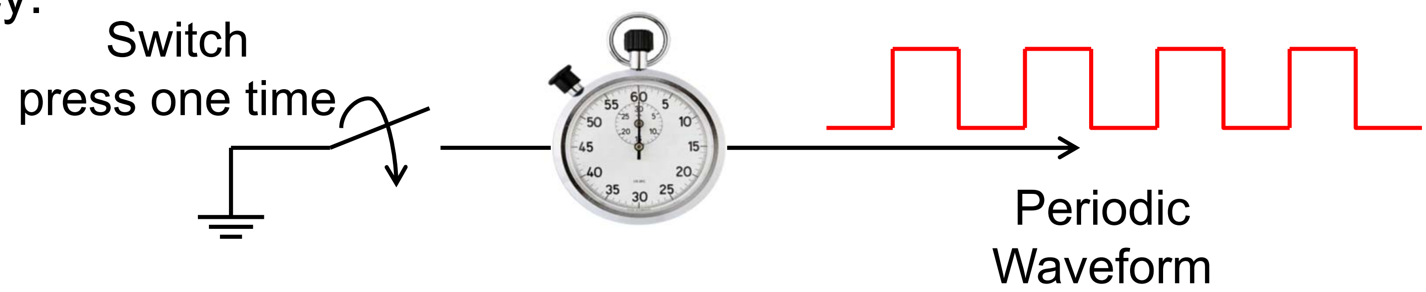
Start as counter:

Generates one pulse for each press



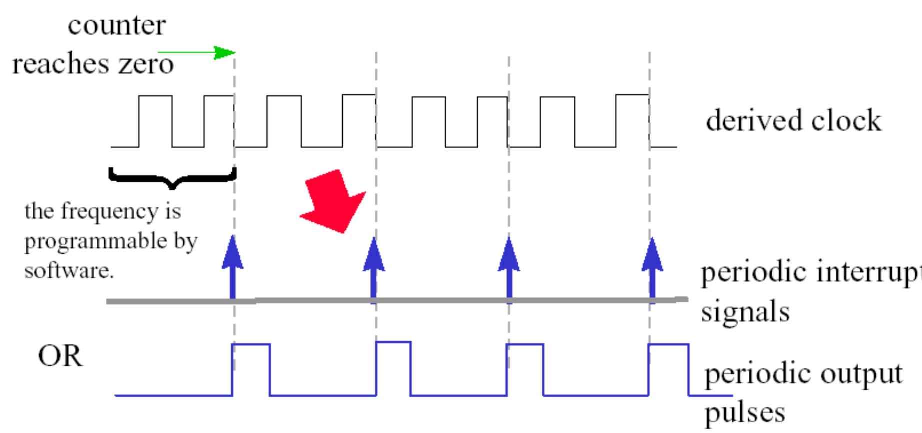
Timer-Mode Operation

- An output pulse generated each time the counter reaches zero (or preset value) provides a signal with a software-programmable frequency.



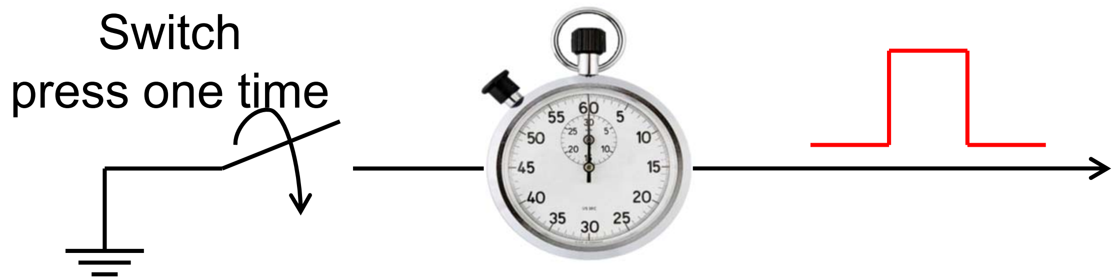
- Applications:
  - A common use for this output is as a baud rate generator for a serial communication interface (different CLK frequencies for different baud rates).
  - generating waveforms for audible tones.

    (Try this https://www.szynalski.com/tone-generator/)
  - Timing the length of event

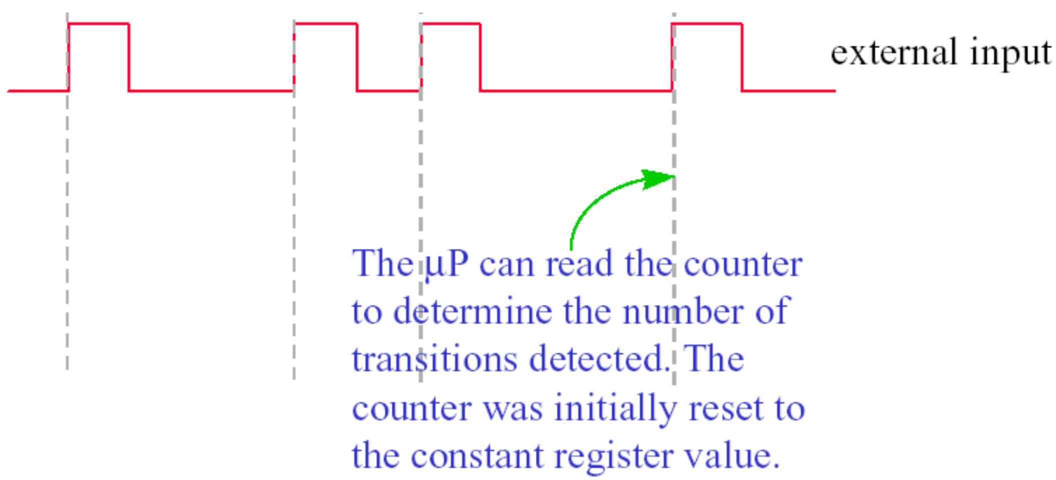


Counter-Mode Operation

- It is useful for interfacing to other devices that produce streams of possibly non-period pulses that must be counted。

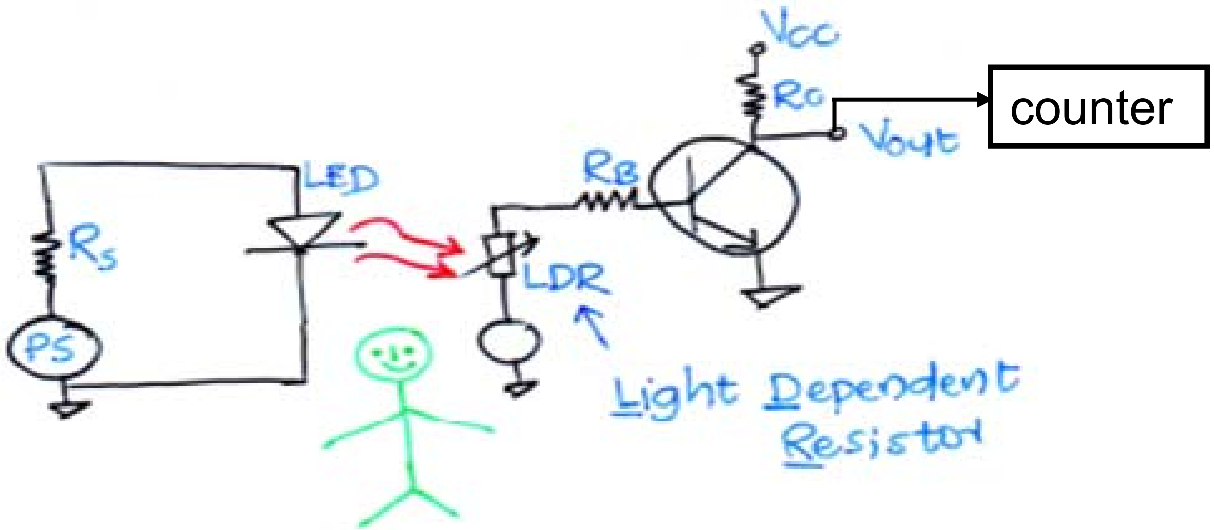


- Applications:
  - Count the number of times that a user has pressed a key.



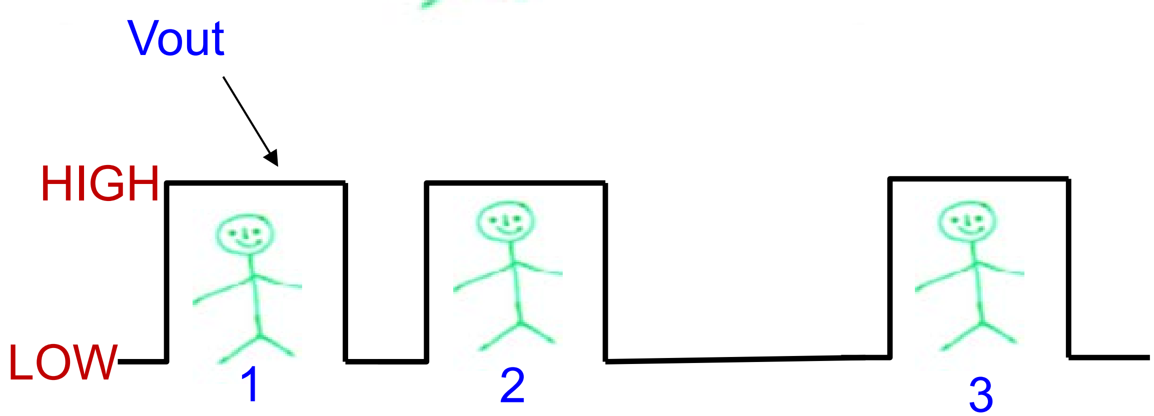
Real-world Examples of Counter-mode operation

Counting People (customers of 7-11 store)



Condition #1 - No person between LED and LDR:

LDR turns ON while receiving light from LED → sufficient base current → large collector current → transistor ON (saturation) → Vout is LOW.



