Timers

Lecture 09

Josh Brake
Harvey Mudd College

Outline

- Timer overview
- Activity: Start your own timer device driver

Learning Objectives

By the end of this lecture you should be able to…

- Explain the basic functions of a timer on the MCU
- Be able to list the steps needed to enable and use a timer

Timers on STM32L432KC

11x timers: 1x 16-bit advanced motor-control, 1x 32-bit and 2x 16-bit general purpose, 2x 16-bit basic, 2x low-power 16-bit timers (available in Stop mode), 2x watchdogs, SysTick timer

26.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 26.3.26: Timer synchronization*.

RM0394 p. 718

27.1 TIM2/TIM3 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 27.3.19: Timer synchronization*.

RM0394 p. 817

28.1 TIM15/TIM16 introduction

The TIM15/TIM16 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 28.5.21: Timer synchronization (TIM15)*.

RM0394 p. 887

29.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

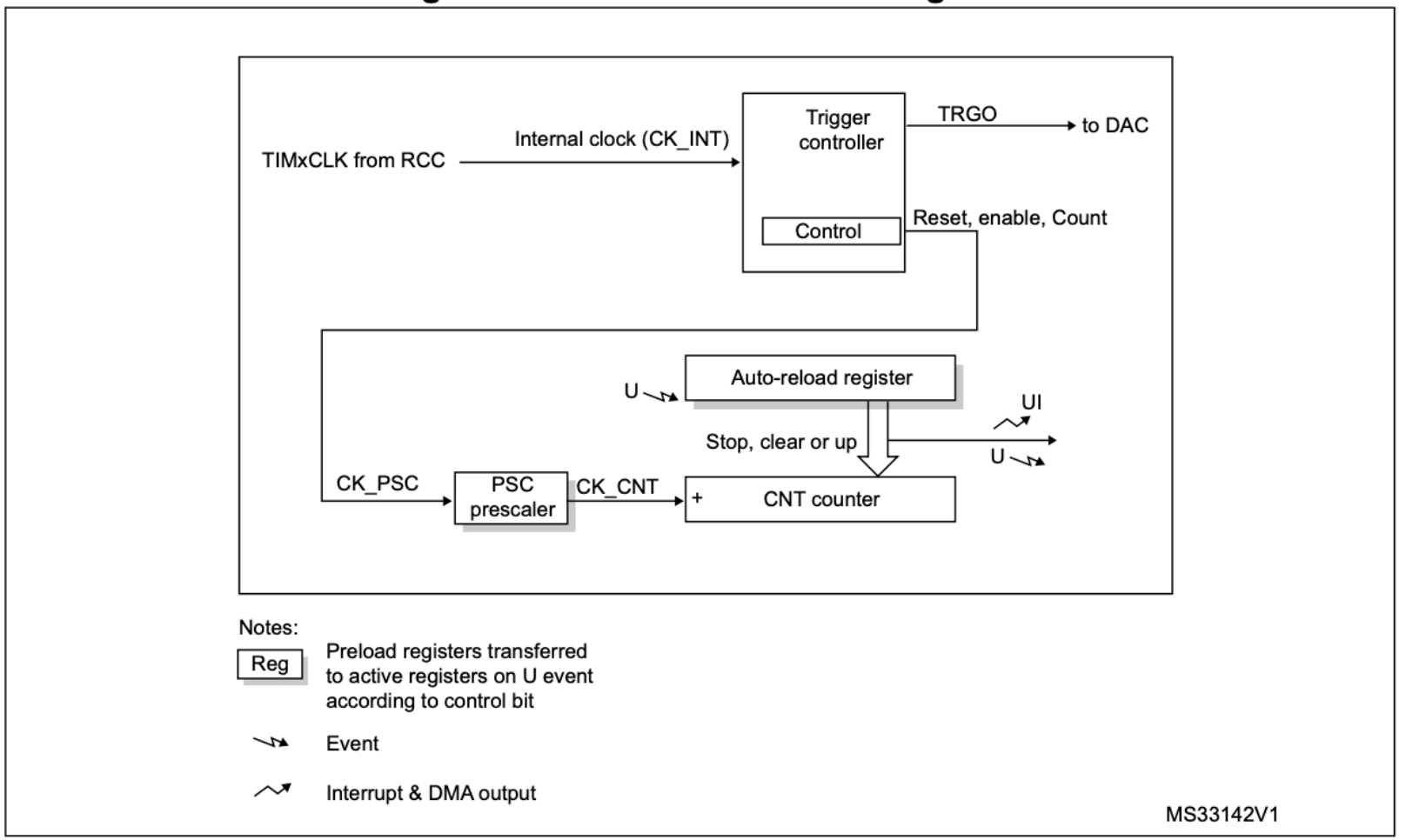
They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

