ELEC 3040/3050
Lab #7

PWM Waveform Generation

References: STM32L1xx Technical Reference Manual
STM32L100RC Data Sheet

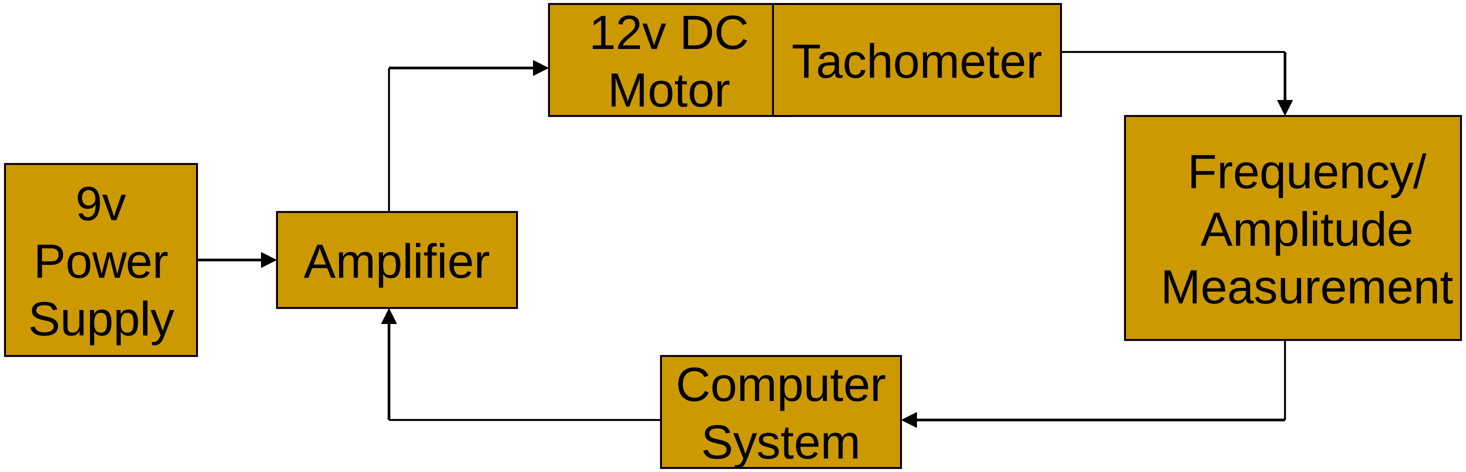
Goals of this lab exercise

- Begin the primary design project for the semester
    - Speed controller for a D.C. motor
- Generate a pulse-width-modulated (PWM) waveform with keypad-selectable duty cycle
    - Using a programmable timer

The generated waveform will be amplified in a the next lab to drive a D.C. motor

Motor Speed Control Project

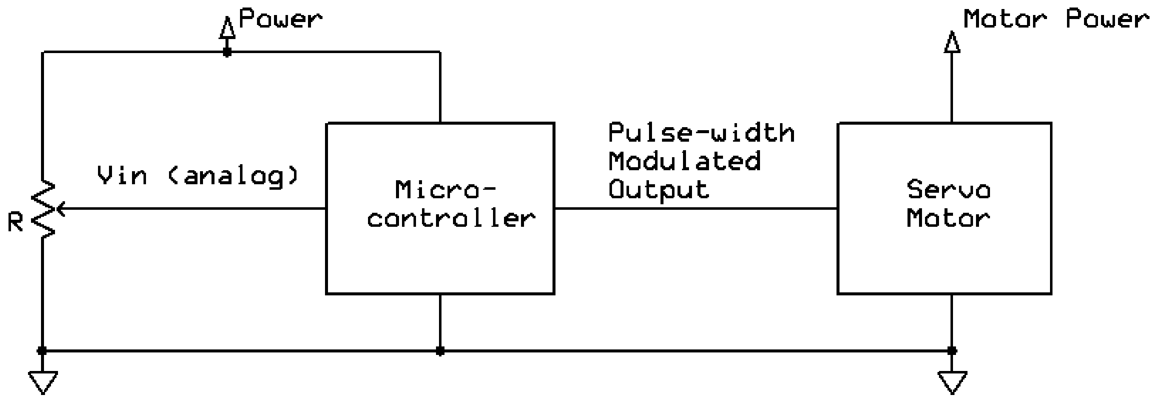
1. Generate a PWM waveform
2. Amplify the waveform to drive the motor
3. Measure motor speed
4. Measure motor parameters
5. Control speed with a PID or other controller



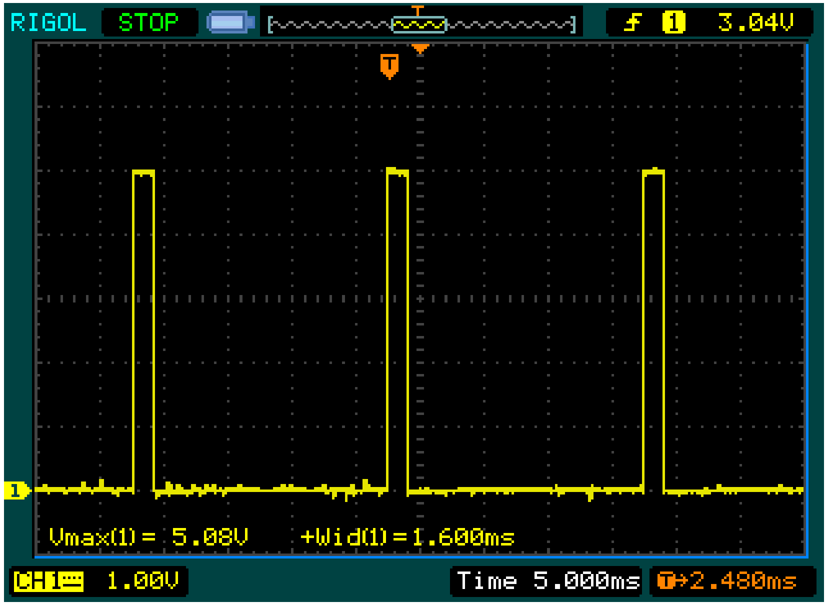
PWM Digital Waveforms

- A pulse-width modulated (PWM) waveform is a periodic signal comprising pulses of varying duration
- Modulation refers to modifying the pulse width (with period held constant) to achieve a desired effect
  - “Effect” often an average voltage to control a device
- PWM signals are often used to drive D.C. motors, commercial lights, etc.

