POLITECNICO
MILANO 1863

Programmable Timers
AA 2020-2021

Prof. William Fornaciari
Politecnico di Milano
Dip. di Elettronica, Informazione e Bioingegneria
william.fornaciari@polimi.it

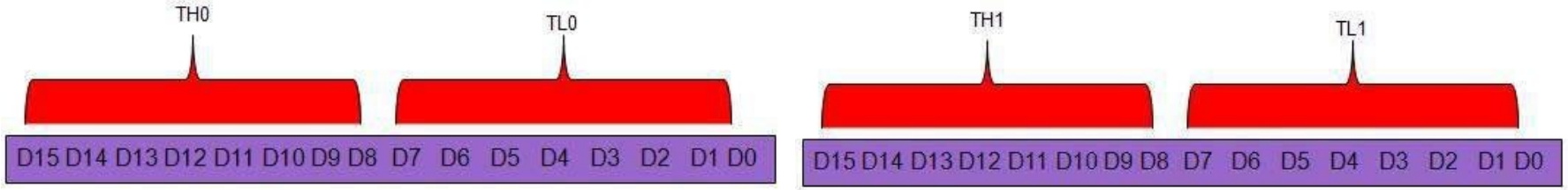
Timer vs counter

- **Timer** - specialized type of clock to measure time intervals
  - A timer that counts from zero upwards for measuring time elapsed is often called **stopwatch**. A device that counts down from a specified time interval is used to generate a time delay
- A **counter** is a device that stores (and sometimes displays) the number of times a particular event or process occurred, with respect to a clock signal
  - It is used to count the events happening outside the MCU. Counters can be implemented easily using register-type circuits

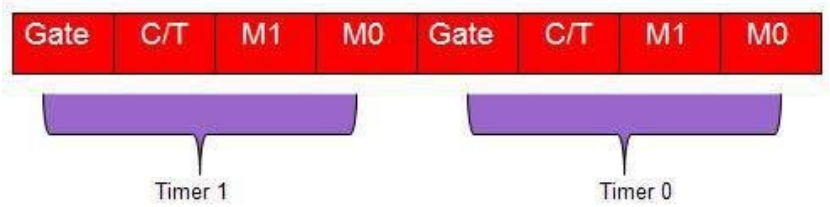
Timer	Counter
The register incremented for every machine cycle.	The register is incremented considering 1 to 0 transition at its corresponding to an external input pin (T0, T1).
Maximum count rate is 1/12 of the oscillator frequency.	Maximum count rate is 1/24 of the oscillator frequency.
A timer uses the frequency of the internal clock, and generates delay.	A counter uses an external signal to count pulses.

Ex. 8051 and associate registers

- Timers of 8051 and their Associated Registers
  - Two timers, Timer 0 and Timer 1, usable as timers or as event counters. Both Timer 0 and Timer 1 are 16-bit wide. Since the 8051 follows an 8-bit architecture, each 16 bit is accessed as two separate registers of low-byte and high-byte.



- TMOD (Timer Mode) Register
  - Both Timer 0 and Timer 1 use the same register to set the various timer operation modes. It is an 8-bit register in which the lower 4 bits are set aside for Timer 0 and the upper four bits for Timers. In each case, the lower 2 bits are used to set the timer mode in advance and the upper 2 bits are used to specify the location



POLITECNICO MILANO 1863

TMOD (Timer Mode) Register

- **Gate** − When set, the timer only runs while INT(0,1) is high
  - Every timer has a means of starting and stopping. Some timers do this by software, some by hardware, and some have both like 8051 timers. The start and stop of a timer is controlled by software using the instruction **SETB TR1** and **CLR TR1** for timer 1, and **SETB TR0** and **CLR TR0** for timer 0
- **C/T** − Counter/Timer select bit
  - Timer frequency is always 1/12th of the frequency of the crystal attached to 8051. Although various 8051 have an XTAL frequency of 10 MHz to 40 MHz, we normally work with the XTAL frequency of 11.0592 MHz. The baud rate for serial communication of the 8051.XTAL = 11.0592 allows the 8051 system to communicate with the PC with no errors
- **M1** − Mode bit 1
- **M0** − Mode bit 0

M1	M2	Mode
0	0	13-bit timer mode.
0	1	16-bit timer mode.
1	0	8-bit auto reload mode.
1	1	Spilt mode.

