ECE/CS 5780-6780 : Embedded Systems Design

Timers

Pierre-Emmanuel Gaillardon
Department of Electrical and Computer Engineering – University of Utah

Salt Lake City, UT, USA

Objectives for Today!

- Get familiar with the ARM M processor architecture
- Understand the architecture of a microcontroller
- Acquire the fundamentals of the hardware/software interface
- Acquire the fundamentals for sensing and controlling the physical world
- Learn modeling techniques for embedded system design
- **Understand the usage of several peripherals** through labs
- Design a complete embedded system – from specs to realization
  - Realization of a PCB
  - Behavioral modeling of the system
  - Complete SW realization

Timers: Generalities

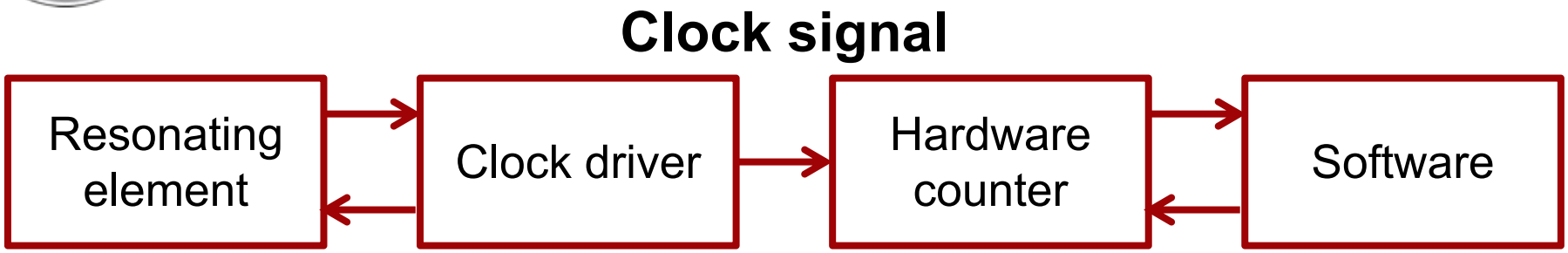
Time in Embedded Systems

- Why do we need to be accurate?
- BECAUSE of the applications:
  - Scheduling of computation
    - Scheduler in operating systems
    - Real time operating systems
  - Signal sampling and generation
    - Audio sampling at 44.1 kHz
    - TV/video generation (sync, vsync)
    - Pulse Width Modulated (PWM) signals
  - Communication
    - Media Access Control (MAC) protocols
    - Modulation
  - Navigation
    - GPS