PWM to Drive a Servo Motor



- Servo PWM signal
  - 20 ms period
  - 1 to 2 ms pulse width



PWM Waveform Parameters

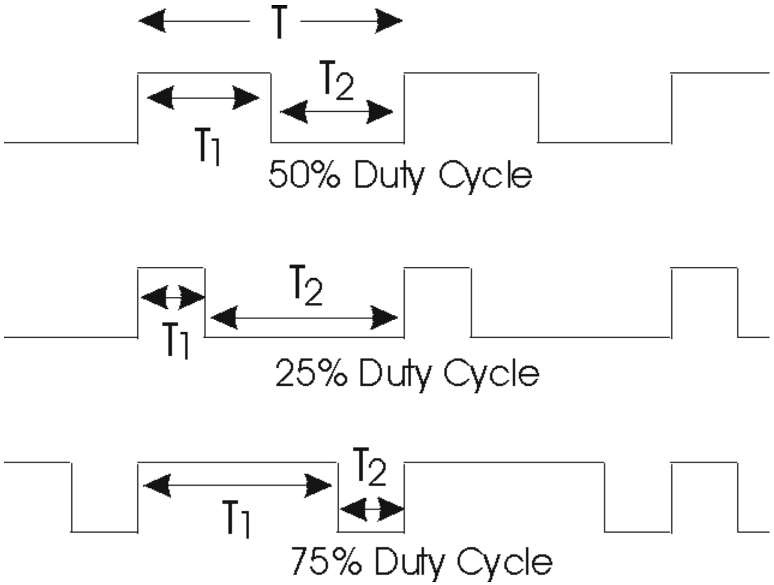
T = *period* of waveform (constant)

T1 = duration of pulse

T2 = T – T1

Duty Cycle = T1/T = T1/(T1+T2)

$V_{avg} = V_{max} \times Duty\ Cycle$



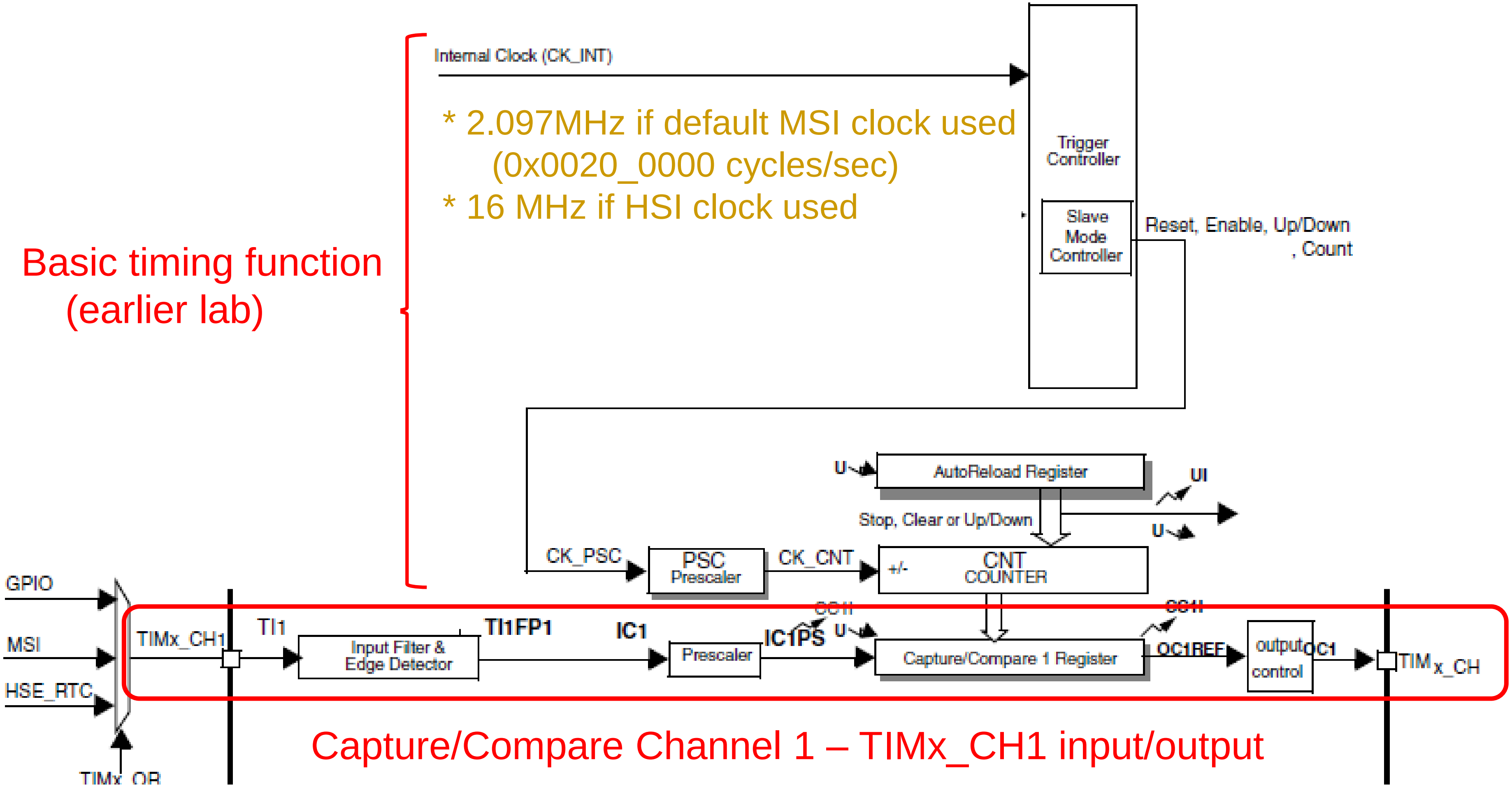
Pulses can also be active-low.

Timer operating modes

Timer capture/compare channels provide operating modes other than periodic interrupts

- **Output compare mode** – Create a signal waveform/pulse/etc.
  - Connect timer output TIMx_CHy to a GPIO pin
  - Compare CNT to value in Capture/Compare Register CCRy
  - Change output pin when CNT = CCRy
- **Pulse-Width Modulated (PWM) waveform generation mode**
  - Setup similar to output compare mode
  - Force output pin active while CNT < CCRy
  - Force output pin inactive while CCRy ≤ CNT ≤ ARR
  - ARR sets PWM period, CCRy determines PWM duty cycle
- **One pulse mode** – Create a single pulse on a pin
  - Setup similar to output compare mode
  - Disable the counter when the event occurs
- **Input capture mode** – Capture time at which an external event occurs
  - Connect a GPIO pin to timer input TIMx_CHy
  - Capture CNT value in Capture/Compare Register CCRy at time of an event on the pin
  - Use to measure time between events, tachometer signal periods, etc