POLITECNICO MILANO 1863

TMOD (Timer Mode) Register

- Initializing a Timer
  - Decide the timer mode. Consider a 16-bit timer that runs continuously, and is independent of any external pins.
  - Initialize the TMOD SFR. Use the lowest 4 bits of TMOD and consider Timer 0. Keep the two bits, GATE 0 and C/T 0, as 0, since we want the timer to be independent of the external pins. As 16-bit mode is timer mode 1, clear T0M1 and set T0M0. The only bit to turn on is bit 0 of TMOD. Execute instruction − MOV TMOD,#01h
  - Now, Timer 0 is in 16-bit timer mode, but the timer is not running. To start the timer in running mode, set the TR0 bit by executing the instruction − SETB TR0
  - Now, Timer 0 will immediately start counting, being incremented once every machine cycle
- Reading a Timer
  - 16-bit timer can be read in two ways. Either read the actual value of the timer as a 16-bit number, or detecting when the timer has overflowed

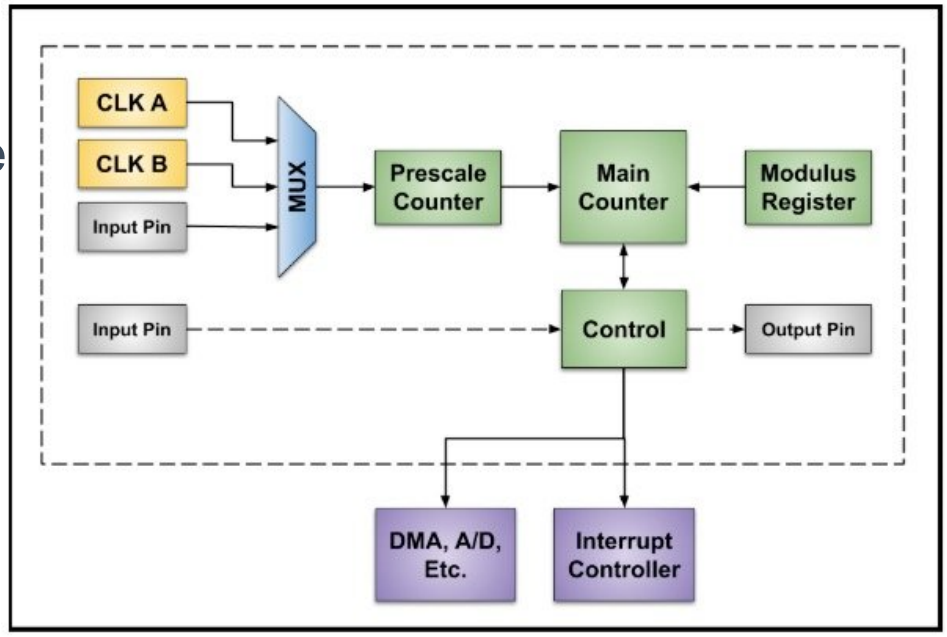
POLITECNICO MILANO 1863

Using Timers and Counters to Create Efficient MCU-based Designs

- Timers and counters are perhaps the most pervasive peripherals in MCU designs
  - About any application can use a timer or counter to improve performance, reduce power, or simplify the design by replacing repetitive- or looping-CPU operations with a simple timer or counter interrupt
  - Specific COTS devices or blocks of MCUs are tailored to support programmable timers
  - Some of the most advanced timer/counter features are used for PWM applications for motor control
    - These counters implement as much often the motor-related PWM functions as possible using dedicated hardware to free the processor for doing higher-level functions
    - Manufacturers are creating more application-oriented development kits and reference designs. As just one example, Renesas provides a complete motor-control development kit, the YMCRPRX62T

Common features in many timers

- Every timer needs a clock source or timebase
  - There are often several clock sources and one is selected. Sometimes an external clock source is an option
  - To increase the count range, the selected clock goes to a "prescaler" which divides the clock before it goes to a main counter
  - The dividing factor of the prescaler is limited to powers of 2. Some prescalers go up to 2^{16} or 65,536
  - The output of the prescaler goes to a main counter which is often 16 bits wide and counts within the range of 0 to 65,535
  - The count range of the main counter is set by a "modulus" value (**M**) stored in a register. This is a down counter which reloads the modulus value on the next clock after it reaches 0