Condition #2 - Person between LED and LDR:

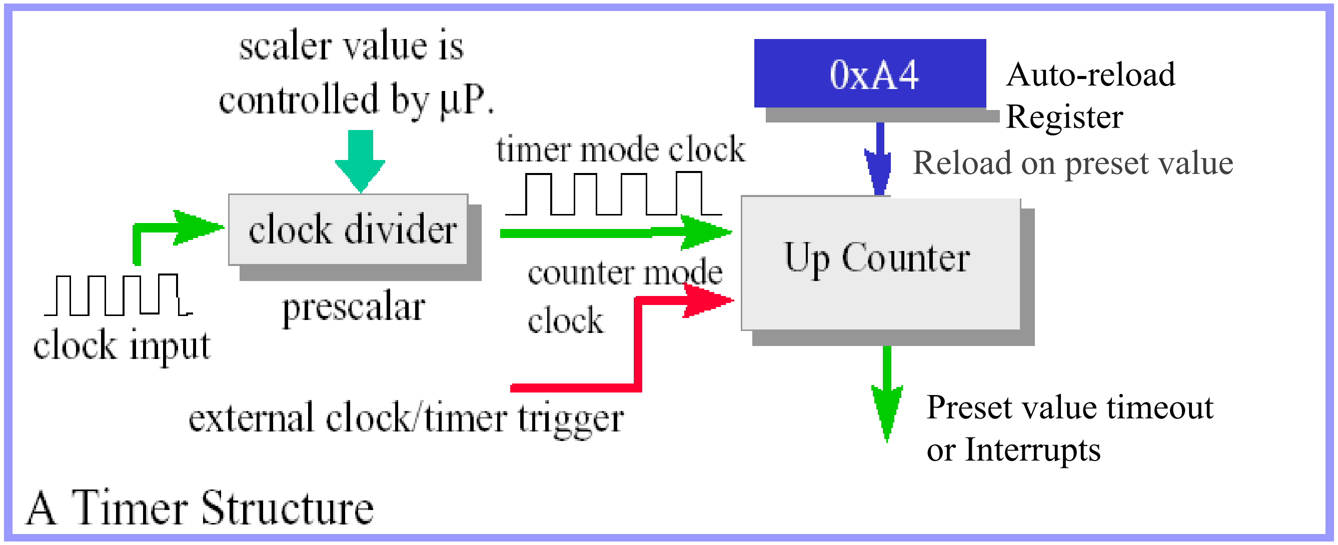
LDR turns OFF as no light from LED → No base current → No collector current → transistor OFF → Vout is HIGH.

Counter value = number of pulses = number of people

Timer Structure

- Devices that use a high speed clock input to provide a series of time or count related events
- Building block of a timer

Auto-reload Register: Defines the count period



- Other important registers:
  - Counter Register, Mode Register, Control Register, Status Register

Counter Register: Stores and updates the current count number (Implementation)

Mode Register: Timer / Counter mode (Initialization)

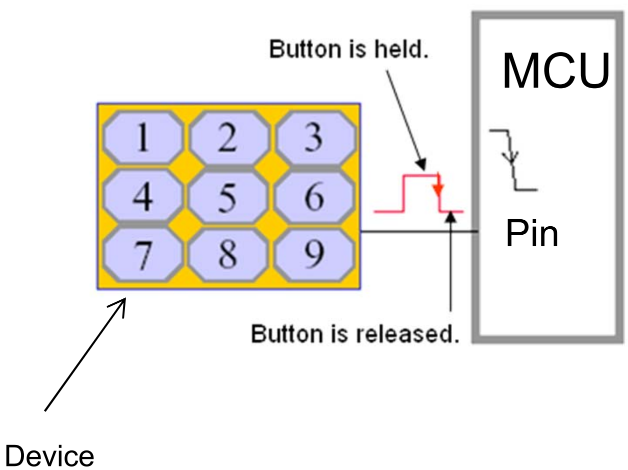
Control Register: Timing/Counting Start/Stop control (Implementation)

Status Register: Current status of Timer/Counter, i.e., running / completed. Usually triggers a follow-up action like interrupt generation (Implementation)

Example 1 – Event Counting

- Say, you want a microprocessor to know how many times a user has pressed a key.
- We can connect the keypad to the microprocessor pin line, and then use Timer X to detect the key stroke which sends a pulse to pin.

Description
Abstract idea of project (Define the functionality of the system)
Data format / representation
Programming Language
Communication Protocol
Physical connection (Pins assignment)
Hardware devices (Microcontroller, Peripherals)



Pseudo code:

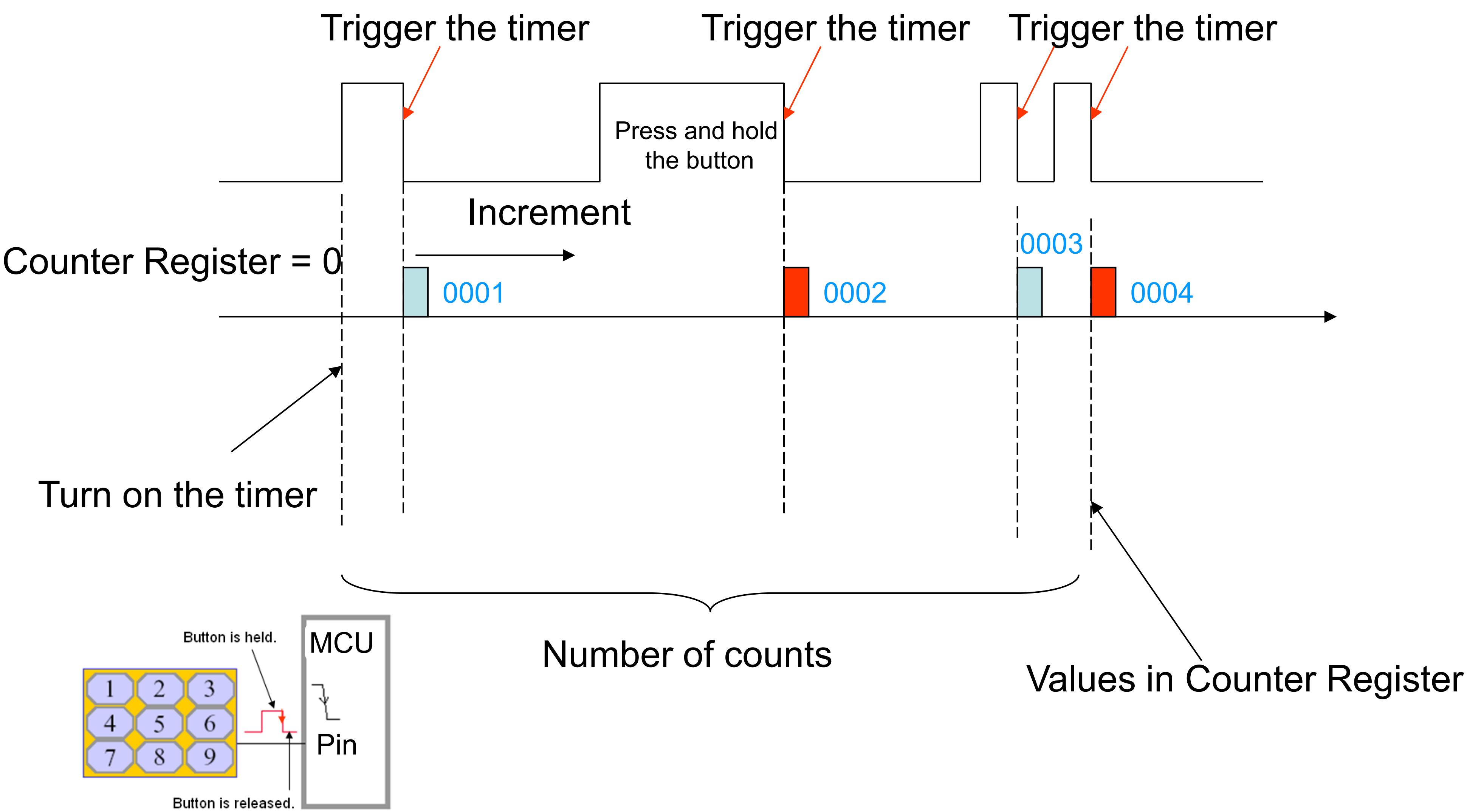
Timer initialization
Turn on Timer X
……
……
while (the button is pressed)
    Event counting

Read counter register

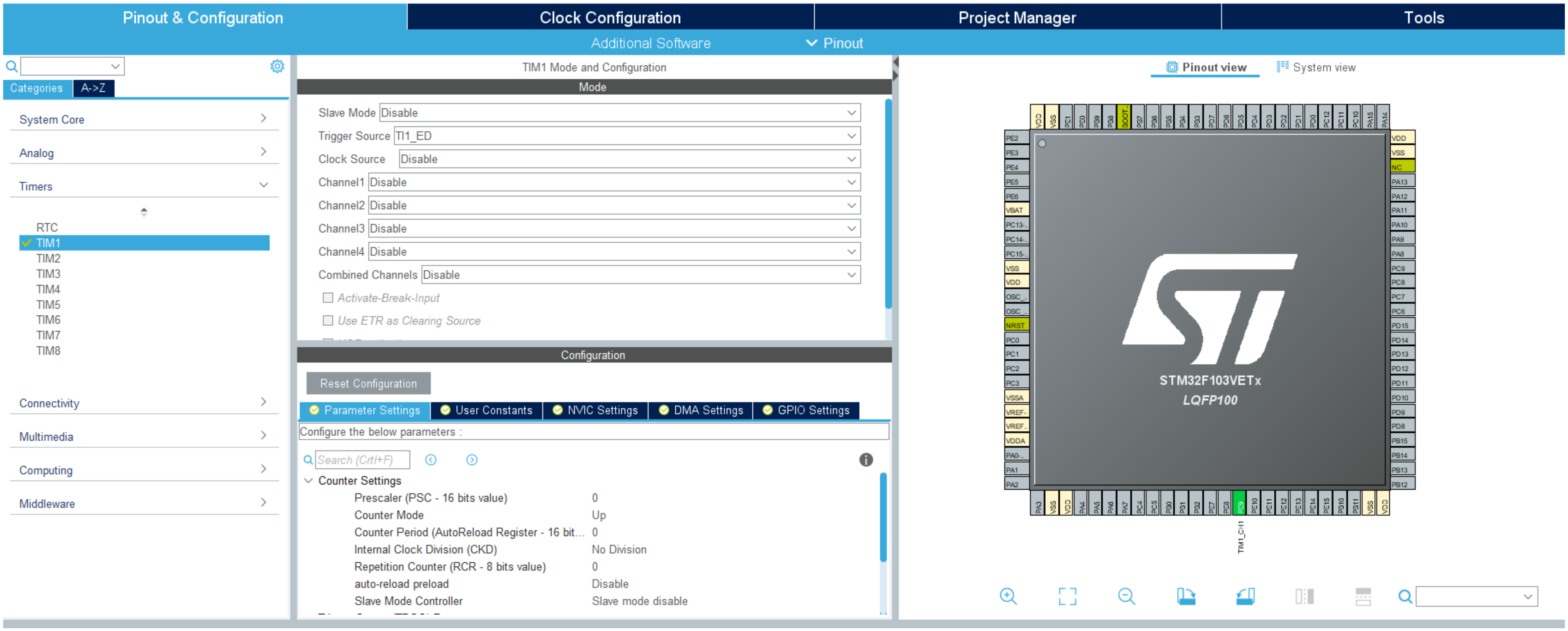
Step 1: Initialization
Configure the type of ~~Timer~~ / Counter