General-purpose timers TIM10/TIM11



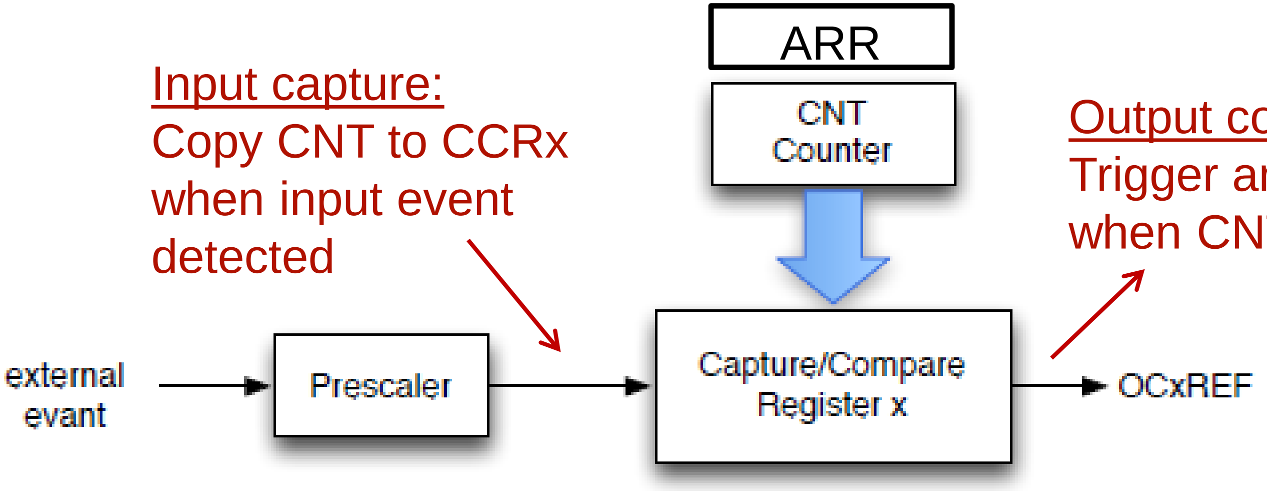
* 2.097MHz if default MSI clock used (0x0020_0000 cycles/sec)
* 16 MHz if HSI clock used

Basic timing function (earlier lab)

Capture/Compare Channel 1 – TIMx_CH1 input/output

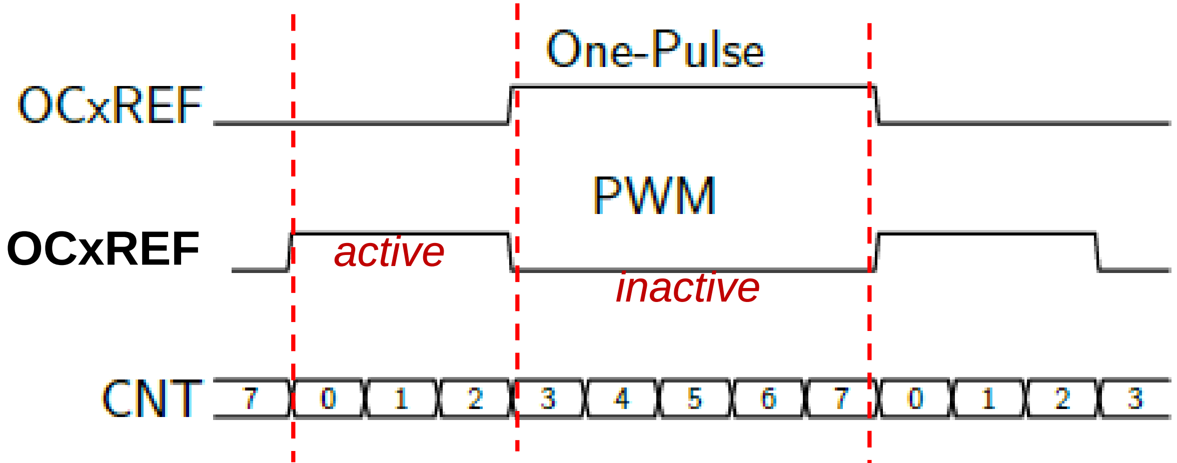
2 channels in TIM9, 4 channels in TIM2-3-4, no channels in TIM6-7
TIM6-7-10-11 have up counters, TIM2-3-4-9 have up/down counters

Timer capture/compare channels



Input capture:
Copy CNT to CCRx when input event detected

Output compare:
Trigger an event when CNT = CCRx



One-pulse

Pulse-width modulation

active

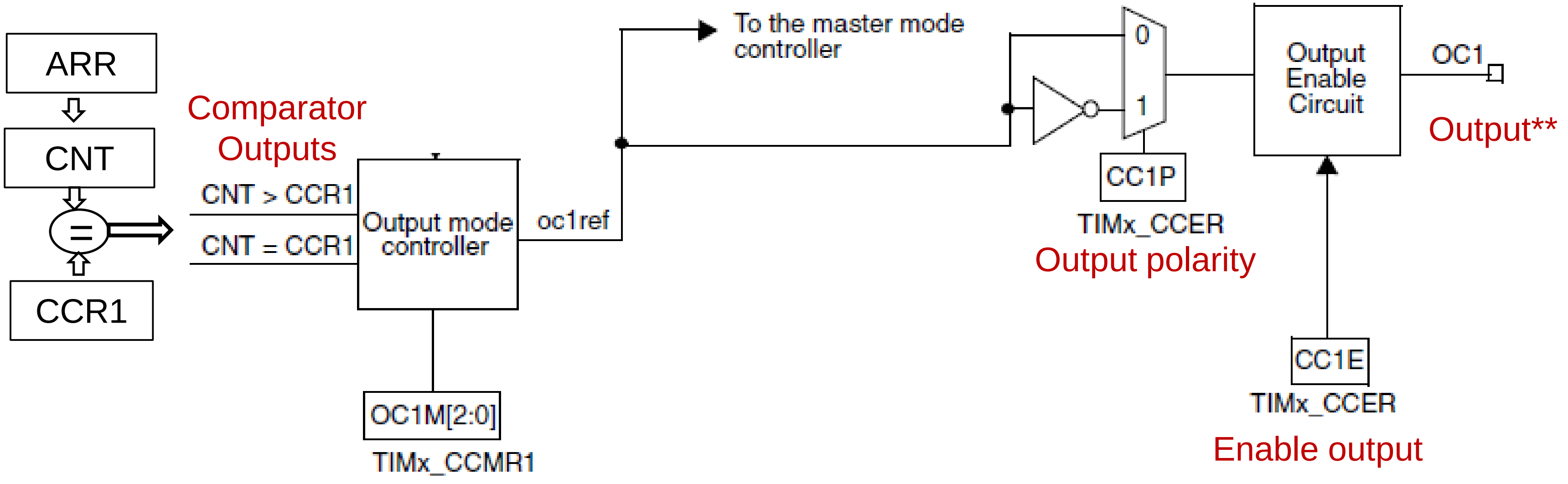
inactive

Period Start

CNT=CCRx=3 (toggle OCxREF)