The timers are completely independent, and do not share any resources.

RM0394 p. 968

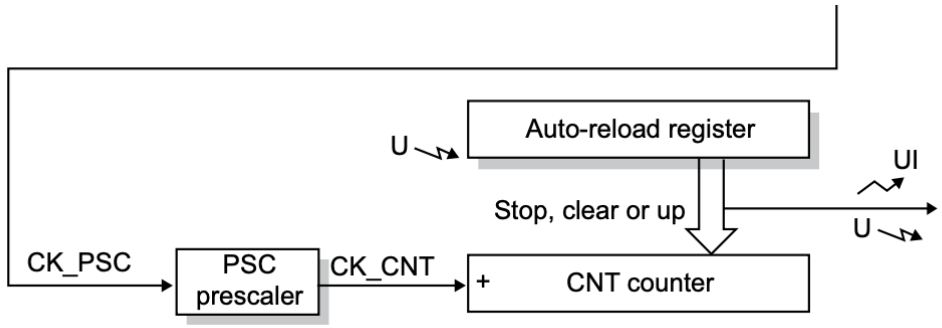
TIM6/TIM7 Block Diagram

Figure 325. Basic timer block diagram



RM0394 p. 968

Single Timer Channel



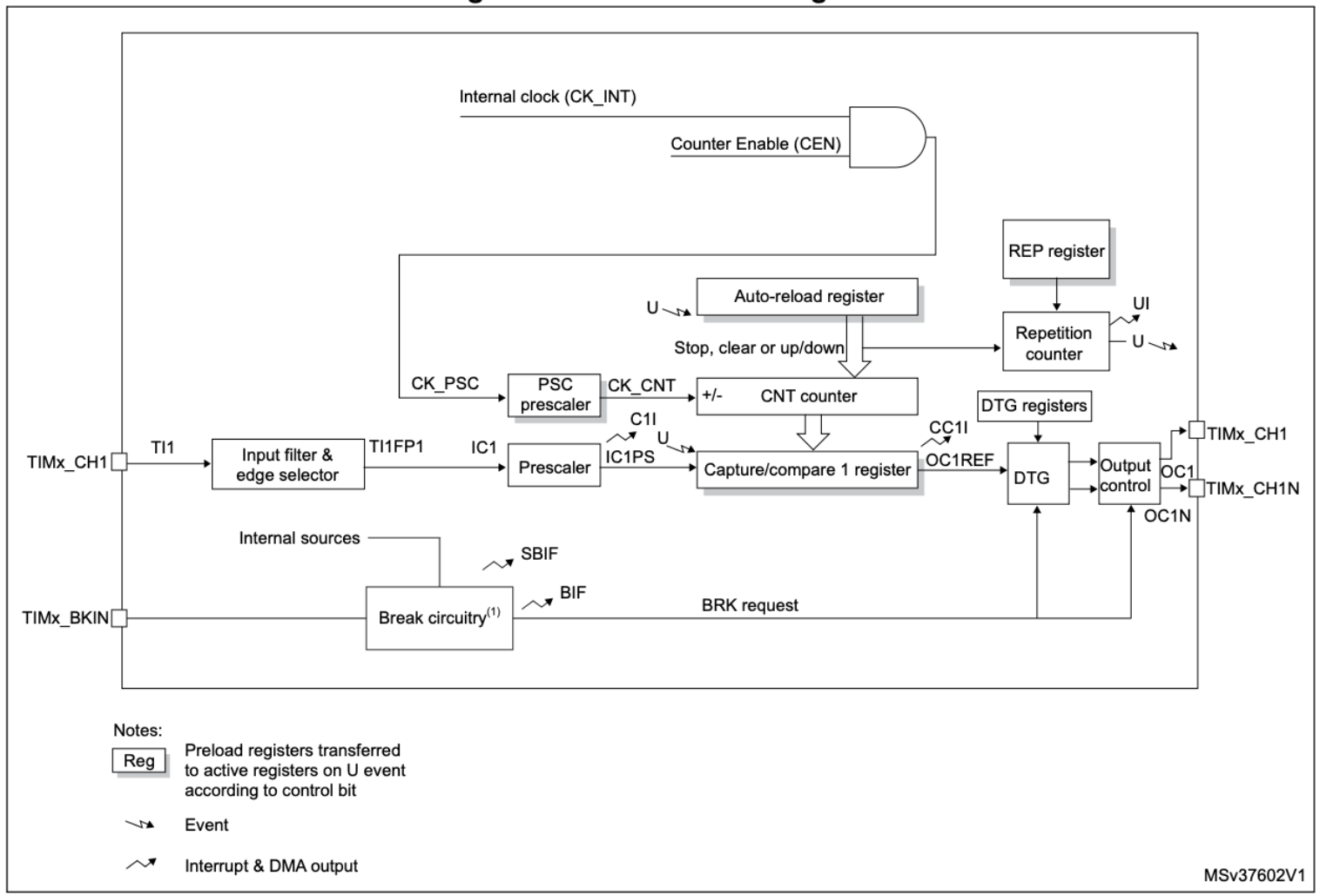
RM0394 p. 968

Main blocks

- Prescaler Register (`TIMx_PSC`): counter to pre-scale the input clock signal (`CK_PSC`)
- Counter Register (`TIMx_CNT`): Counter which holds the current count value. Counts at the rate specified by the prescaled clock (`CK_CNT`)
- Auto-Reload Register (`TIMx_ARR`): Holds the max value for the counter
- U: Event
- UI: Interrupt & DMA Output