Step 2: Implementation
Run / Stop the timer,
Read the counter register

Example 1 – Event Counting

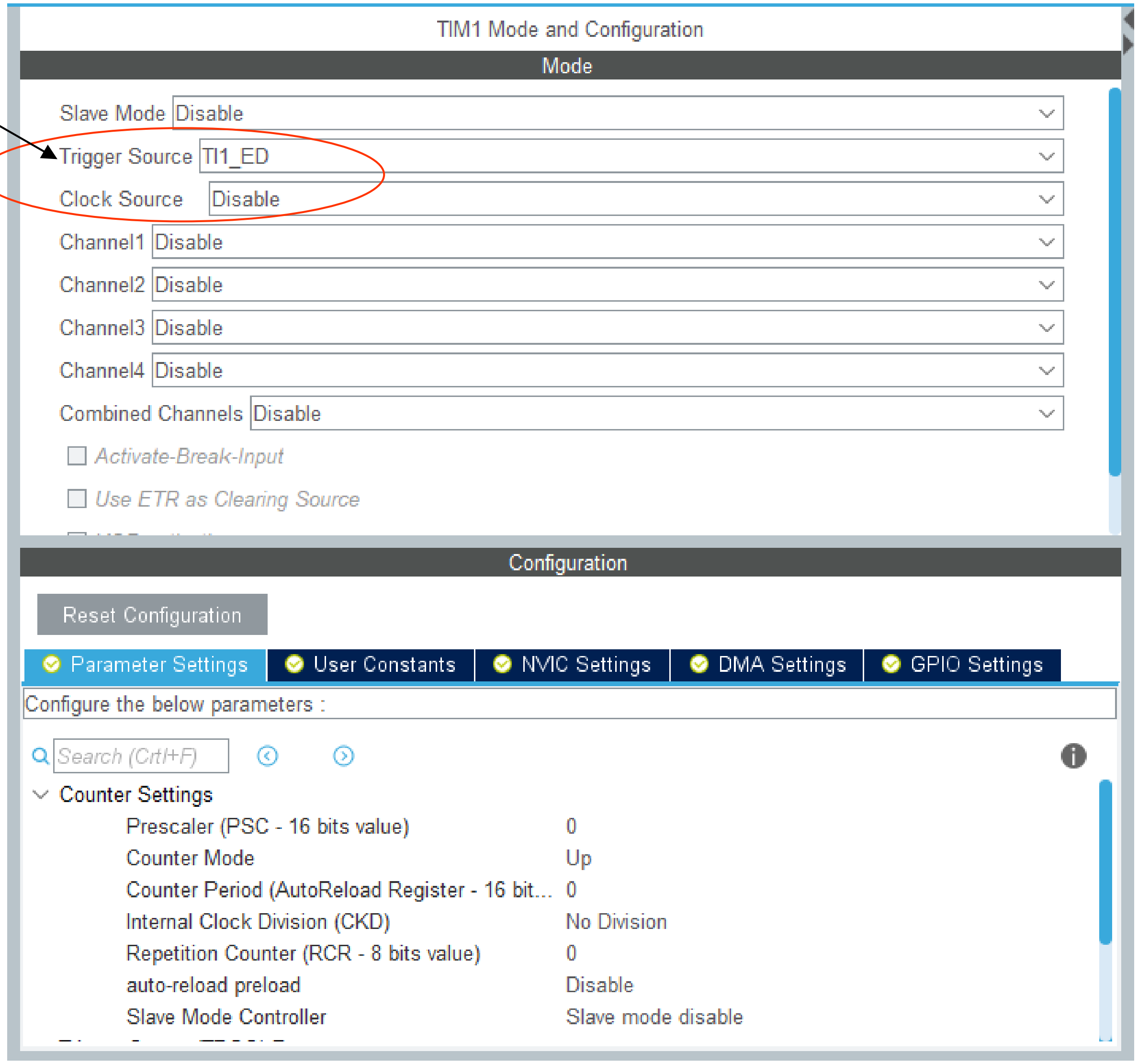


Example 1 – Event Counting using STM32



Example 1 – Event Counting using STM32

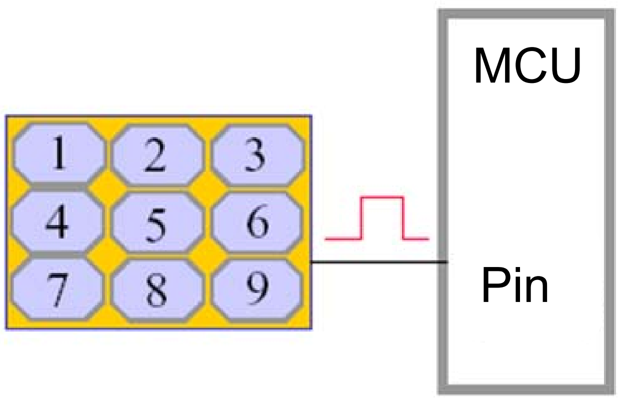
External switch



Example 2 – Timing the Length of Events

Description
Abstract idea of project (Define the functionality of the system)
Data format / representation
Programming Language
Communication Protocol
Physical connection (Pins assignment)
Hardware devices (Microcontroller, Peripherals)

- To measure how long (time) a key is being pressed:
- We can connect the keypad to another pin.
- In this way, the timer X will run when pulse is high (or a key is held down). When pulse is low (or a key is released), the timer X will be stopped.



Pseudo code:

Timer initialization
Turn on Timer X
……
……
If the pulse is high, trigger the timer X
If the pulse is low, stop the timer X

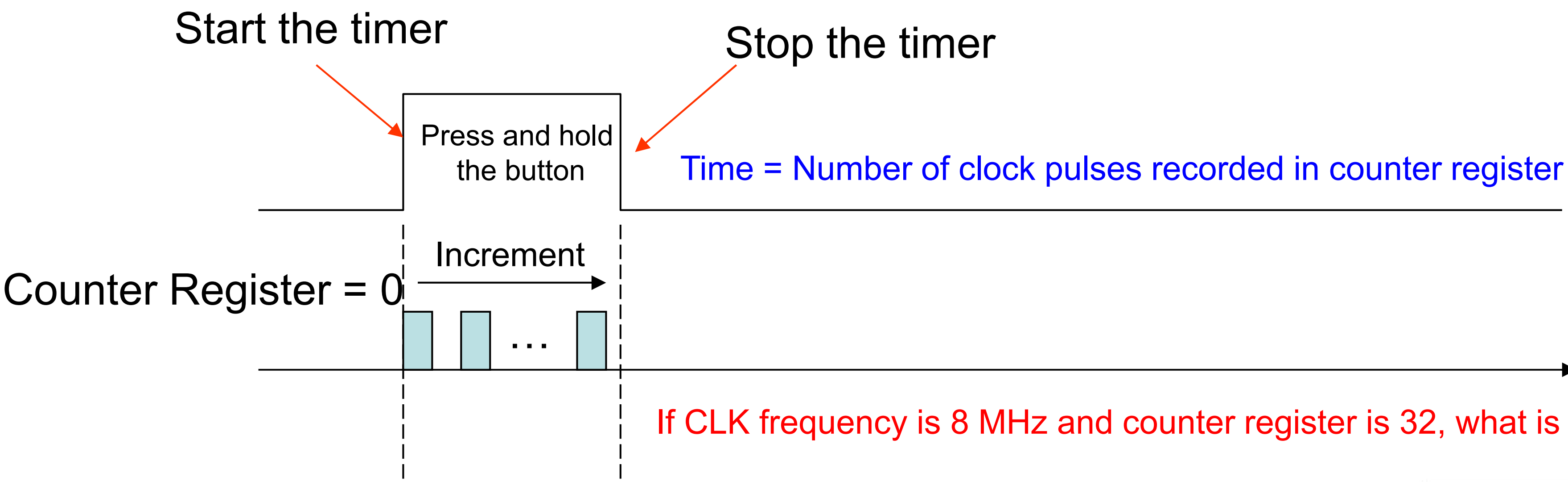
Read counter register

Step 1: Initialization
Configure the type of Timer / ~~Counter~~

Step 2: Implementation
Run / Stop the timer,
Read the counter register

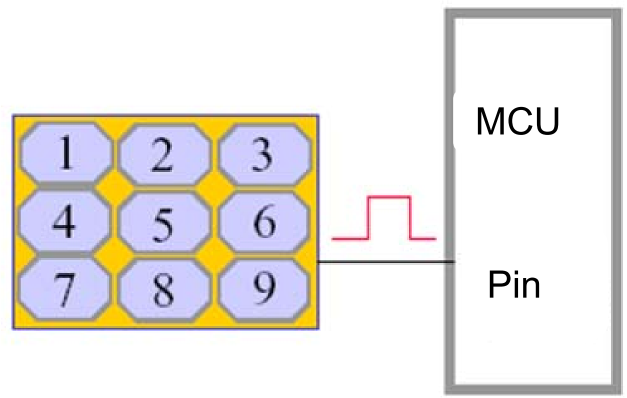
- From the current counter value and the frequency of the input clock, we can calculate the time elapsed for the event.