Time Precision in Control – ABB Robots

http://www.youtube.com/watch?v=SOESSCXGhFo

Keeping time in Electronics

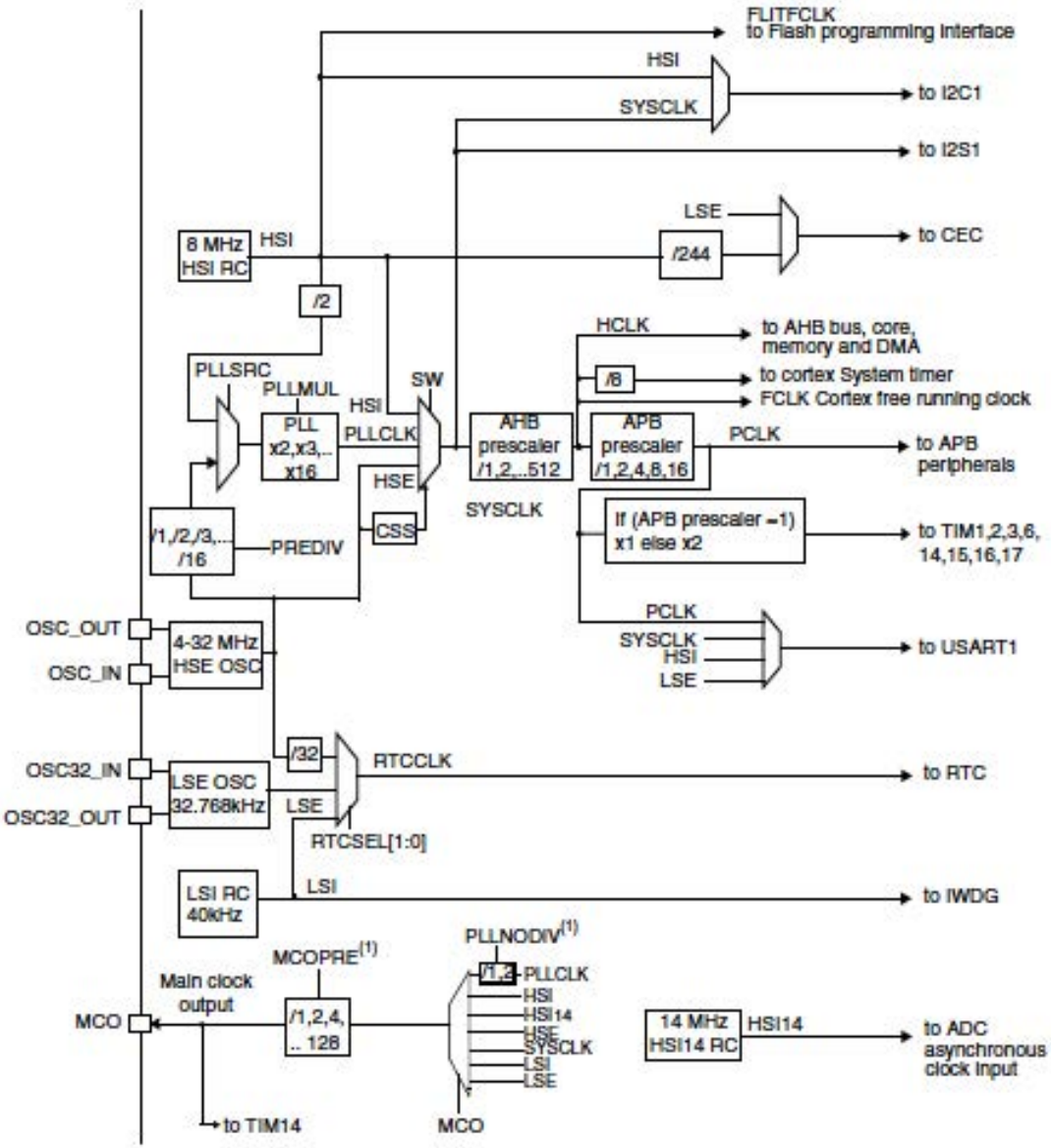


- Quart Resonator
  - Quartz crystal can be made to resonate due to Piezoelectric effect.
  - Resonate frequency depends on length, thickness, and angle of cut.
  - Very stable (<100ppm) but not all frequencies possible.
- MEMS Resonator
  - Arbitrary frequency, potentially cheap, susceptible to temperature variations.
- Inverter Ring
- LC/RC circuits
- Atomic clock, and many more.

Clock sources in STM32F072 MCU

- Various clock sources can be used to drive the system clock (SYSCLK):
  - HSI 8 MHz RC oscillator clock
  - HSE oscillator clock
  - PLL clock
- The devices have the following additional clock sources:
  - 40 kHz low speed internal RC (LSI RC) which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop/Standby mode.
  - 32.768 kHz low speed external crystal (LSE crystal) which optionally drives the real-time clock (RTCCLK)
  - 14 MHz high speed internal RC (HSI14) dedicated for ADC.

Clock Distribution Network is a Nightmare!



How to Configure the Clock Distribution Network?

- Program a bunch of registers:
  - Clock Control Registers: RCC_CR, RCC_CR2
  - Clock Configuration Register: RCC_CFGR
  - …
- How do we set them? Follow the documentation!