CNT=ARR=7 (reset CNT and OCxREF)

Capture/Compare Output Stage



Comparator Outputs

Output polarity

Output**

Enable output

Mode = Output Compare or PWM

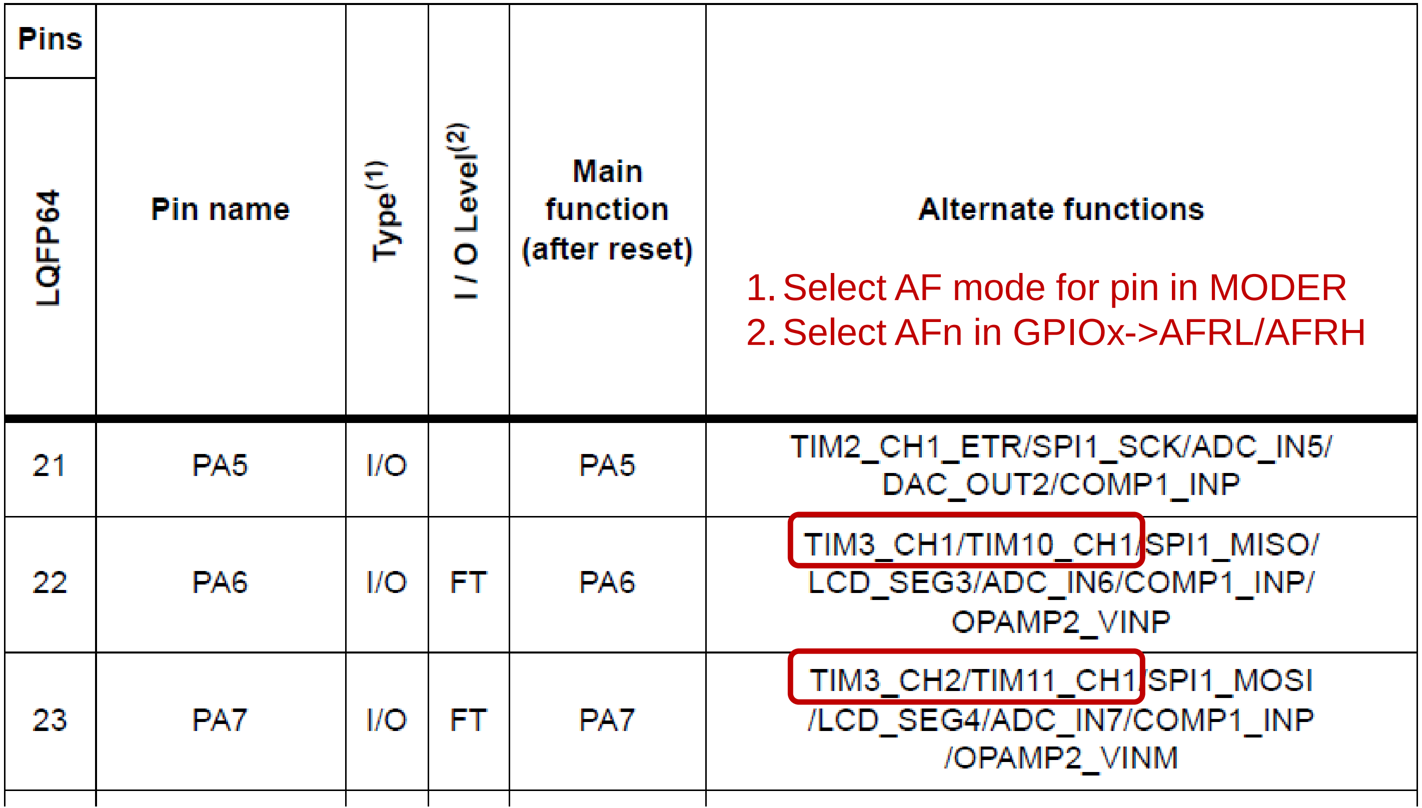
** Route output OC1 to a GPIO pin as an “alternate function”.
(each GPIO pin can connect to one or two timer channels)

Timer outputs as GPIO pin alternate functions

Each GPIO pin configurable as: INPUT, OUTPUT, ANALOG, ALTERNATE FUNCTION

- Select pin modes in **GPIOx->MODER** *(10 = alternate function)*

From STM32L100RX Data Sheet Table 7. "Pin Definitions" (partial)



Pins	Pin name	Type(1)	I / O Level(2)	Main function (after reset)	Alternate functions
LQFP64					
21	PA5	I/O		PA5	TIM2_CH1_ETR/SPI1_SCK/ADC_IN5/ DAC_OUT2/COMP1_INP
22	PA6	I/O	FT	PA6	TIM3_CH1/TIM10_CH1/SPI1_MISO/ LCD_SEG3/ADC_IN6/COMP1_INP/ OPAMP2_VINP
23	PA7	I/O	FT	PA7	TIM3_CH2/TIM11_CH1/SPI1_MOSI /LCD_SEG4/ADC_IN7/COMP1_INP /OPAMP2_VINM

1. Select AF mode for pin in MODER
2. Select AFn in GPIOx->AFRL/AFRH

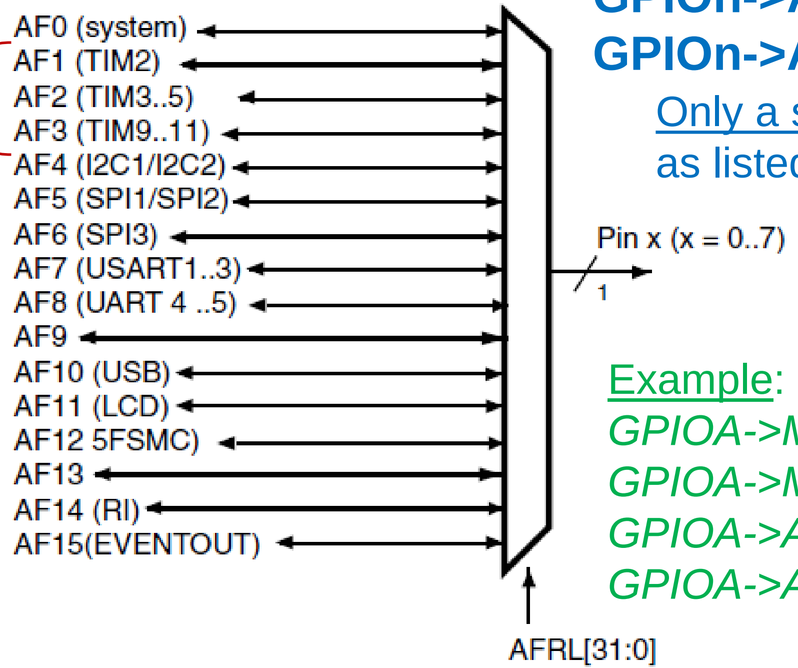
We will use TIM10_CH1 (Pin PA6)