Example 2 – Timing the Length of Events



Time = Number of clock pulses recorded in counter register x the CLK period.

If CLK frequency is 8 MHz and counter register is 32, what is the time?

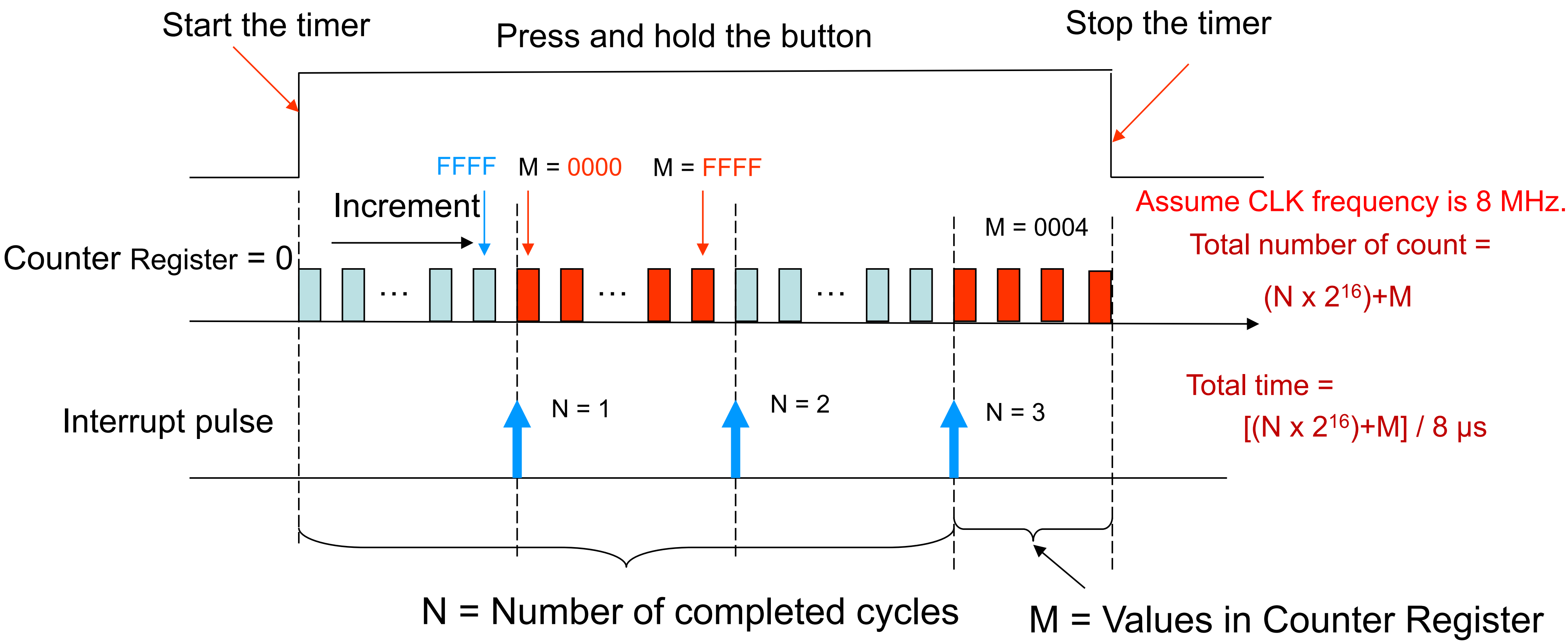


Counter register in STM32 is 16 bits. Max count = 2^{16} = 65,536.

Maximum time duration possible = (65,536 x 1/8) μs = 8.192 ms.

Assume that your application needs to time more than 8.192 ms. How will you achieve this?

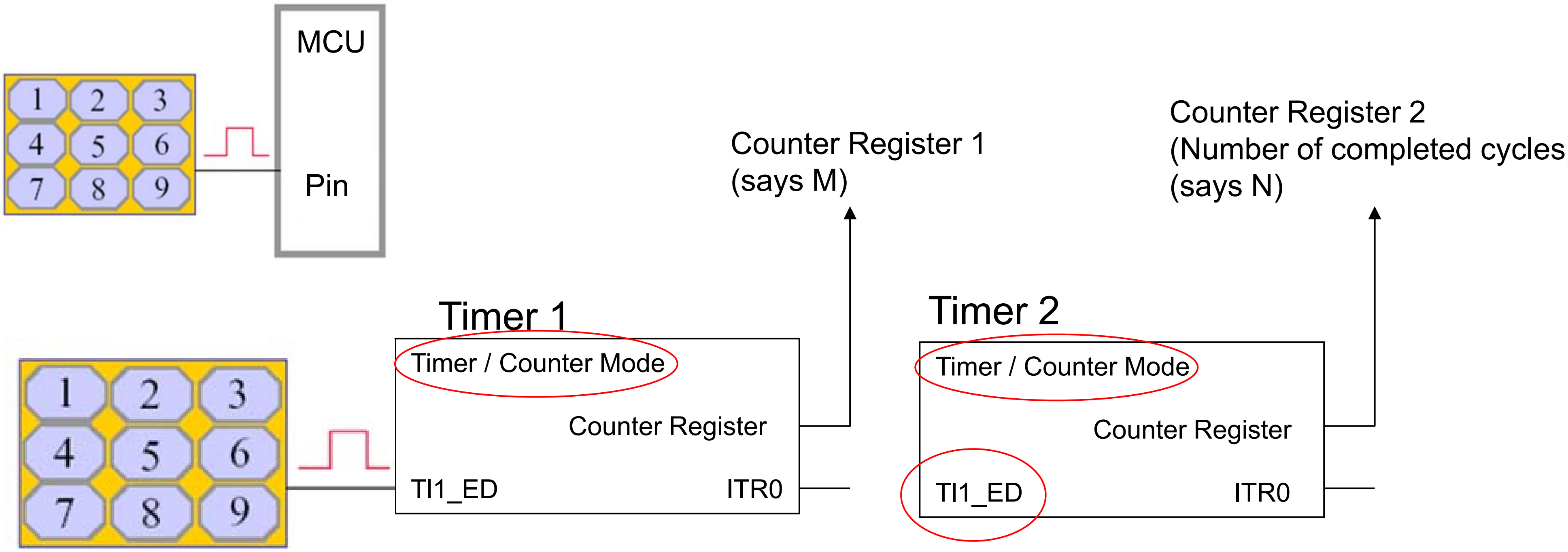
Example 2 – Timing the Length of Events



Design Problem: The application engineer tells you that the time needed for an event is at least 1 sec. How will you meet the requirement? Assume clock frequency is 4 MHz.

Solution: $N.\,2^{16}.\frac{1}{4MHz} \geq 1\,sec \rightarrow N \geq 61.03$

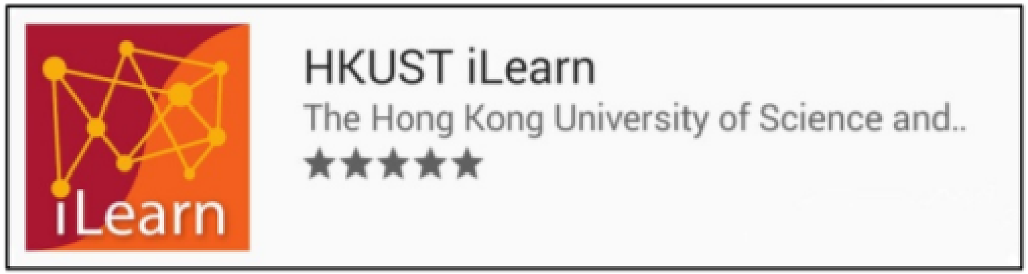
Example 2 – Timing the Length of Events



For every increment in 'N' (which happens when timer count hits FFFF), interrupt pulse will be generated at ITR0 of Timer 1, which triggers the input TI1_ED of Timer 2.

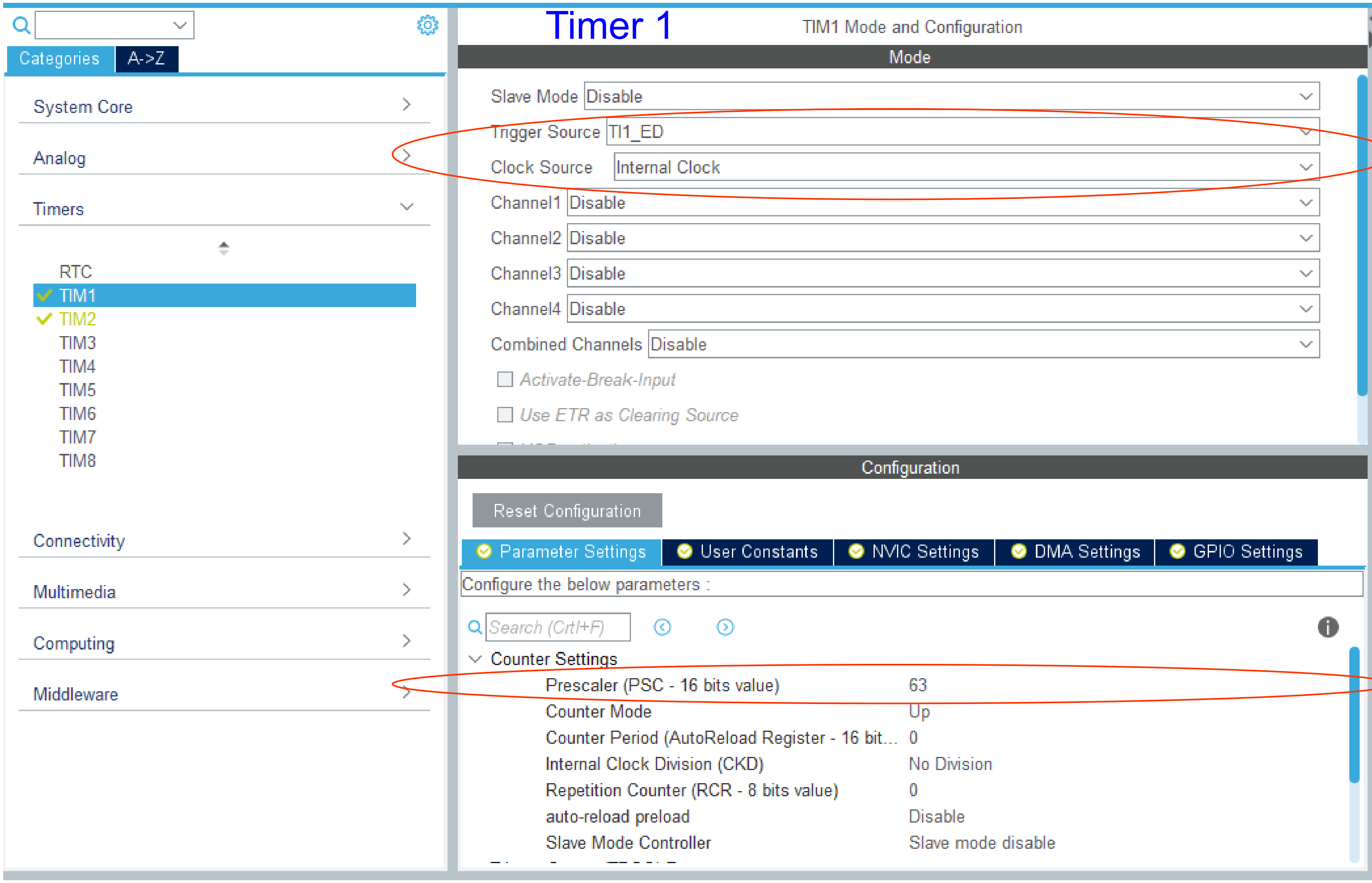
In-class Quiz (Question 1 – 4)