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL NODIV	MCOPRE[2:0]			MCO[3:0]				Res.	Res.	PLLMUL[3:0]				PLL XTPRE	PLL SRC[1]
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL SRC[0]	ADC PRE	Res.	Res.	Res.	PPRE[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw				rw	rw	rw	rw	rw						

Bit 31 **PLLNODIV:** PLL clock not divided for MCO

This bit is set and cleared by software. It sw

0: PLL is divided by 2 for MCO

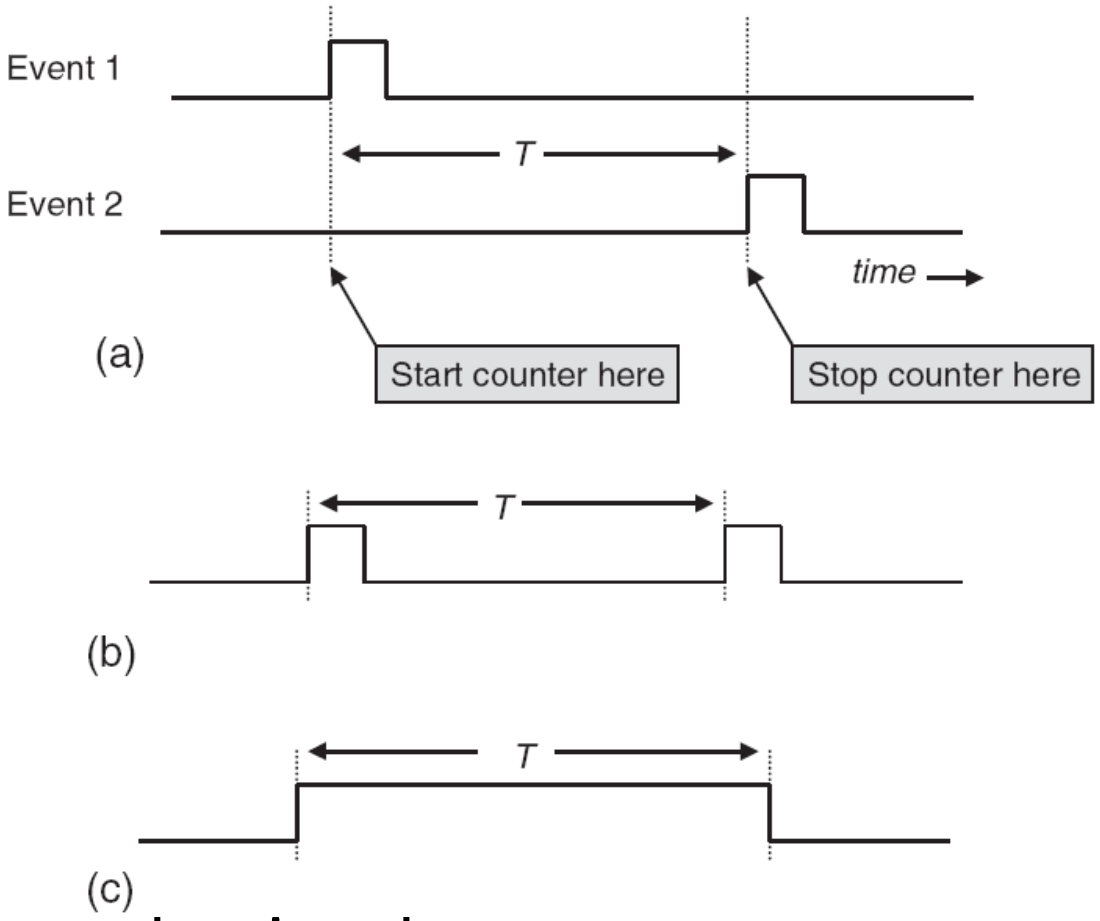
1: PLL is not divided for MCO

Luckily enough – it gets configured in *system_stm32f0xx.c*

Keeping track of Time

- Now that we have a clock, how do we keep track of time?
- With a basic counter!
- How do we make a counter?
  - In C? Get in groups and write a simple program.
  - In Hardware? Get in groups and sketch a quick architecture.

