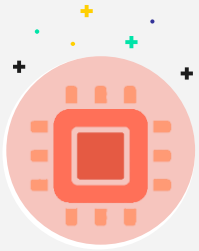
1402/11/22



Microcontrollers

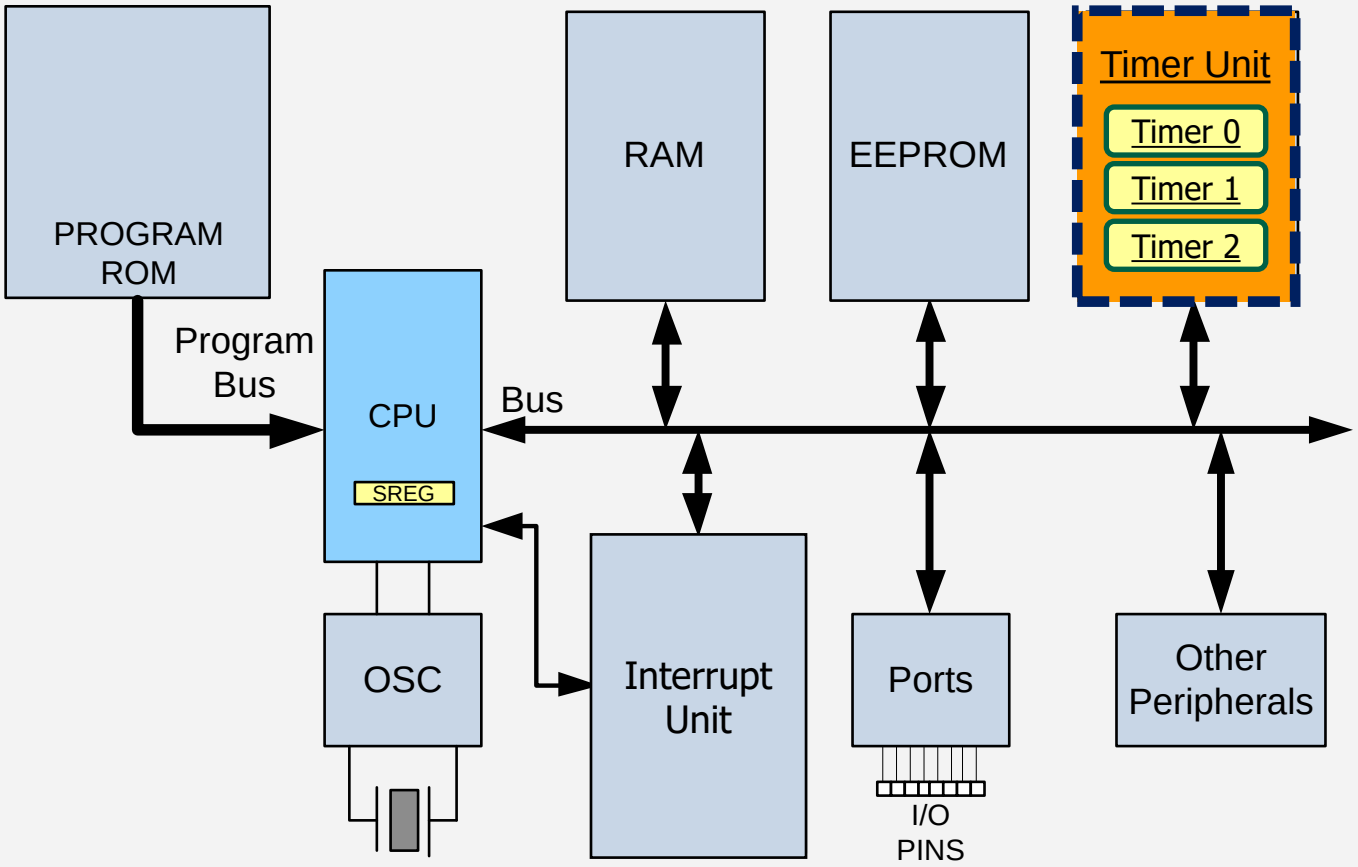
Lecture 5:

Timer/counter (Normal,CTC Modes)

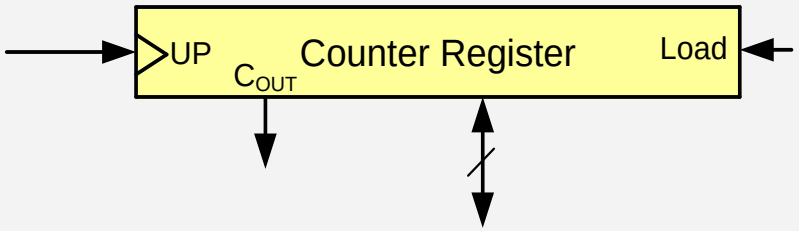
By: M.Razeghizadeh

Start now!

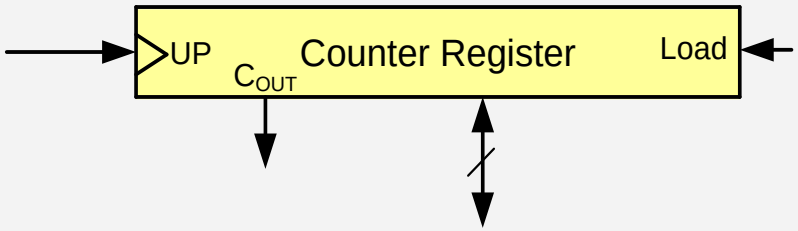
Overview


PROGRAM ROM
Program Bus
CPU
SREG
OSC
RAM
EEPROM
Timer Unit
Timer 0
Timer 1
Timer 2
Bus
Interrupt Unit
Ports
I/O PINS
Other Peripherals