For Android devices, search **HKUST iLearn** at Play Store.

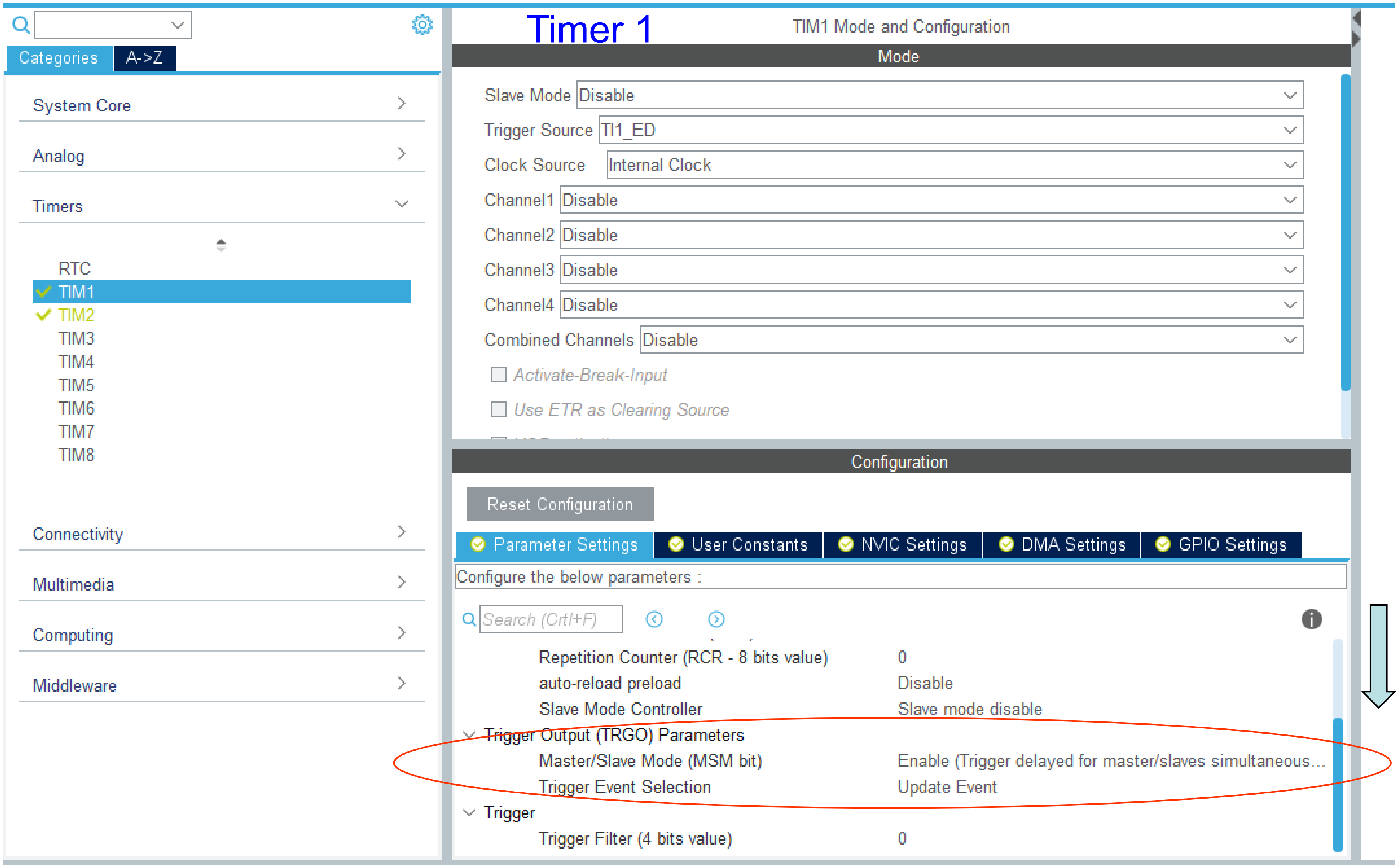


For iOS devices, search **HKUST iLearn** at App Store.