Timer Applications?



(a) Measure the time between two events

(b) Measure the time between two pulses

(c) Measure a pulse duration

General Purpose Timers

General Purpose Timers

- One of the most important issues in embedded control is the programming of *timers*
- Without accurate timing, digital control engineering is not possible – the control signals (controller action) have to happen at the exact right moment in time, e.g. timing control of an engine, etc.
- Large embedded systems are often centered around small *real-time kernels* which offer services for *multi-tasking* applications; smaller applications make use of *Interrupt Service Routines (ISR)* to achieve accurate timing control

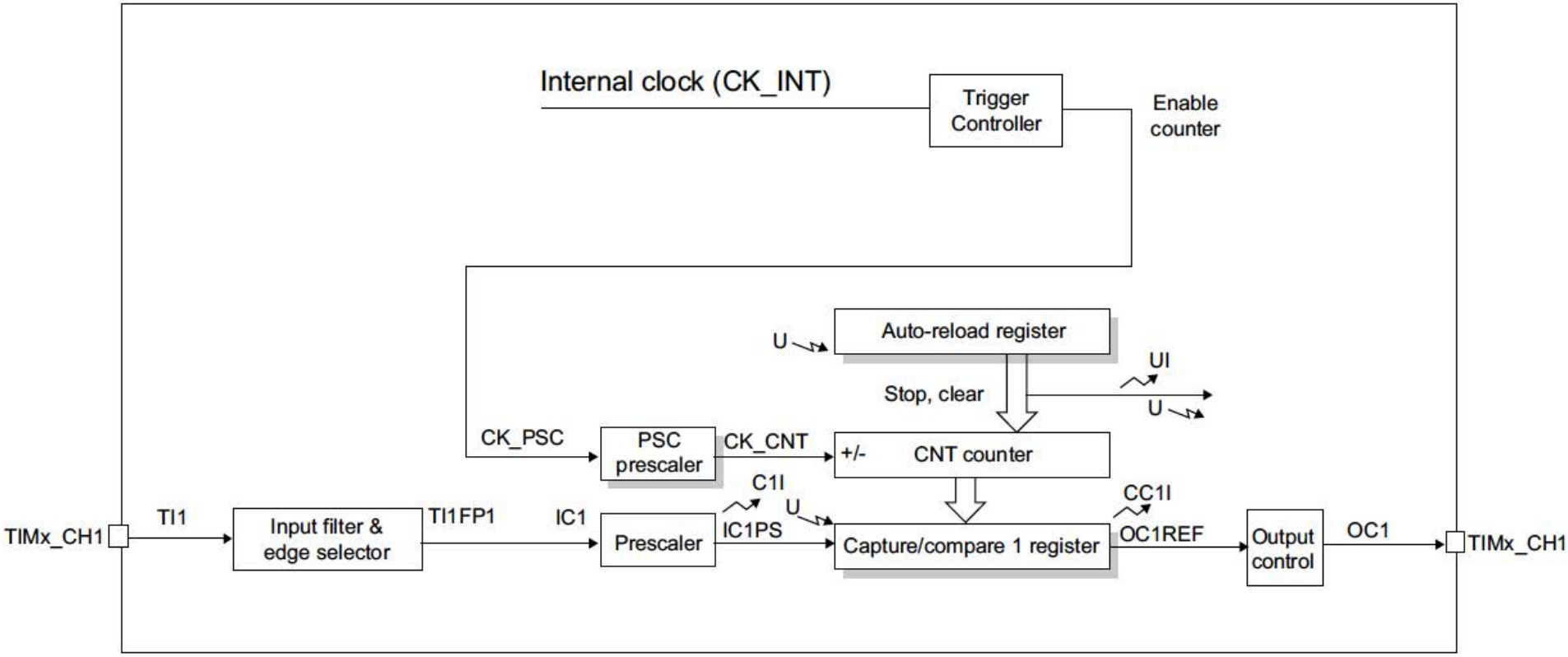
Microcontroller Counter/Timers

- Timers are special registers within the microcontroller that are configured as binary counters
- Timer registers can be preloaded with a starting value and possibly have an ending value (other than overflow) specified
- Overflow or hitting the end value generates an interrupt to the processor
- Timer register clocking:
  - Internal CPU clock - timer mode
  - External clock via a dedicated I/O pin - counter mode
- Timers are the basis for PWM outputs and Capture inputs
- Timers are also sometimes used by the external communications functions (UARTS, etc.)