Selecting an alternate function

GPIOn->MODER selects AF mode for pins (10)
GPIOn->AFR[0] selects AFs for pins Pn0-Pn7
GPIOn->AFR[1] selects AFs for pins Pn8-Pn15

Only a subset of AF's available at each pin, as listed in data sheet. (see previous slide)

Timers

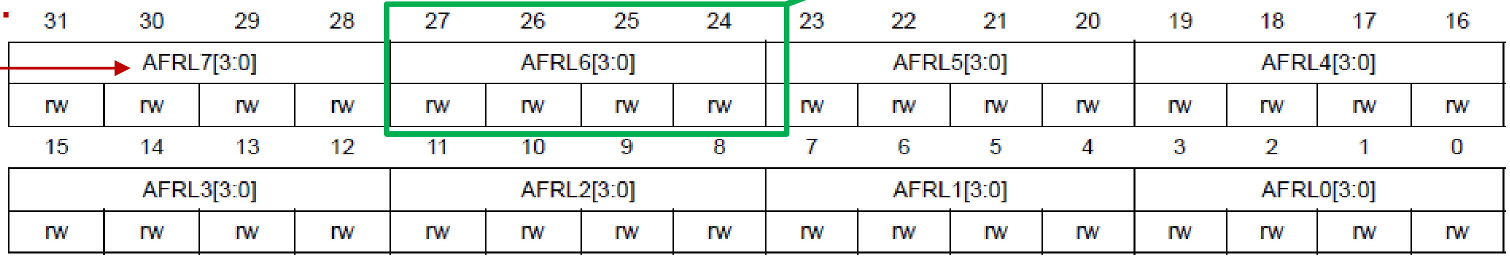


Example: Configure PA6 as TIM3_CH1 (AF2)

```
GPIOA->MODER &= ~0x00003000;  //clear PA6 mode
GPIOA->MODER  |=   0x00002000;  //PA6 = AF mode
GPIOA->AFR[0]   &= ~0x0F000000;  //clear AFRL6
GPIOA->AFR[0]    |=    0x02000000;  //PA6 = AF2
```

AFR[0]:

AFRLn defines pin n, n=0..7

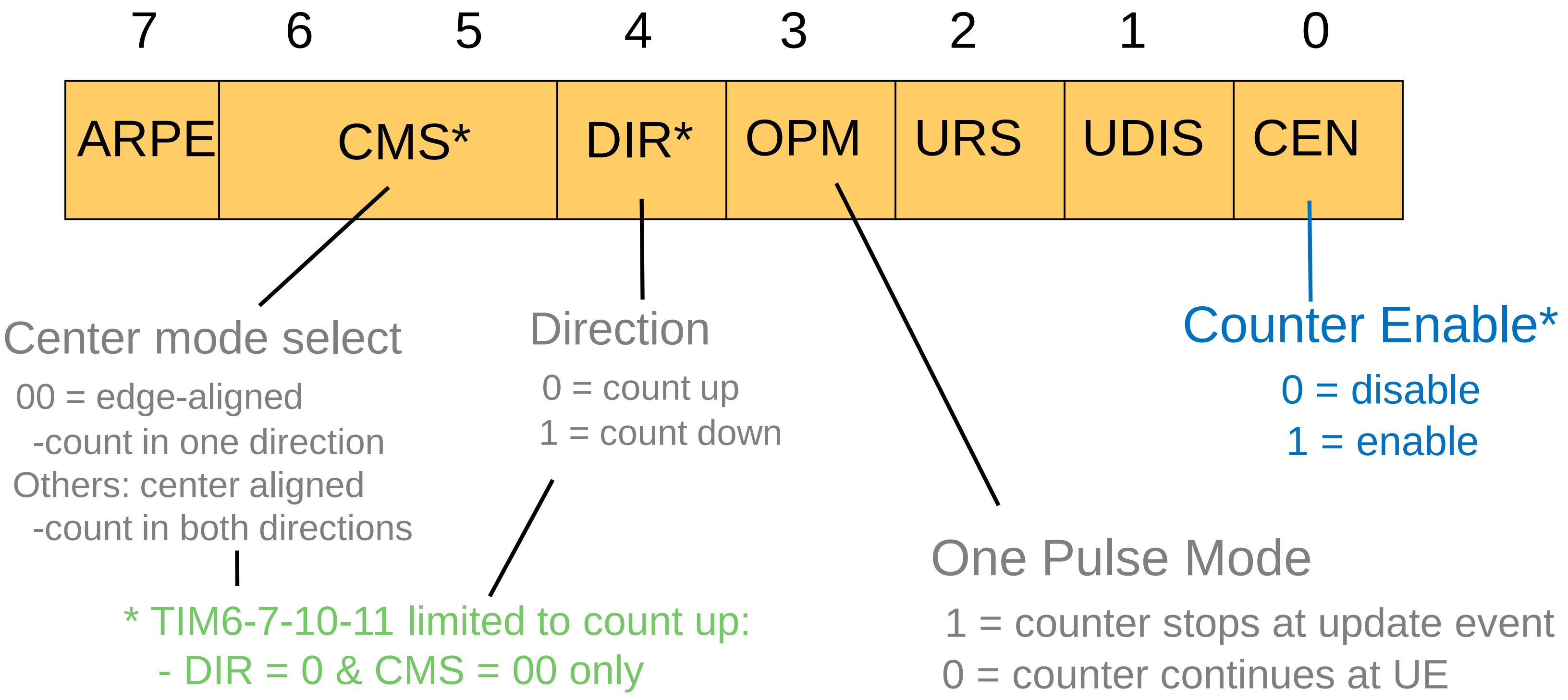


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Timer System Control Register 1

See timer overview from earlier lab

TIMx_CR1 (reset value = all 0’s)



7	6	5	4	3	2	1	0
ARPE	CMS*		DIR*	OPM	URS	UDIS	CEN

Center mode select

00 = edge-aligned
-count in one direction
Others: center aligned
-count in both directions