Example 2 – Timing the Length of Events

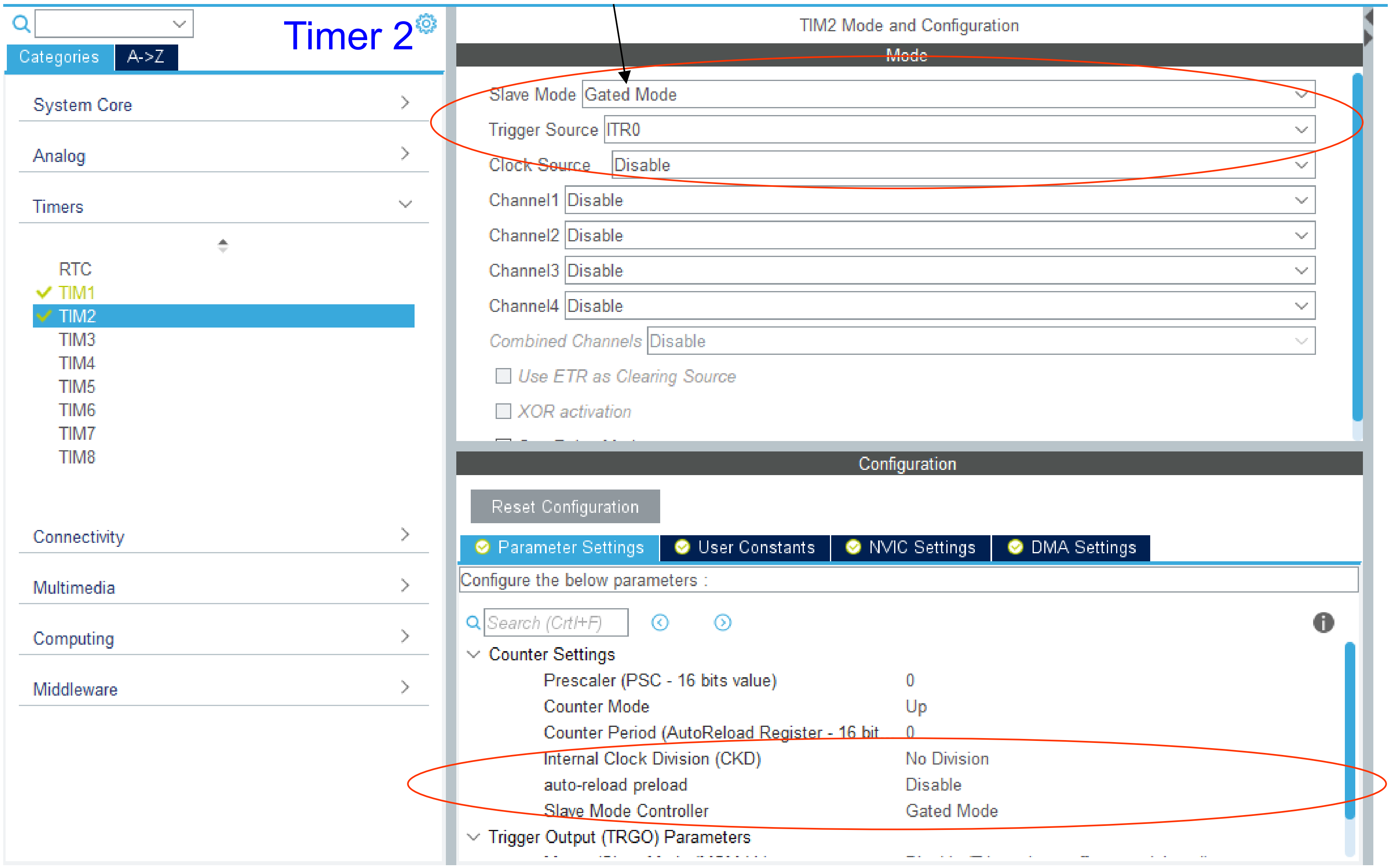


Example 2 – Timing the Length of Events



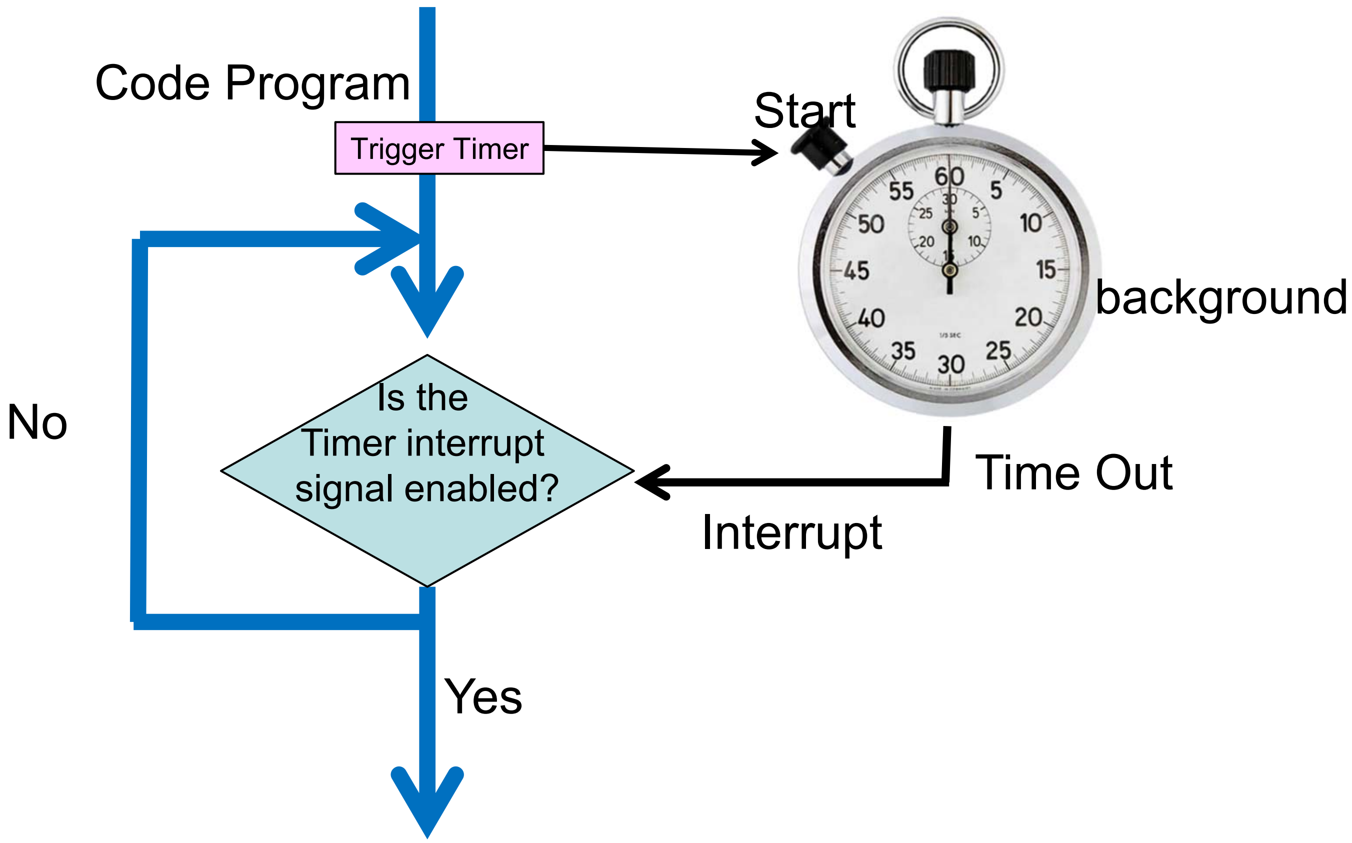
Example 2 – Timing the Length of Events

Gated mode: counter Active only when the level of input signal is High.



Program flow

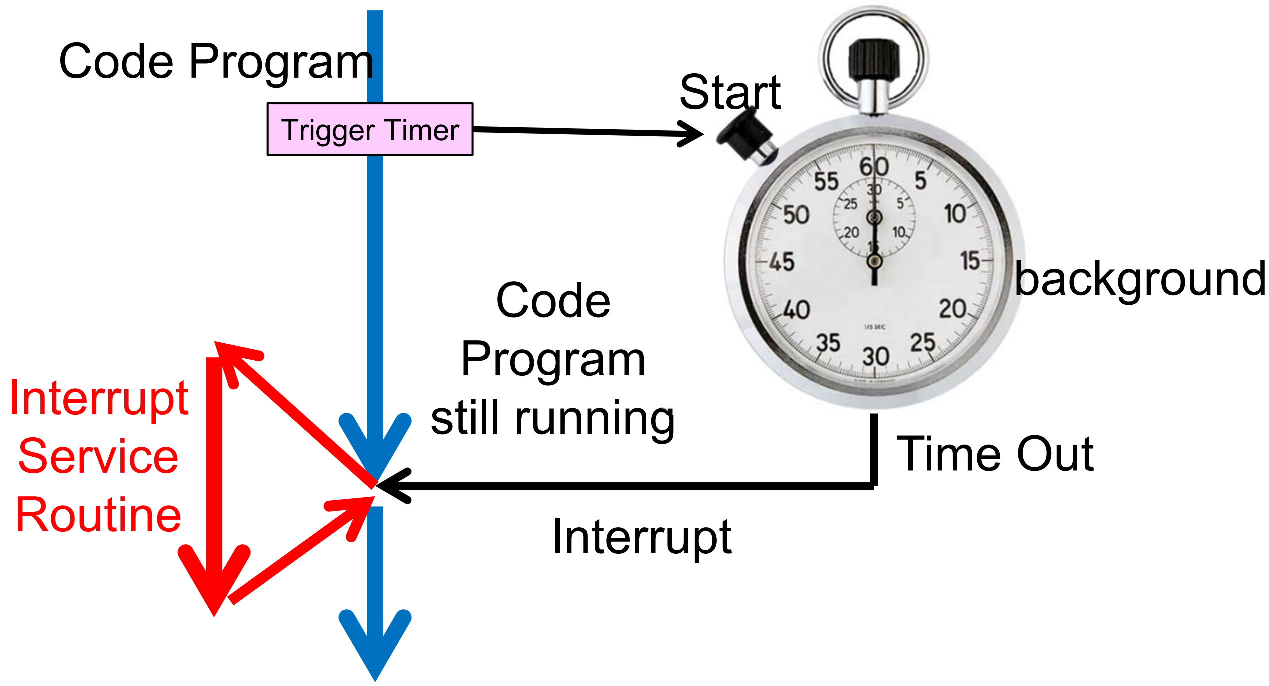
- Timer can be used as a **delay function** call.



Main program is put under hold until the timer time out and trigger the interrupt, which will enable the main program to resume. Purpose - To introduce a delay to the main program.

Program flow

– Interrupt I/O allows the clock to interrupt the CPU announcing that the device requires attention. This allows CPU to ignore devices unless they request servicing via interrupt



Main program continues to run until the timer time out and triggers the interrupt. Upon receiving the interrupt signal, the main program will stop execution, branch to execute the ISR and return to resume the main program after completing the ISR.

STM32 : Structure of general-purpose timer (TIM2 -TIM5)

- The high-density STM32F103xx performance line devices include up to two advanced control timers, up to four general-purpose timers, two basic timers, two watchdog timers and a SysTick timer.
  - TIM1 / TIM8 – advanced control timers
  - TIM2 / TIM3 / TIM4 / TIM5 – general purpose timers
  - TIM6 / TIM7 – basic timers

Table 4 compares the features of the advanced-control, general-purpose and basic timers.

Table 4. Timer feature comparison

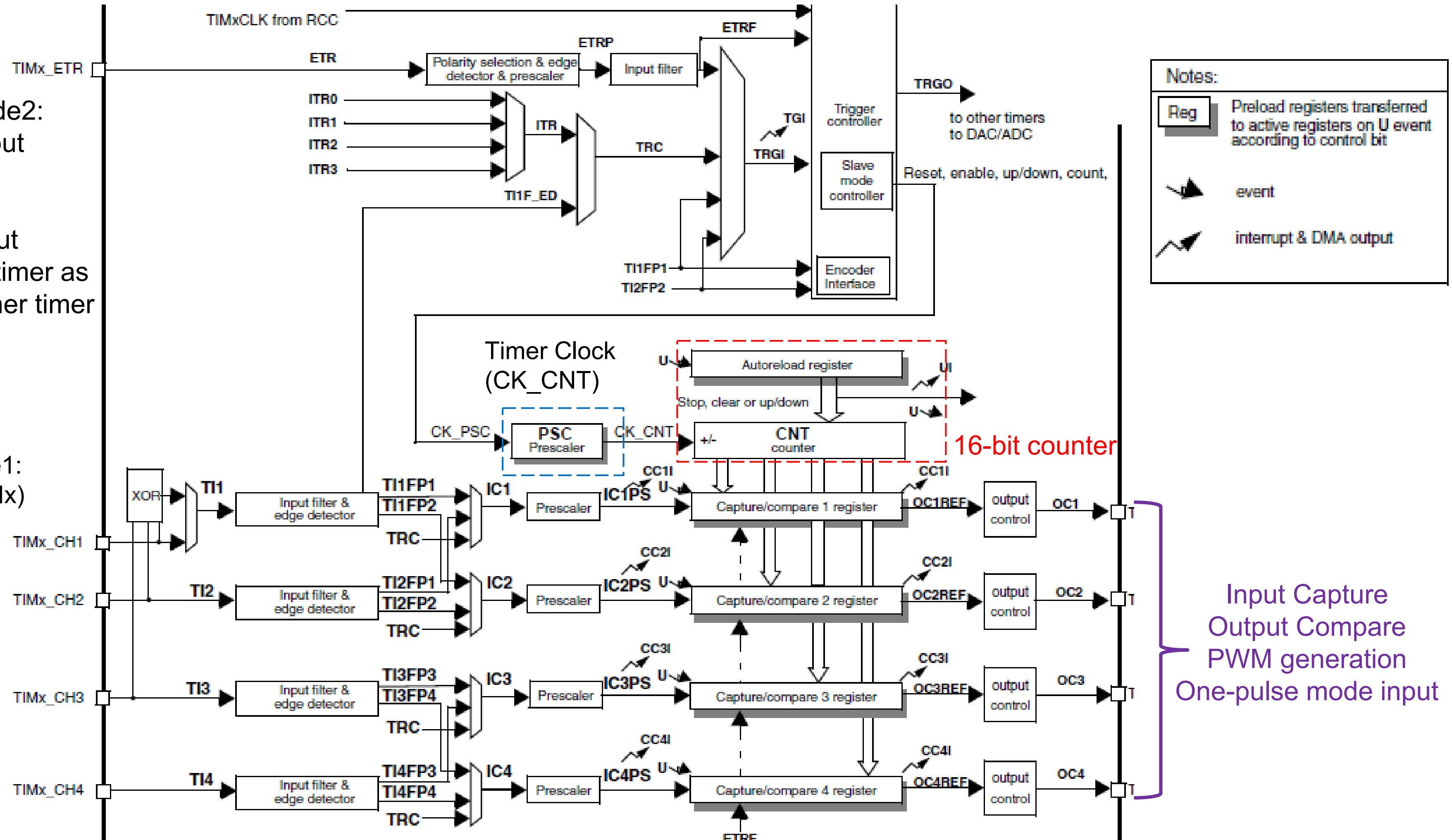
Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/compare channels	Complementary outputs
TIM1, TIM8	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4	Yes
TIM2, TIM3, TIM4, TIM5	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4	No
TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No