Timers in our STM32F072

- Up to 11 timers
  - One 16-bit 7-channel advanced-control timer for 6 channels PWM output, with deadtime generation and emergency stop
  - One 32-bit and one 16-bit timer, with up to 4 Input Capture/Output Compare capabilities, usable for IR control decoding
  - One 16-bit timer, with 2 IC/OC, 1 OCN, deadtime generation and emergency stop
  - Two 16-bit timers, each with IC/OC and OCN, deadtime generation, emergency stop and modulator gate for IR control
  - One 16-bit timer with 1 IC/OC
  - Independent and system watchdog timers
  - SysTick timer: 24-bit downcounter
  - One 16-bit basic timer to drive the DAC

Let's look at a simple one – TIM4



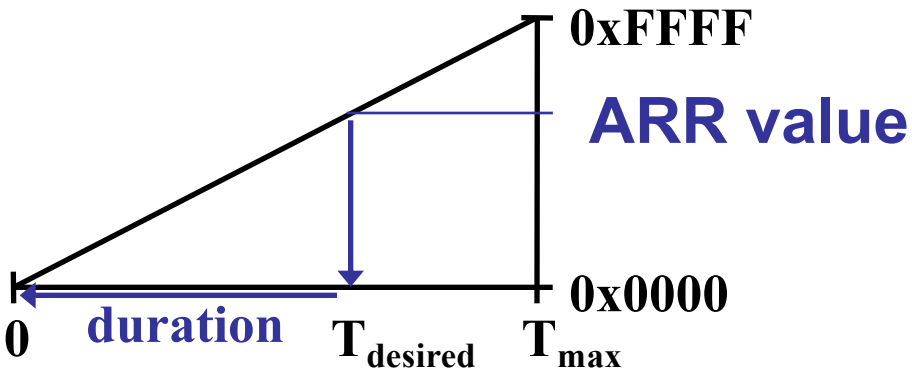
What is inside a Timer?

- A counter – 16 bits here – with its auto-reload register.
- The counter can count up.
- The counter clock can be divided by a prescaler.

- As a result, the time-base unit includes:
  - Counter register (TIMx_CNT)
  - Prescaler register (TIMx_PSC)
  - Auto-reload register (TIMx_ARR)

Timers Principle

- Timers are probably best explained using the following diagram:

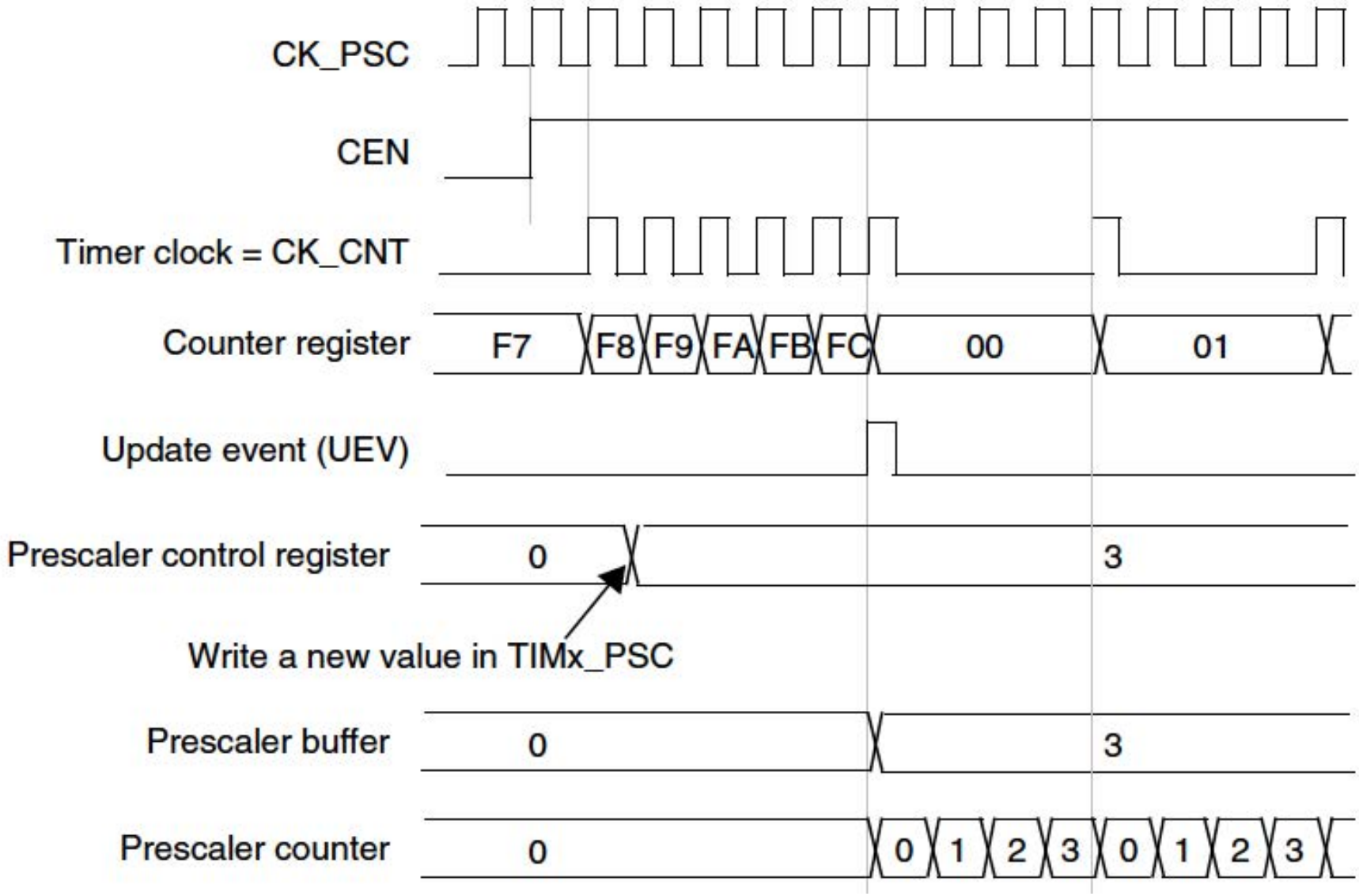


- The *CNT* register counts until it reaches the value contained in *ARR* then it resets and triggers an interrupt (or at least raises a flag…)

What is a Prescaler?

- The prescaler can divide the counter clock frequency by any factor between 1 and 65536.
- It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register).
- It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Counter Timing Diagram with PSC change