Direction

0 = count up
1 = count down

Counter Enable*

0 = disable
1 = enable

* TIM6-7-10-11 limited to count up:
- DIR = 0 & CMS = 00 only

One Pulse Mode

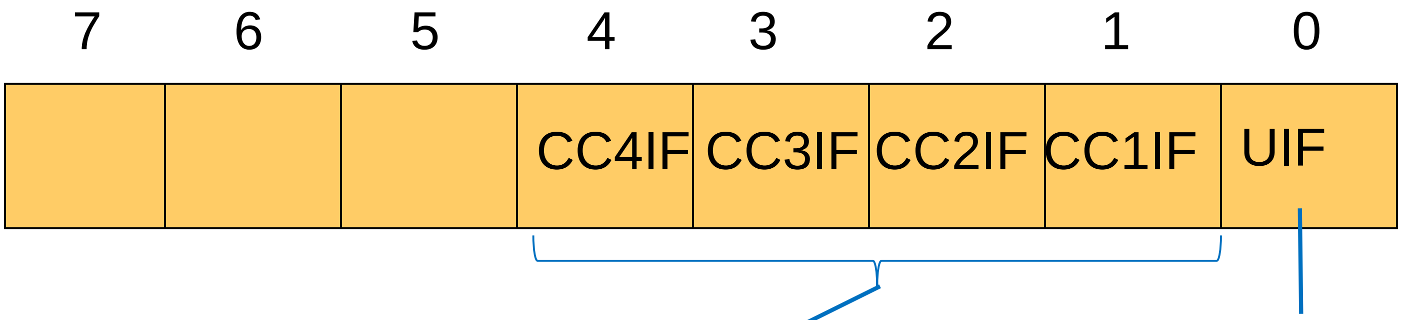
1 = counter stops at update event
0 = counter continues at UE

*CEN only bit that needs to be changed for simple PWM

Timer Status Register

See timer overview from earlier lab

TIMx_SR (reset value = all 0’s)



7	6	5	4	3	2	1	0
			CC4IF	CC3IF	CC2IF	CC1IF	UIF

Capture/compare interrupt flags

1 = capture/compare interrupt pending

0 = no capture/compare event occurred

Set by hardware on capture/comp event

Cleared by software

(reset CCxIF bit to 0)

Update interrupt flag

1 = update interrupt pending

0 = no update occurred

Set by hardware on update event

Cleared by software

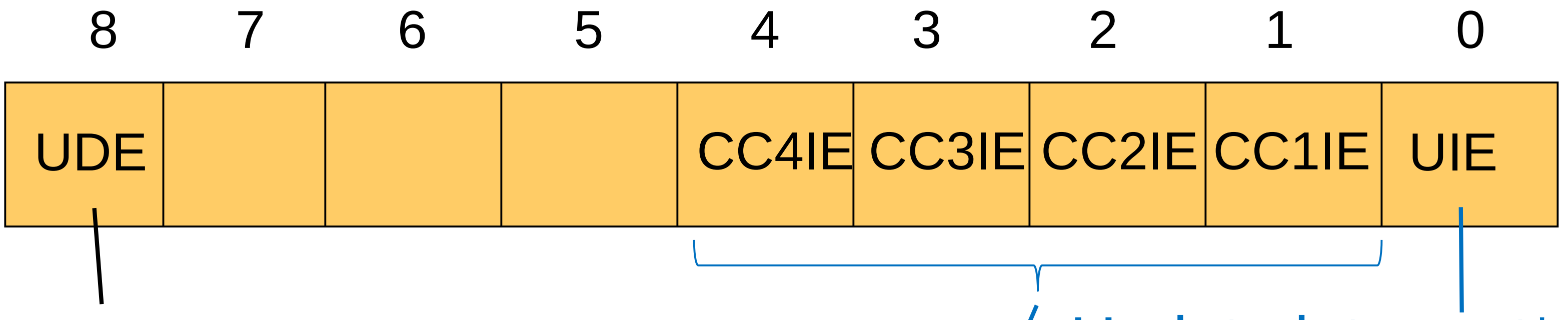
(reset UIF bit to 0)

TIM10 has only CC1IF

Timer DMA/Interrupt Enable Register

See timer overview from earlier lab

TIMx_DIER (reset value = all 0’s)



8	7	6	5	4	3	2	1	0
UDE				CC4IE	CC3IE	CC2IE	CC1IE	UIE

Update DMA request enable
1 = enable, 0 = disable

Update interrupt* enable
1 = enable, 0 = disable

Capture/Compare interrupt* enable
TIMx interrupt on capture/compare event
1 = CCx interrupt enabled, 0 = disabled

TIM10 has only CC1IE

* Capture/compare and update events generate the **same IRQn signal**, and use the **same interrupt handler**. Handler reads status register flags to determine source.

Capture/Compare Register

- Compared to TIMx_CNT to trigger operations at specified times.
- **TIMx_CCRy** = TIMx capture/compare register, channel y
  - TIM2-3-4: y=1,2,3,4;   TIM9: y = 1,2;   TIM10-11: y=1
  - CCRy register width same as CNT/ARR registers (16 bits)

- **Input capture mode**: TIMx_CNT captured in TIMx_CCRy when a designated input signal event is detected
- **Output compare mode**: TIMx_CCRy compared to TIMx_CNT; each match is signaled on OCy output
- **One pulse mode**: same as output compare, but disable after match
- **PWM mode:** TIMx_CCRy compared to TIMx_CNT
  - CNT < CCRy   => output active
  - CNT ≥ CCRy   => output inactive

TIMx_CNT operates as discussed previously for periodic interrupt generation:
  - Signal update event and reset to 0 when CNT = ARR while counting up
  - Signal update event and reload ARR when CNT = 0 while counting down

Capture/Compare Mode Registers

TIMx_CCMR1: bits 7:0 configure channel 1; bits 15:8/channel 2
TIMx_CCMR2 (TIM2-3-4): bits 7:0/channel 3; bits 15:8/channel 4
(reset values = all 0's)

	7	6	5	4	3	2	1	0
Output mode ->	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
Input mode** ->	IC1F[3:0]				IC1PSC[1:0]			
	rw	rw	rw	rw	rw	rw	rw	rw

** discussed later

Output Compare 1 Mode

000 = frozen (no events)
001 = Set CH1 active* on match
010 = Set CH1 inactive* on match
011 = Toggle CH1 on match
100 = Force CH1 to inactive* (immediate)
101 = Force CH1 to active* (immediate)
110 = PWM mode 1 (active* to inactive*)
111 = PWM mode 2 (inactive* to active*)