STM32 : Structure of general-purpose timer (TIM2 -TIM5)

External clock mode2: external trigger input (ETR)

Internal trigger input (ITRx): using one timer as prescaler for another timer

External clock mode1: external input pin (TIx)



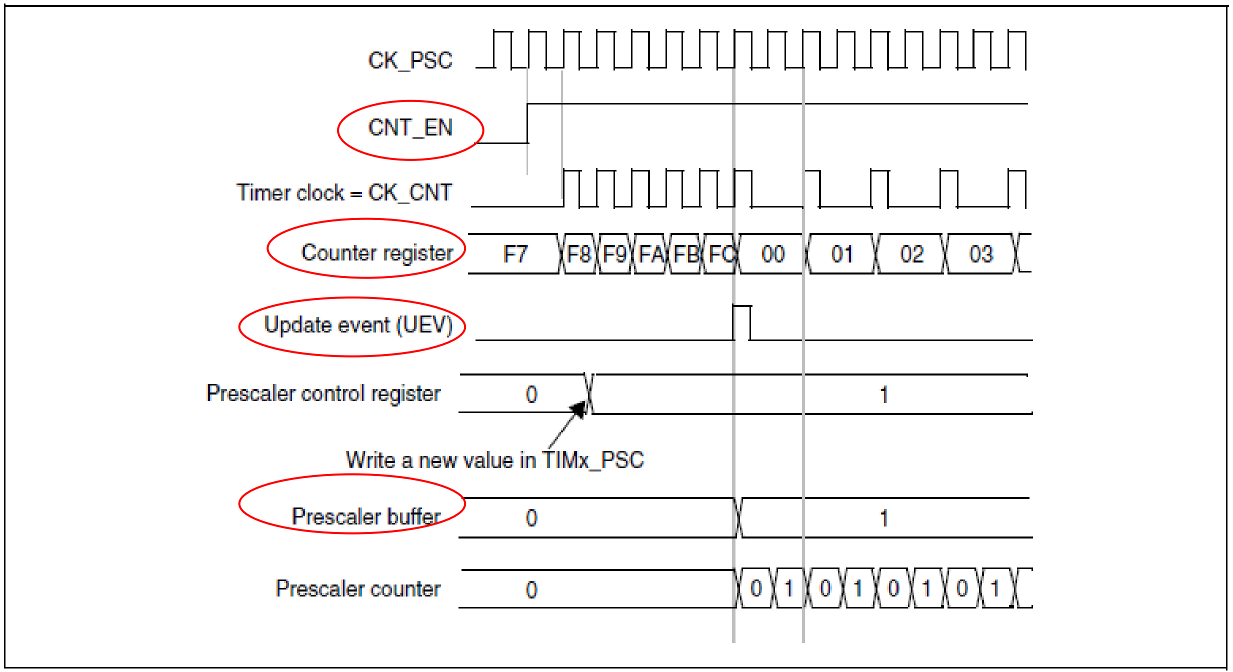
Output compare: When the counter matches the compare register value (eg. To indicate that a period is elapsed).

Capture event transfers counter value to capture register and triggers an interrupt/DMA request.

STM32 : Structure of general-purpose timer (TIM2 -TIM5)

Generate the Timer Clock

Figure 101. Counter timing diagram with prescaler division change from 1 to 2



* Upcounting configuration is employed.

STM32 : Structure of general-purpose timer (TIM2 -TIM5)

With auto-reload features, says TIMx_ARR = 0X36h

Figure 103. Counter timing diagram, internal clock divided by 1

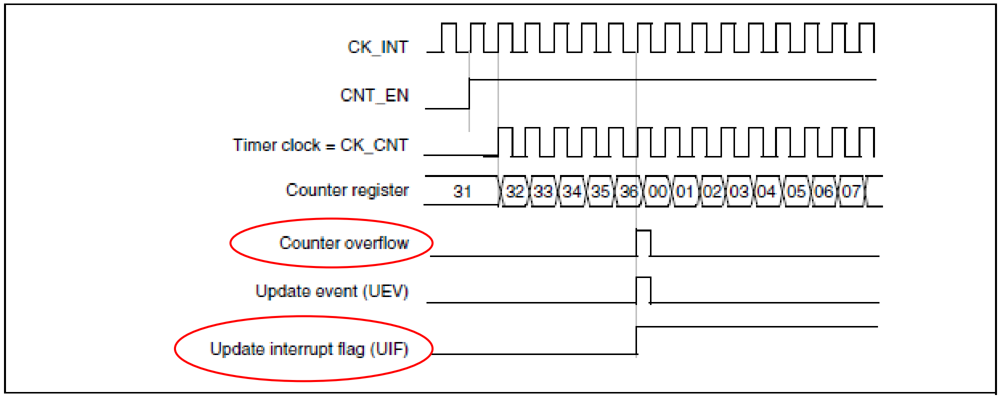


Figure 105. Counter timing diagram, internal clock divided by 4

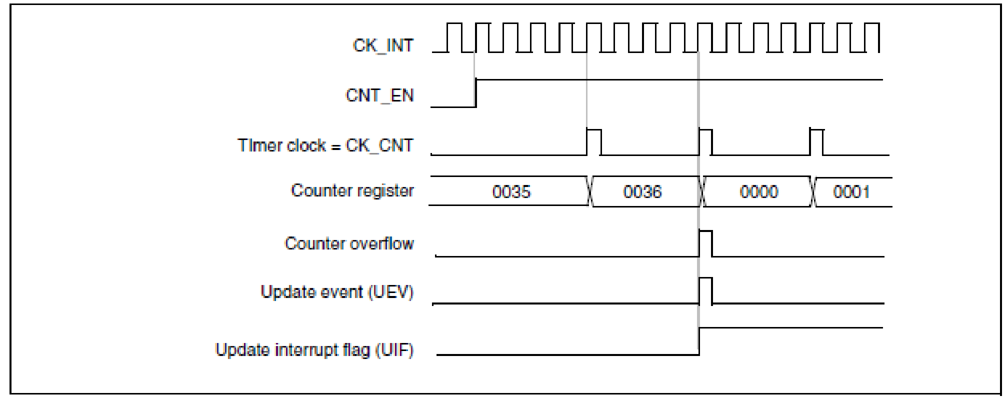


Figure 104. Counter timing diagram, internal clock divided by 2

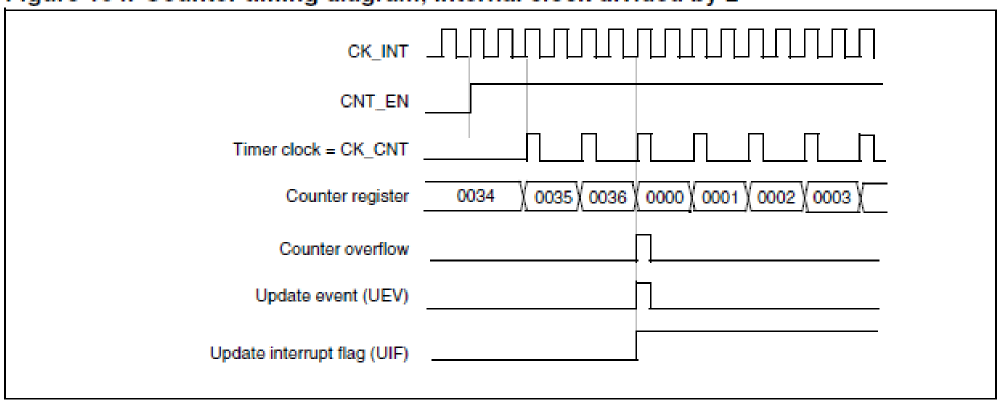
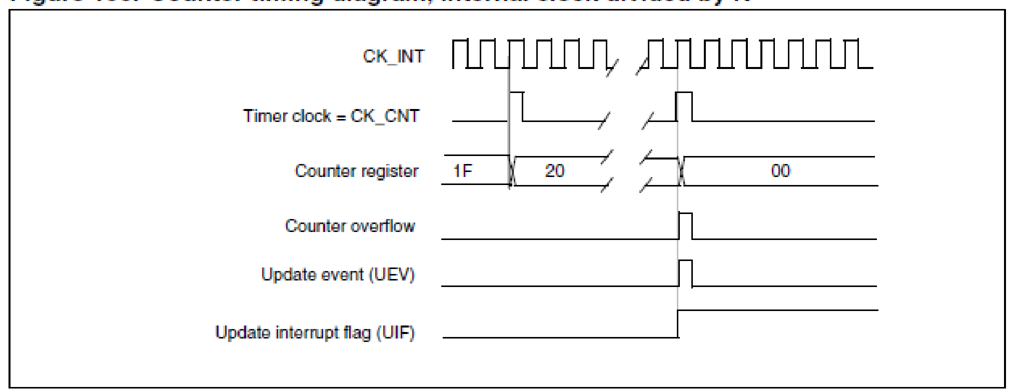