Timer Counter Mode

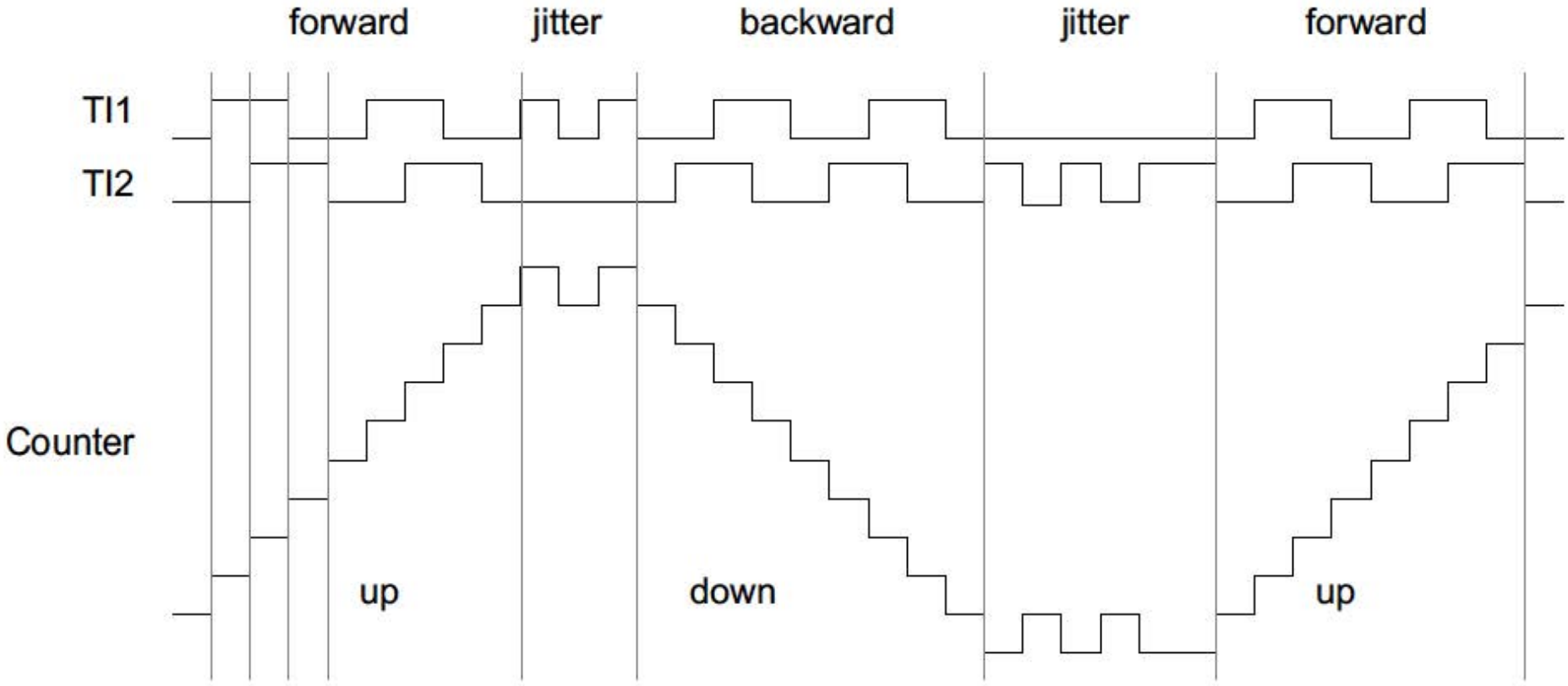
Timer / Counter modes

- In *timer mode*, the *timer register* is incremented with every tick of the internal clock source
- In *gated timer mode*, the (internally incremented) timer can be switched on/off via an external signal
- In *counter mode*, the *counter register* is incremented with rising/falling/both edges of an external signal
- The *incremental interface mode* supports interfacing to *incremental encoders;* both rate and direction are determined and the counter register is incremented / decremented accordingly

Input Capture / Counter Mode

- Capture inputs are used to count external events
- A general purpose timer is associated with each capture register
- The timer is incremented each time the external event occurs
- The external event can be a rising or falling edge on the capture input.
- The capture (timer) register value can be read or an interrupt can be generated when it equals a specified value

Incremental (Quadrature) Encoder



What is the usage of a Quadrature Encoder?
How is it made?
Why is that of our interest?

Quadrature Encoder Interface

- An external incremental encoder can be connected directly to the MCU without external interface logic – through inputs TI1 and TI2.
- Instead of working with interrupts – the incremental counter mode will be really helpful.
- Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the ARR.
- Note that the third encoder output (optional), which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Thank you for your attention

Questions?

Laboratory for NanoIntegrated Systems
Department of Electrical and Computer Engineering
MEB building – University of Utah – Salt Lake City – UT – USA