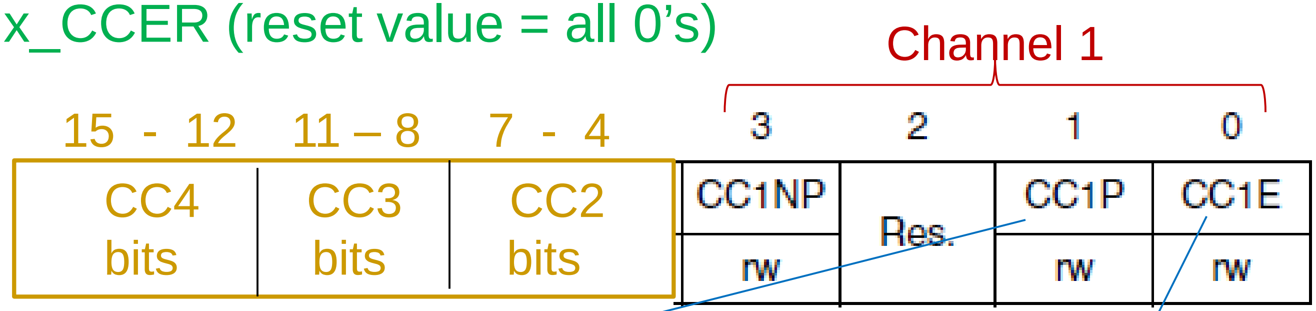
Capture/Compare 1 Select

00 = output
01 = input**: IC1 = TI1
10 = input**: IC1 = TI2
11 = input**: IC1 = TRC

* Active/inactive levels selected in TIMx_CCER register

Capture/Compare Enable Register

TIMx_CCER (reset value = all 0’s)



Channel 1

15 - 12	11 – 8	7 - 4	3	2	1	0
CC4 bits	CC3 bits	CC2 bits	CC1NP	Res.	CC1P	CC1E
			rw		rw	rw

CC1 Polarity
If CC1 = output, CC1P selects:
  0 = OC1 active high
  1 = OC1 active low

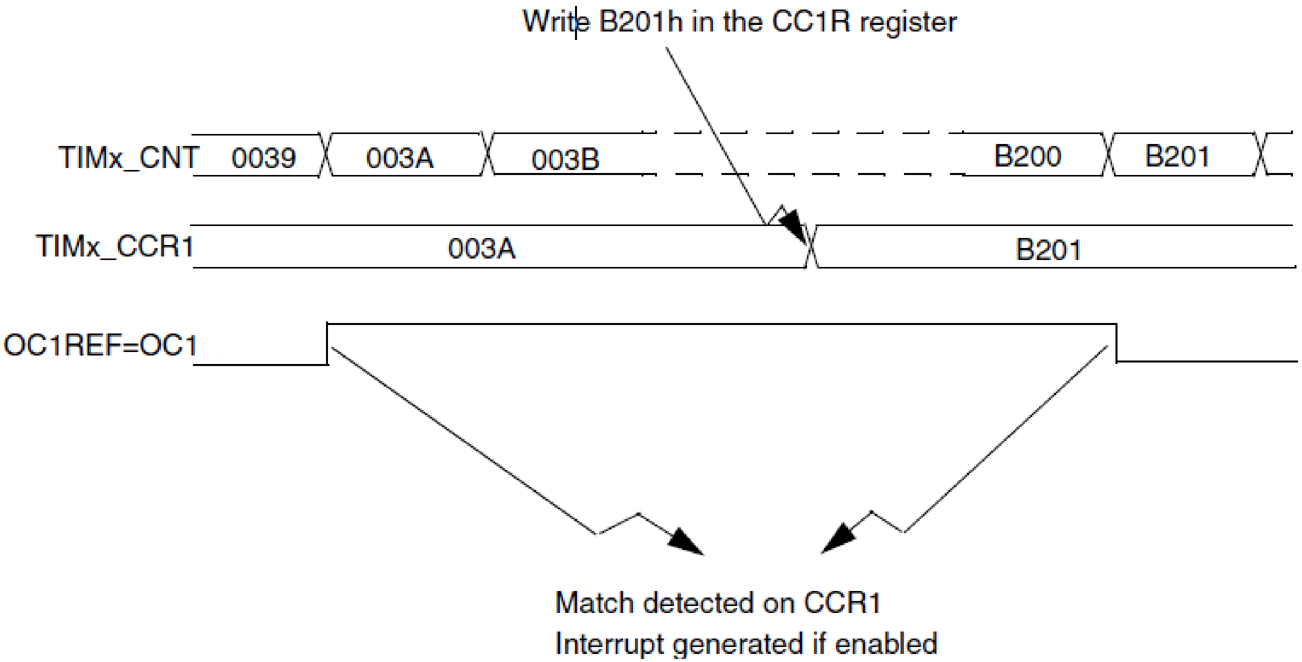
If CC1 = input:
CC1NP/CC1P select capture trigger:
  00: falling edge of input
  01: rising edge of input
  11: both edges of input

CC1 Enable
If CC1 = output:
  1 = OC1 drives output pin
  0 = OC1 does not drive output

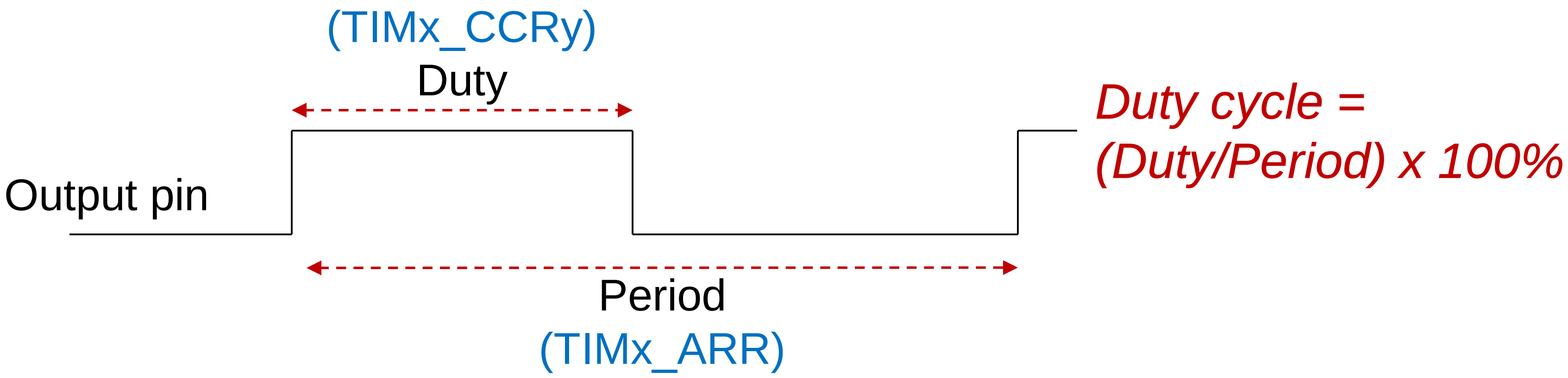
If CC1 = input:
  1 = Capture enabled
  0 = Capture disabled