Figure 106. Counter timing diagram, internal clock divided by N



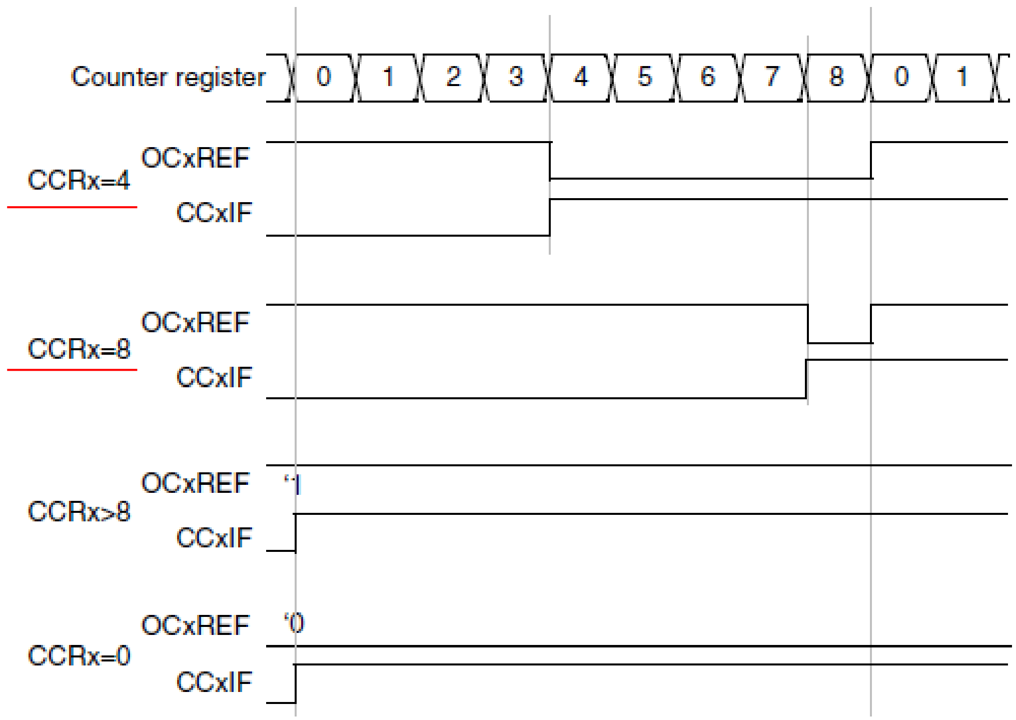
* Upcounting configuration is employed.

STM32 : Structure of general-purpose timer (TIM2 -TIM5)

ARR: Auto Reload Register. When ARR = 8, it will produce 0 – 8 counts (total 9 counts).

PWM generation

Figure 130. Edge-aligned PWM waveforms (ARR=8)



CCRx=4: Output drops to zero when count=4. Duty cycle = 4/9

CCRx=8: Output drops to zero when count=8. Duty cycle = 8/9

CCRx>8: Output drops to zero when count>8. Duty cycle = 9/9=1

CCRx=0: Output drops to zero when count=0. Duty cycle = 0/9=0

Resolution of duty cycle = 1/9.

How can we obtain a duty cycle of 50%?

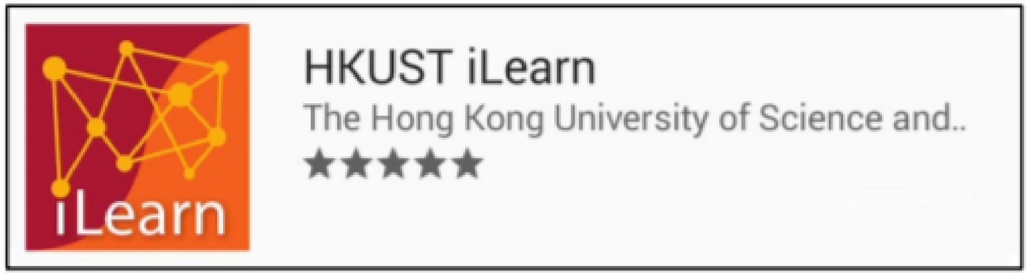
Solution: Set ARR = 9.

* Upcounting configuration is employed.

CCR – Capture Compare Register

In-class Quiz (Question 5)

For Android devices, search **HKUST iLearn** at Play Store.



For iOS devices, search **HKUST iLearn** at App Store.

STM32 : Structure of general-purpose timer (TIM2 -TIM5)

If you are not using CubeMX:

- In the initialization of timer, we need to configure
  - The parameters of input clock rate
    - TIMx Timebase includes
      - prescalar register (TIMx_PSC), auto-reload register (TIMx_ARR), etc
  - The parameters of output waveform
    - TIMx Output Compare, TIMx_CCRx Register
- For the implementation stage, we just need to enable the timer / counter pin
  - The result will be stored in the Counter Register (TIMx_CNT), Output Compare Register (OCxREF), etc

Can you configure these settings in CubeMX?

Real-world Examples

Bottle counting (Mineral water bottling plant)

To count the number of bottles:
Optical sensors + Counter

A bit more challenging problem: Can you propose a system that counts and displays the number of customers in the 7-11 store at a given instance of time?

Clue: You have to increase the count when customer enters and decrease the count when customer leaves.

Different problems, Similar solutions: Can you propose a system that counts and displays the number of vacant seats in the upper deck of KMB double decker bus?

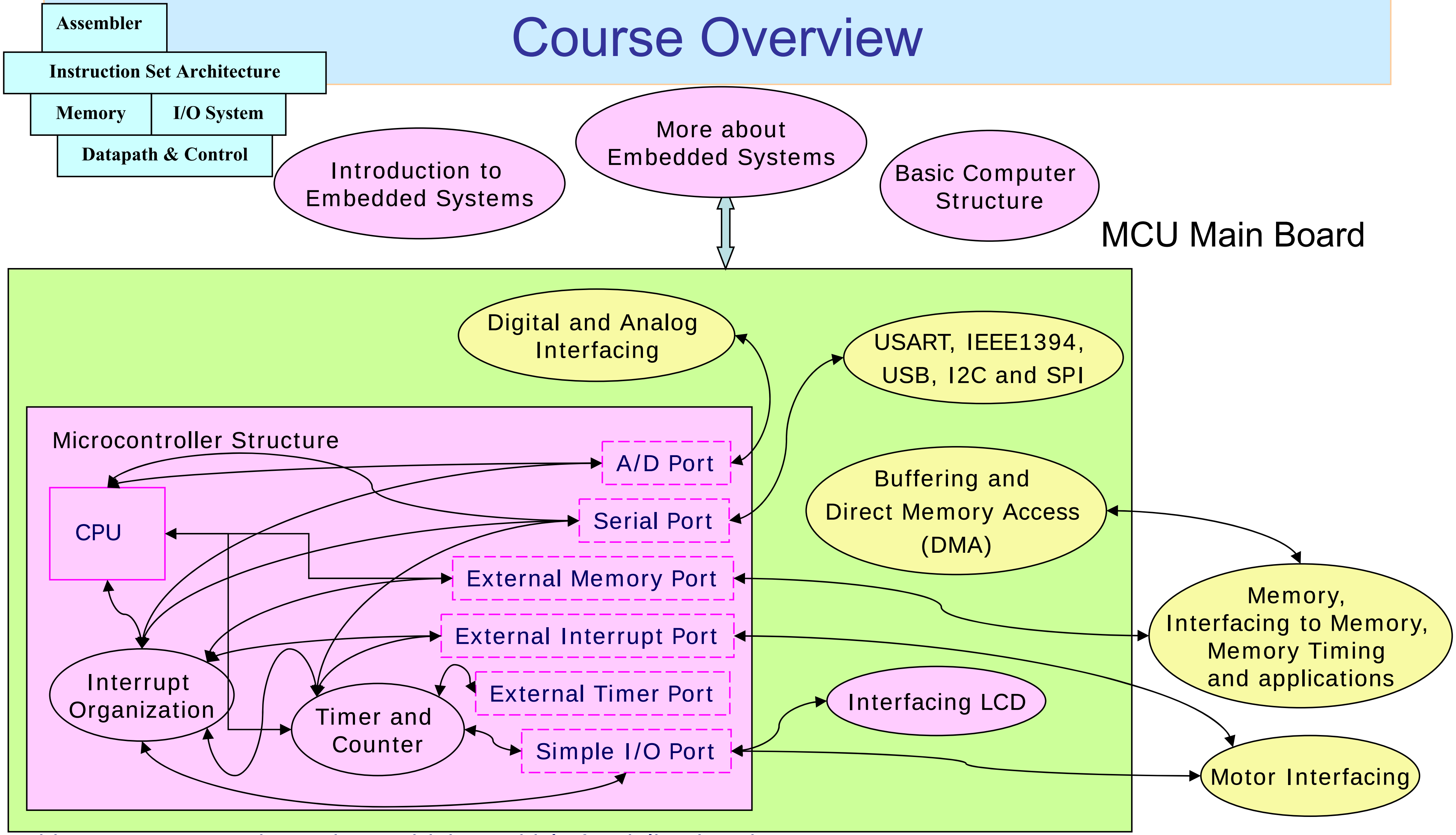
Laboratory Experiment

Description	Choices in this course
Abstract idea of project (Define the functionality of the system)	Generate PWM signals and drive the servo motor at different positions (left / right / center)
Data format / representation	Unsigned 16 bits
Programming Language	C-language
Communication Protocol	-
Physical connection (Pins assignment)	Pins for Timers
Hardware devices (Microcontroller, Peripherals)	Microcontroller: STM32 ARM Platform Servo motor

Reflection (Self-evaluation)

- Do you …
  - List some applications for timer / counter operation ?
  - State the important registers in the timer / counter operation?
  - Write the sub-routines for event counting or timing the length of event?
  - Describe the counter timing diagram in the count-down operation with different parameters (Clock frequency, up/down; auto-reload values)?

Course Overview



In this course, STM32 is used as a driving vehicle for delivering the concepts.

To be covered	In progress	Done