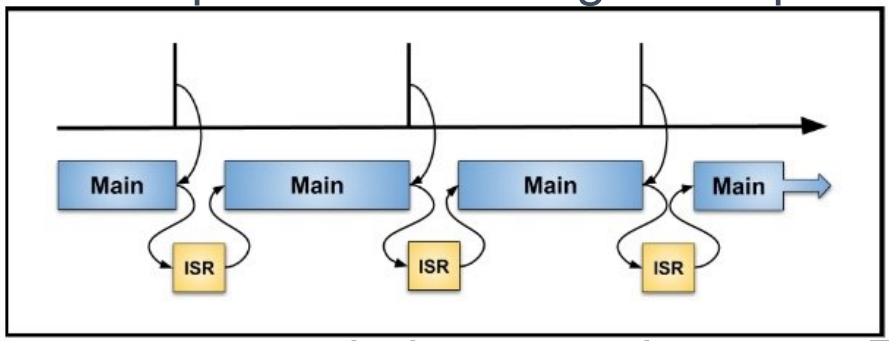
POLITECNICO MILANO 1863

Common features in many timers

- Control logic defines the type of timer and varies widely for different types. Included in the control logic are control and status registers. These registers are very different for different timers
- Initial configuration often includes:
  - Selecting a clock source
  - Setting a divide factor for the prescaler
  - Setting a modulus value
  - Setting various control bits
  - If used, enable a processor interrupt
  - If triggering a DMA operation or other peripheral, set up the trigger conditions
  - If the timer is connected to an input or output pin, set up the pin connections. This is typically done as part of the general configuration of the microcontroller

Polling and Interrupts

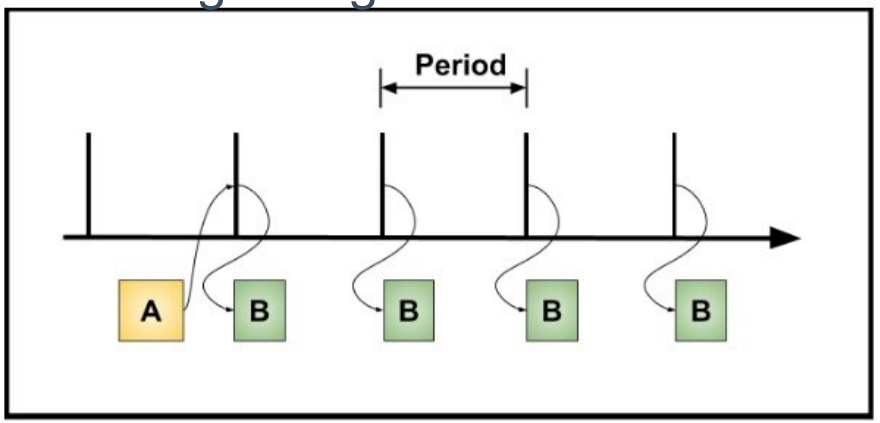
- Polling is periodically reading status registers to detect timer events or the current value of a counter
  - This type of coordination can use a lot of processing time or the response time can vary a lot with a complicated program. The solution for these problems is using interrupts



  - A timer event occurs and triggers an interrupt. For example, an interrupt occurs when a down counting timer reaches 0 and reloads the modulus in the main counter
  - The timer sends a hardware signal to an “interrupt controller” which suspends execution of the main program and makes the processor jump to a software function called ISR, then resume to Main
  - Main program does not spend time checking for a timer event. The response to the timer event can be very fast and predictable

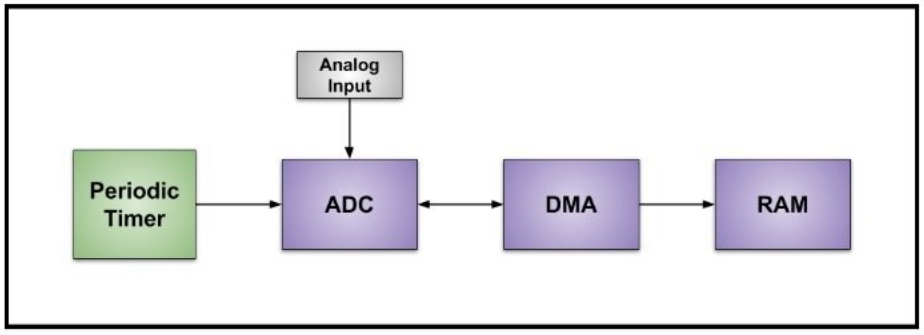
Periodic Timers

- Periodic timers produce repetitive markers or “ticks” with a fixed period as shown below
  - The major parameter is the period which is set with a counter modulus. Ex. provide the baud rate clock for the on-chip serial port
- “A” causes an action “B” to occur periodically under tight control of the timer. Some examples are:
  - Controlling the polling of digital inputs like pushbuttons
  - Scheduling of tasks by a real-time operating system (RTOS)
  - Accurately pacing DMA transfers to a digital-to-analog converter
  - Triggering an analog-to-digital converter for accurate sample rate