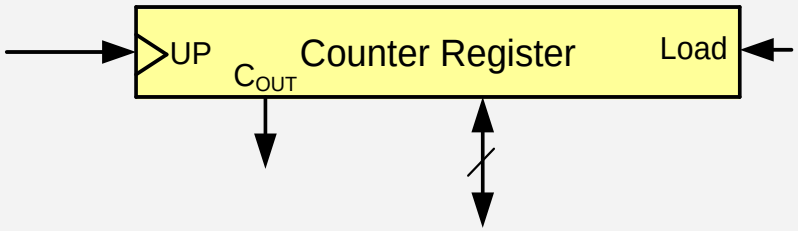
A counter register



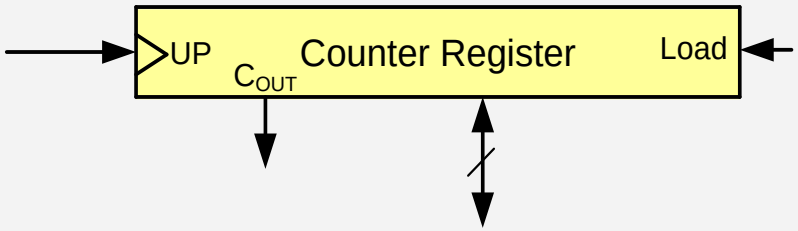
A counter register



A counter register

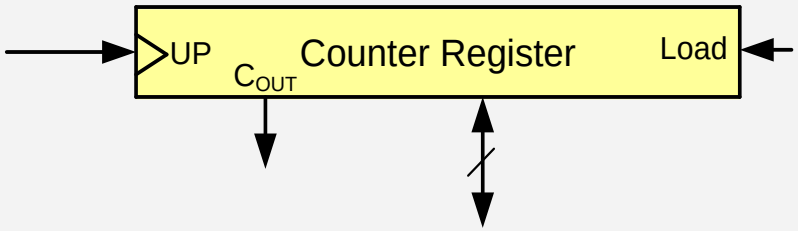


A counter register

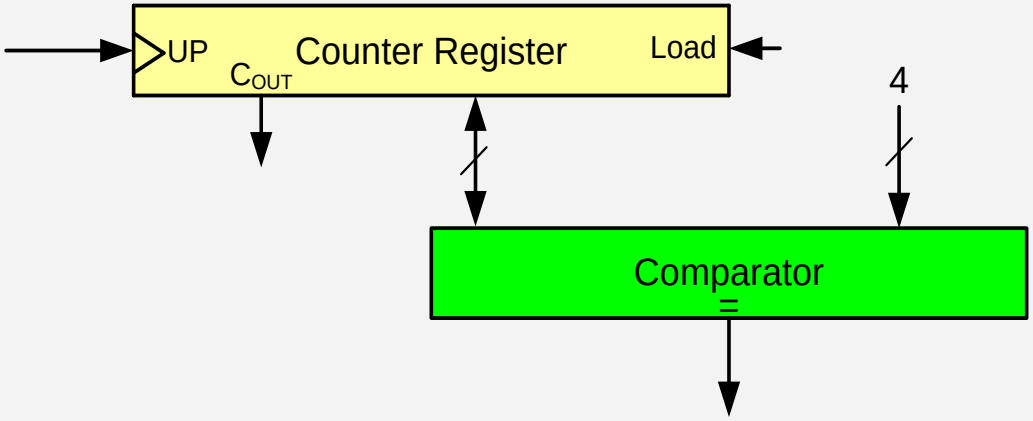


A counter register - First design