Output Compare Mode

- Change output pin state or indicate when a period of time has elapsed
- When a match occurs (CCRx = CNT):
  - Generate specified output on corresponding pin
  - Set CCxIF = 1 (interrupt flag) in the SR
  - Generate interrupt if configured (CCxIE = 1)



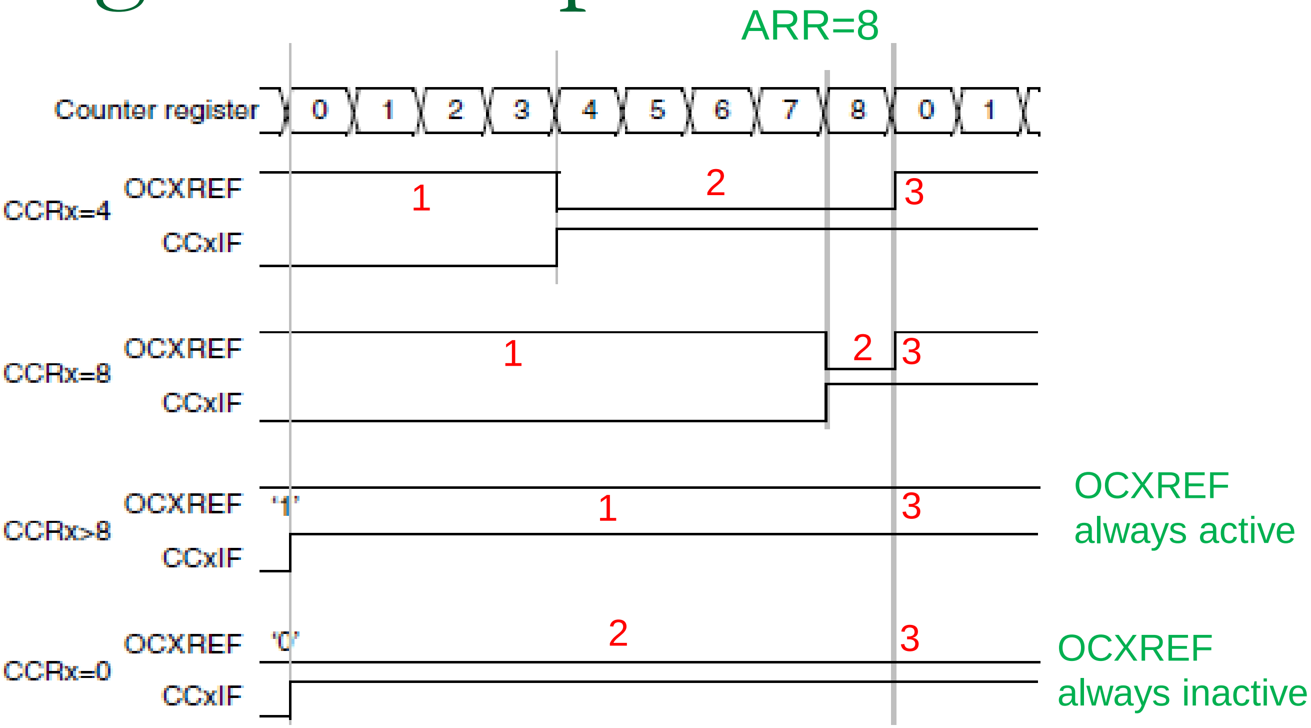
Pulse-Width Modulation (PWM) Mode



Duty cycle = (Duty/Period) x 100%

- **PWM** by comparing **TIMx_CNT** to both **TIMx_CCRy** and **TIMx_ARR**
  - Set **TIMx_ARR** = Period
  - Set **TIMx_CCRy** = Duty
- **TIMx_CCMRn** (capture/compare mode)
  - Set bit CCxE = 1 to configure the channel as output
  - Set bits OCxM = 110 (PWM mode 1) – active if CNT < CCRy, inactive otherwise
    OCxM = 111 (PWM Mode 2) - inactive if CNT < CCRy , active otherwise
- **TIMx_CCER**:
  - Set bit CCxP = 0/1 to select active level high/low (output polarity) of OCx
  - Set bit CCxE = 1 to enable OCx to drive the output pin
- Configure GPIO **MODER** and **AF** registers to select alt. function TIMx_CHn for the pin

PWM Signal Examples



1. OCXREF active (high) when TIMx_CNT < TIMx_CCRx

   Assumes OCxM = 110 and CCxP = 1

2. OCXREF inactive (low) when TIMx_CNT ≥ TIMx_CCRx
3. Update Event when TIMx_CNT = TIMx_ARR (resets TIMx_CNT to 0)

Example:
20KHz PWM signal with 10% duty cycle on pin PB6

- Configure TIM4, Channel 1
  - Since TIM4_CH1 = AF2 for pin PB6
- Assume timer clock = 16MHz* and prescale = 1
  - PWM Period = 16MHz/20KHz = 800 = TIM4_ARR
  - PWM Duty = 800 x 10% = 80 = TIM4_CCR1
- Configure TIM4_CCMR1 bits:
  - CC1E = 0 (make channel 1 an output)
  - CC1M = 110 (PWM mode 1: active-to-inactive)
- Configure TIM4_CCER bits:
  - CC1P = 0 to define OC1 as active high
  - CC1E = 1 to enable output OC1 to drive the pin
- Configure PB6 as alternate function TIM4_CH1
  - Select AF mode for PB6 in GPIOB->MODER
  - Select TIM4_CH1 (AF2) for PB6 in GPIOB->AFRL

** What if timer clock*
= 2.097 MHz ?
(0x0020_0000 Hz)

Lab Procedure

- Generate a PWM waveform with timer TIM10
  - Period should be 1 ms (frequency 1 KHz)
  - First, generate a waveform with one duty cycle value
  - Then, verify that you can generate waveforms with each of the 11 specified duty cycles, from 0% to 100%, as selected by keypad keys 0 – A.
    - Measure and record the 11 duty cycle values
    - Plot measured duty cycle vs. selection key #

- Repeat with PWM frequency = 100 Hz
  - What needs to be changed?
- *(Time permitting)* Repeat with PWM frequency = 10 KHz