TIM16 Block Diagram

Figure 293. TIM16 block diagram



RM0394 p. 890

Timer Register Structure

29.4.9 TIM6/TIM7 register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

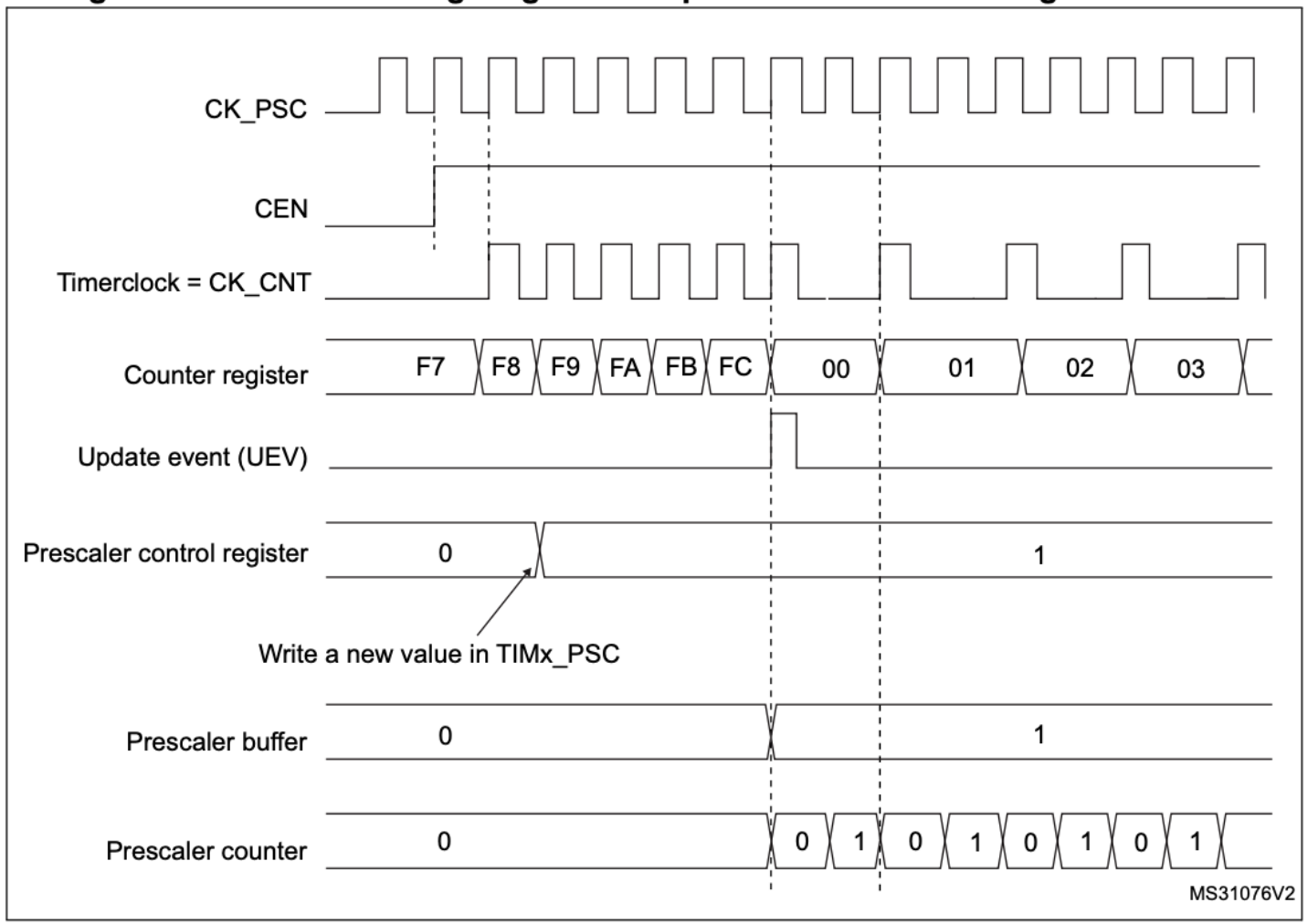
Table 144. TIM6/TIM7 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0				0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Timer Examples