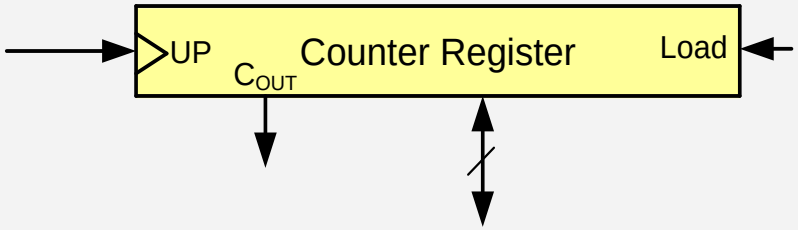
A counter register



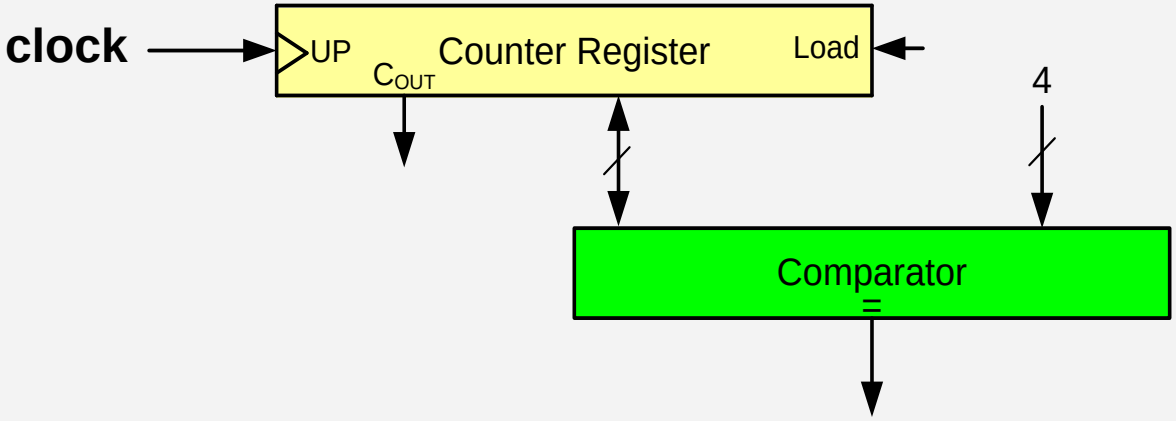
A counter register - First design



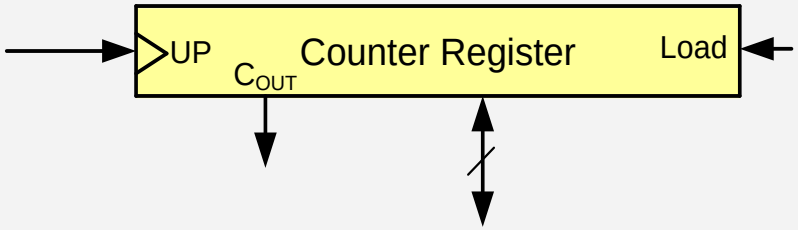
A counter register



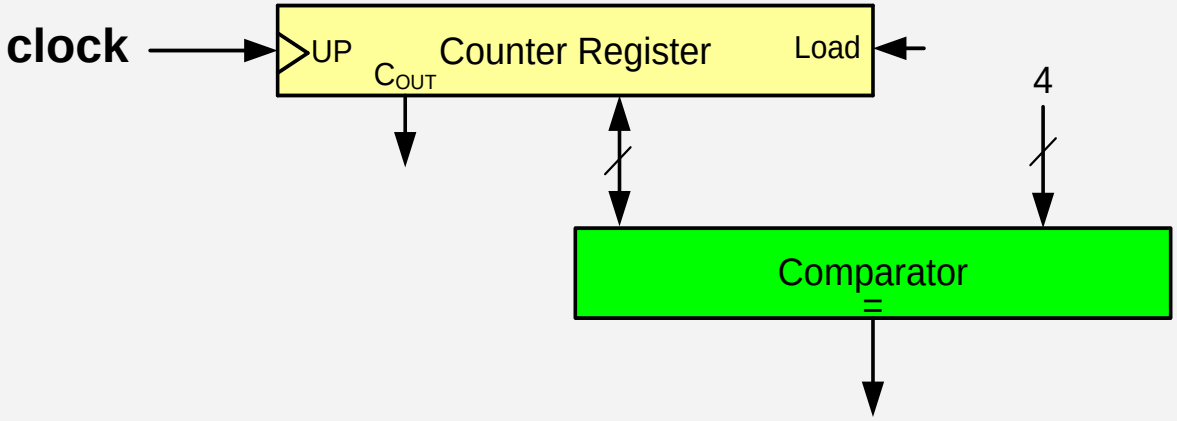
A counter register - First design