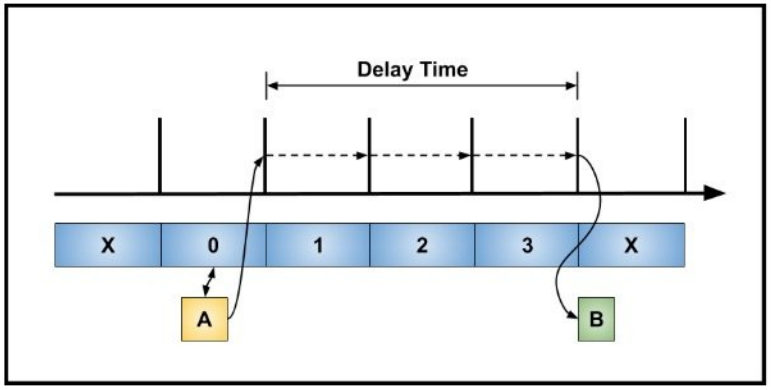
Periodic Timers – ADC example

- The periodic timer starts by triggering an analog-to-digital conversion of an analog signal
- When the conversion is done, the analog-to-digital converter tells the DMA controller to move the result to memory
- Then, everything waits for another trigger from the periodic timer. When the required amount of data is acquired, the DMA controller tells the main program
- An important part of this data acquisition is accurate pacing by the periodic timer without the uncertainty of timing by software
- The software is completely free to do something else. With some low-power microcontrollers, the processor, ADC, and DMA can even go to sleep and save power while waiting for their turn



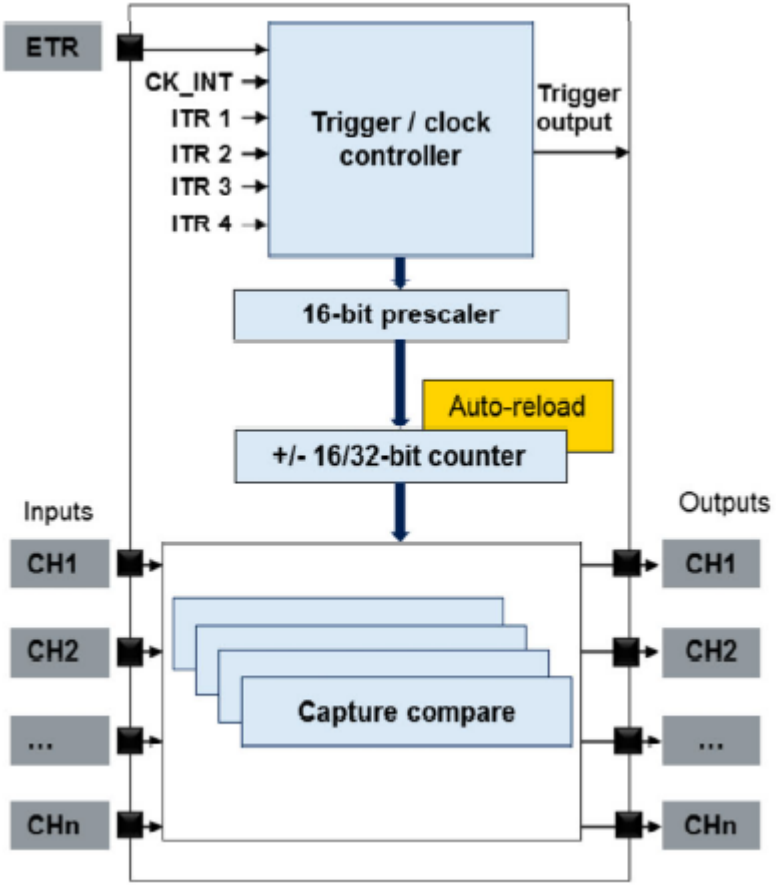
Delay

- Delay functions cause something to happen after a period of time has elapsed. The diagram shows “A” resetting a timer to 0 and starting a delay of three ticks before “B”
- An example of delay is debouncing a pushbutton
  - An initial press of a pushbutton is detected by polling an input, or there is an interrupt. Then, a delay is started by “A” to wait for any mechanical bounce of the pushbutton to end. After the delay, the pushbutton is sampled again in “B”. If the pushbutton is still active, a valid button press is detected
  - Using a timer for the delay allows the processor to do other things during the debounce time and the delay is accurate and repeatable.



STM32G0-GPTIM

- Gen. purpose and basic timers embedded in STM32
  - Multiple timing units providing timing resources



- Internally (triggers, time bases)
- Externally, for outputs and inputs
  - Waveform generators (PWM)
  - Measurement of frequency or timing
- Linked to peripherals (ADC, DAC, internal connectivity among peripherals, Real time clock, comparator output can trigger events, …)

8254 programmable interval timer

- 3 independent 16-bit counters each capable of handling clock inputs up to 10 MHz and with two inputs (Clock and Gate) and one output. Gate is to enable or disable counting
  - When any value of count is loaded and value of gate is set(1), after every step value of count is decremented by 1 until it becomes zero. Depending upon the value of CS, A1 and A0 we can determine addresses of selected counter

CS	A1	A0	SELECETION
0	0	0	C0
0	0	1	C1
0	1	0	C2
0	1	1	Control Register