Timer Example Diagrams: Prescaler

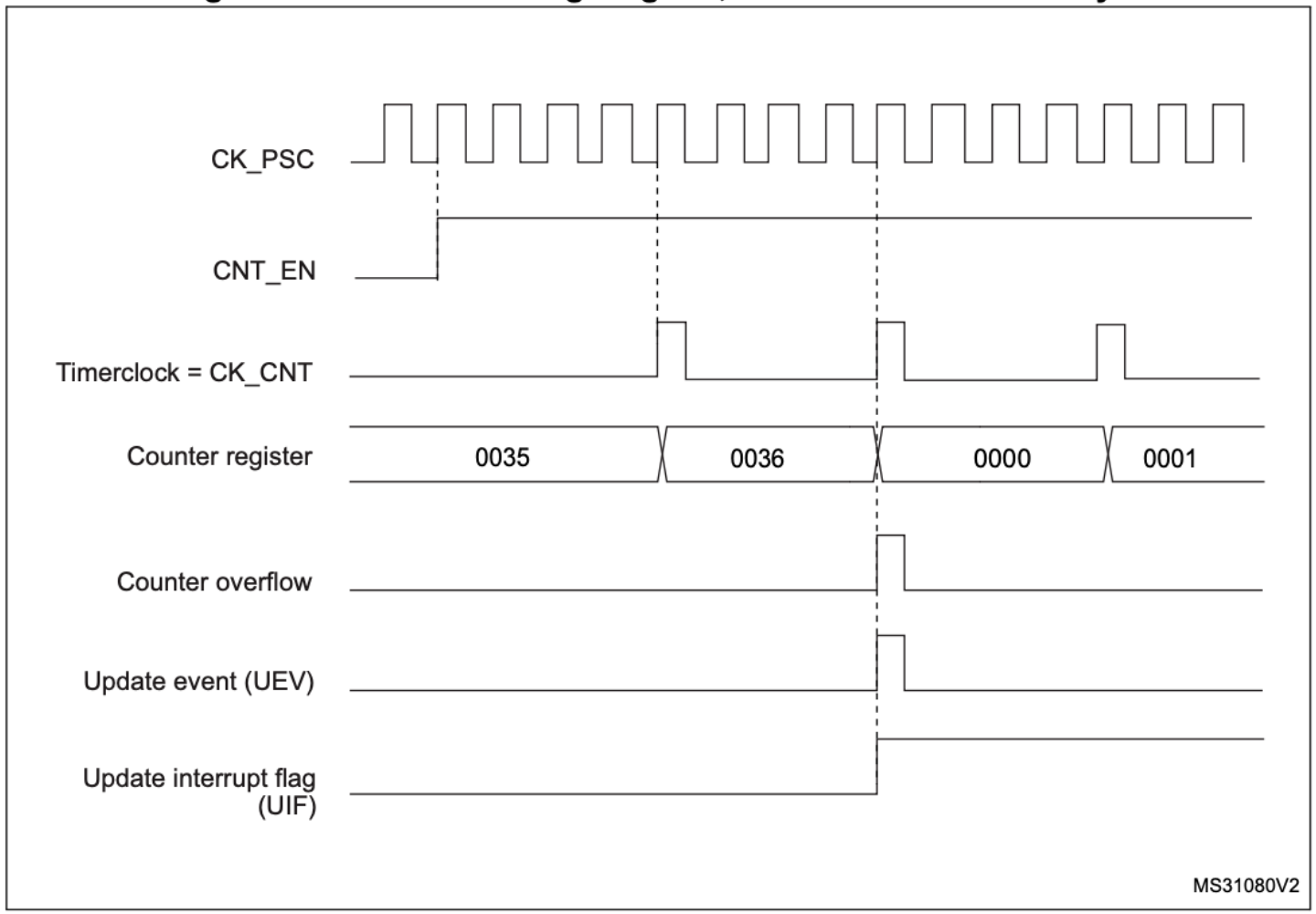
Figure 184. Counter timing diagram with prescaler division change from 1 to 2