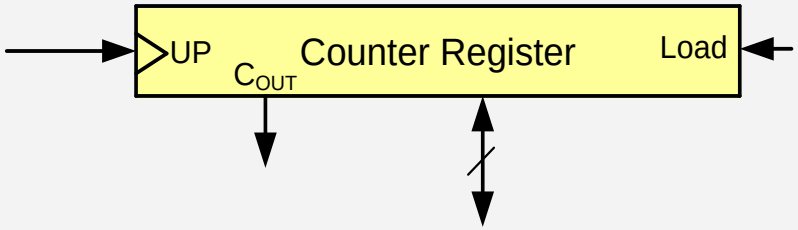
A counter register



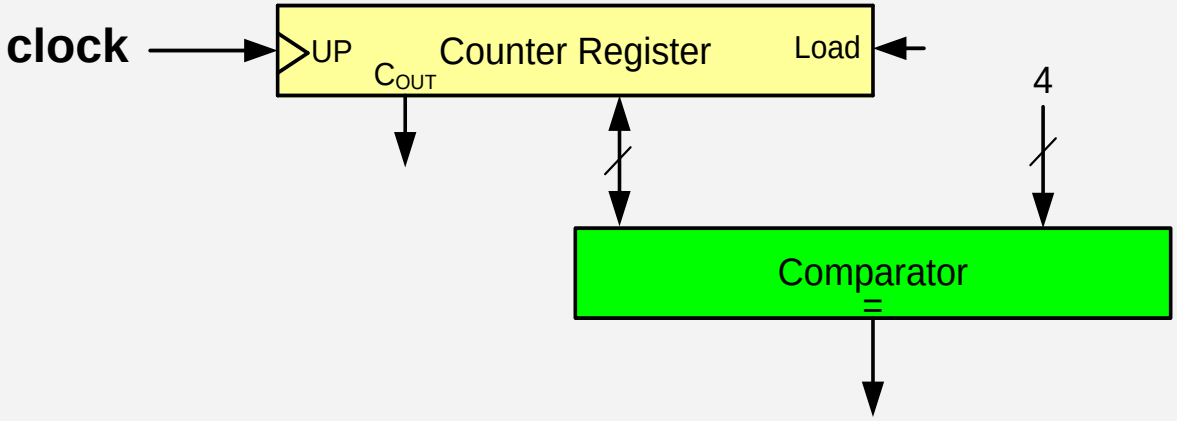
A counter register - First design



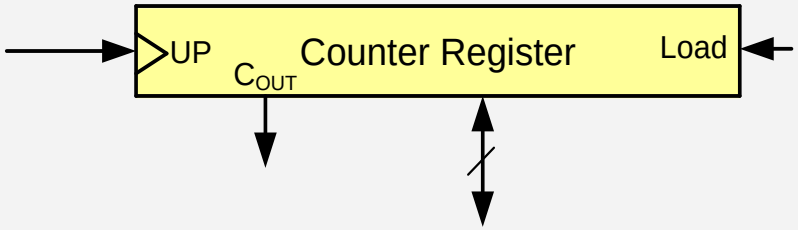
A counter register



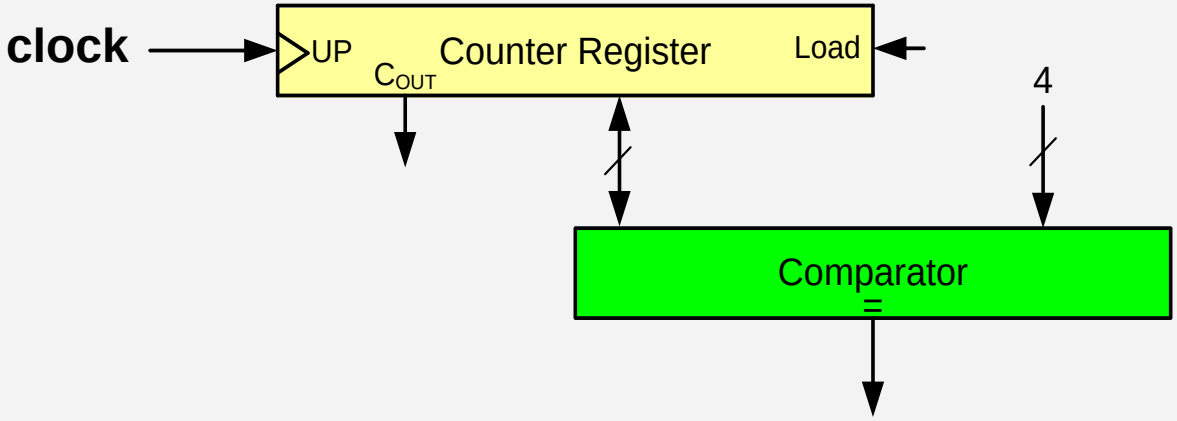
A counter register - First design



A counter register

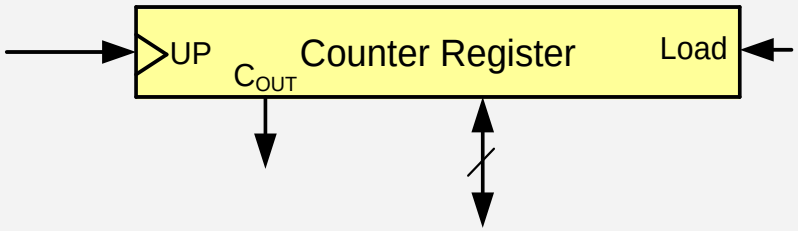


A counter register - First design

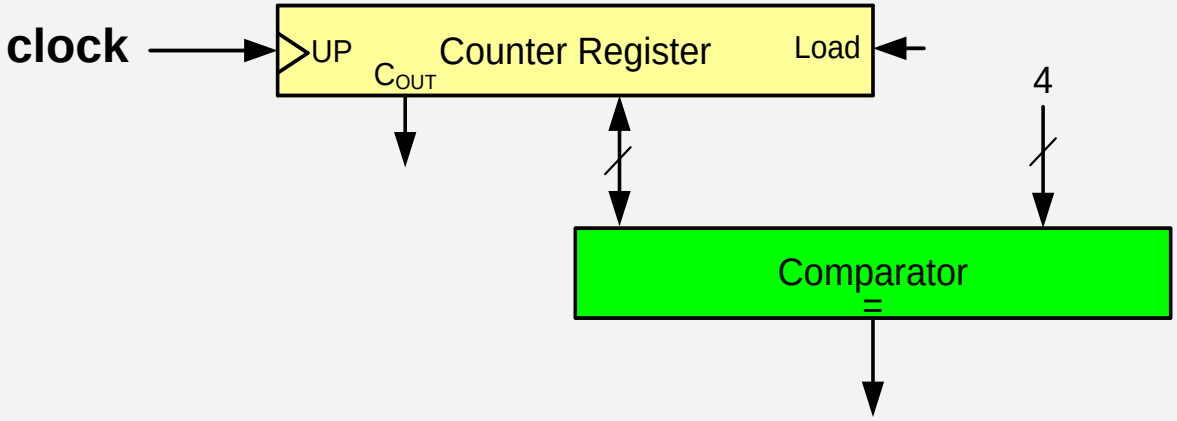


A counter register - Second design

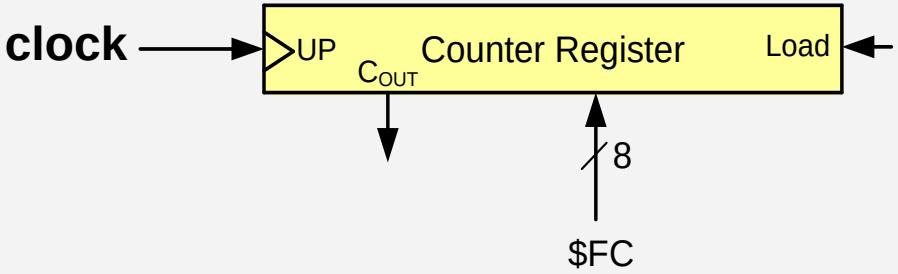
A counter register



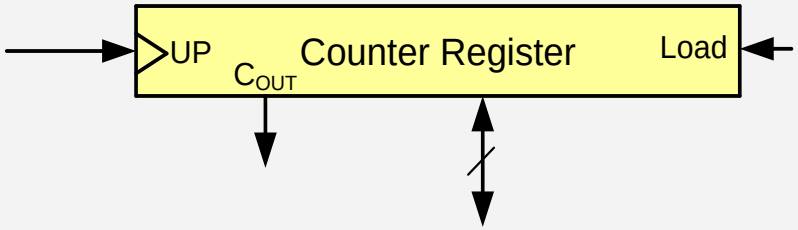
A counter register - First design