RM0394 p. 722

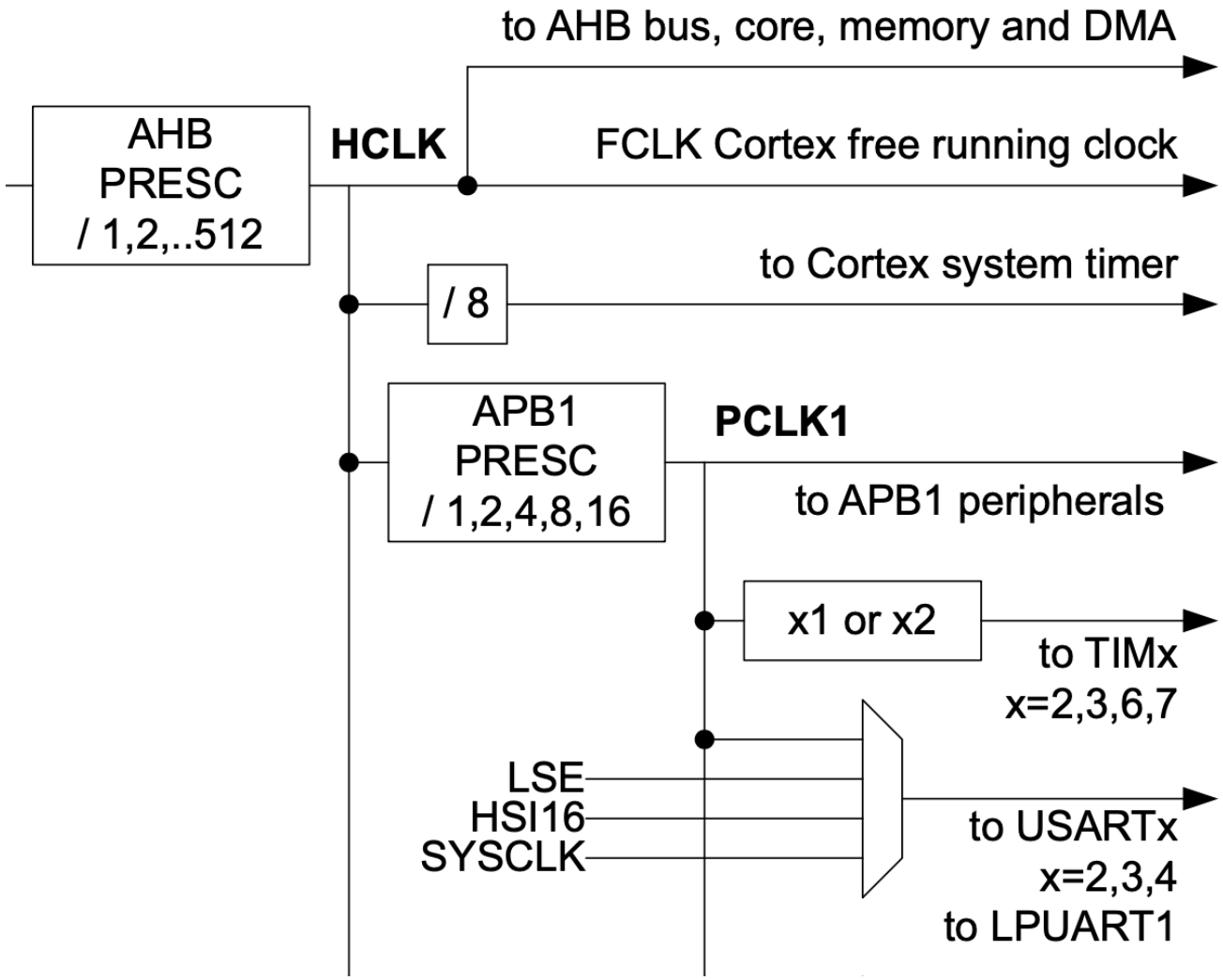
Counter: Upcounting mode

Figure 188. Counter timing diagram, internal clock divided by 4



RM0394 p. 725

Where is CK_INT coming from?



RM0394 p. 180

Activity: Timer Configuration

- Work in small groups to read the reference manual to determine how to configure a timer.
- Consider the following questions
  - What registers in RCC need to be set to enable the timer?
  - What is the base address for the timer(s) you wish to use?
  - What clock source to use?
  - How do you select it?
  - What should the prescaler be set to?
  - How does this impact the resolution and maximum delay you can measure?
  - What do the functions need to do?

Sketch out a timer device driver

Create a new device driver for the timer (e.g., `STM32L432KC_TIM.c/.h`)

Begin to fill in the needed code based on what you gathered on the last slide.

Consider the following function prototypes:

- `void initTIM(TIM_TypeDef * TIMx);`
- `void delay_millis(TIM_TypeDef * TIMx, uint32_t ms);`

Hints for Digital Audio lab

- Use timers to generate square waves of a given frequency
- Some hints on timer configuration configuration
  - Configure clocks in RCC (as shown earlier in this lecture; take note of system clock frequency)
  - Turn on clock to timer in RCC
  - Select correct clock source in TIM control (make sure slave mode is disabled)
  - Configure counter
  - Prescaler register (`TIMx_PSC`)
  - Auto-reload register (`TIMx_ARR`)
  - Enable counter `CEN` in `TIMx_CR`

Wrap Up

- Timers are widely used on an MCU to generate accurate timing.
- As a peripheral, they can be used in the background and can be used to trigger other actions on the MCU via interrupts and direct memory access (DMA).