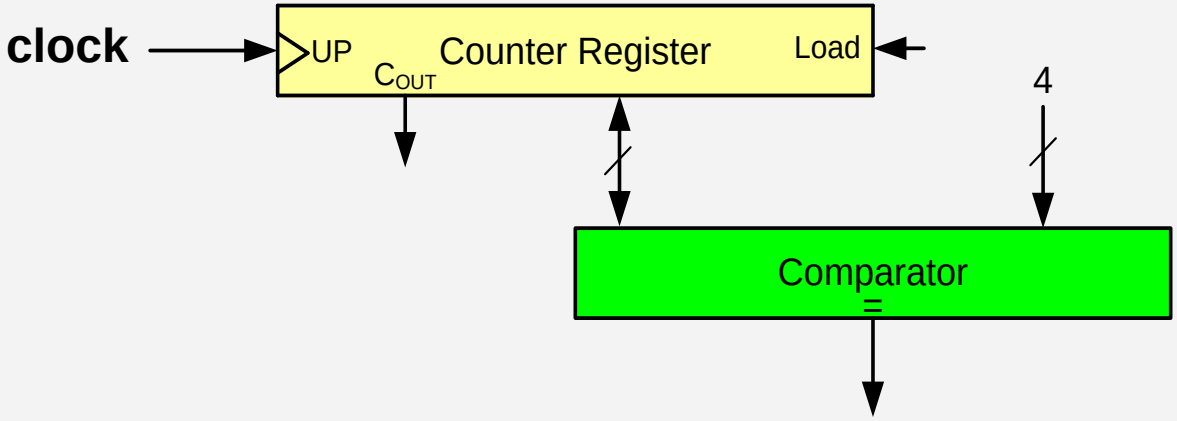
A counter register - Second design



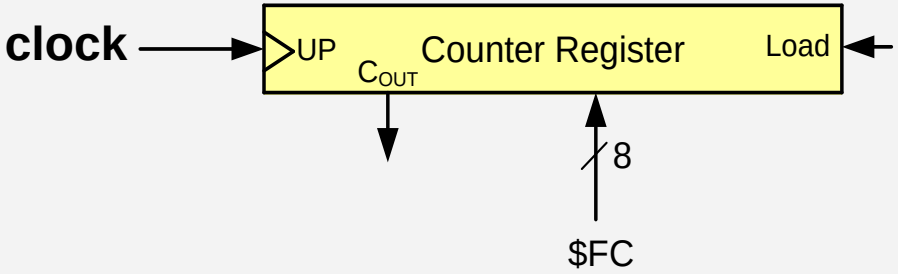
A counter register



A counter register - First design

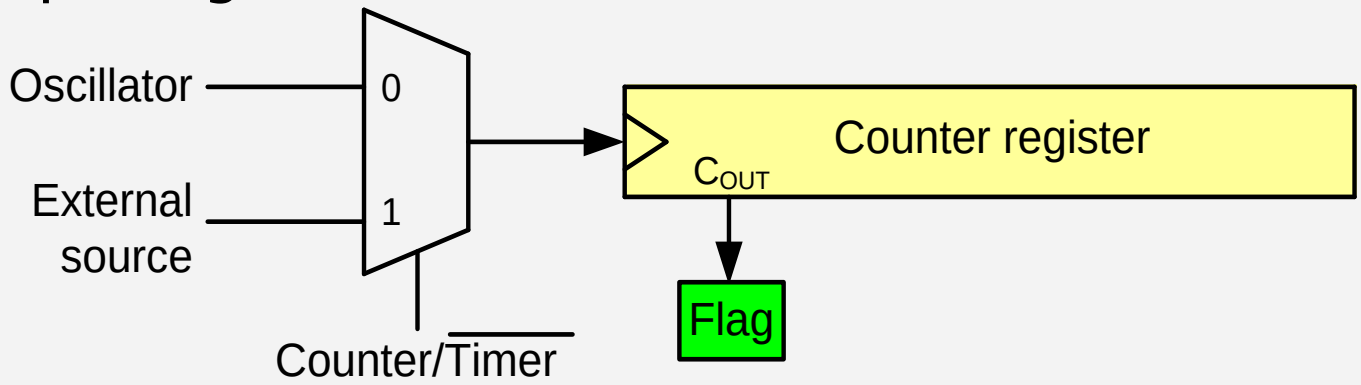


A counter register - Second design



A generic timer/counter

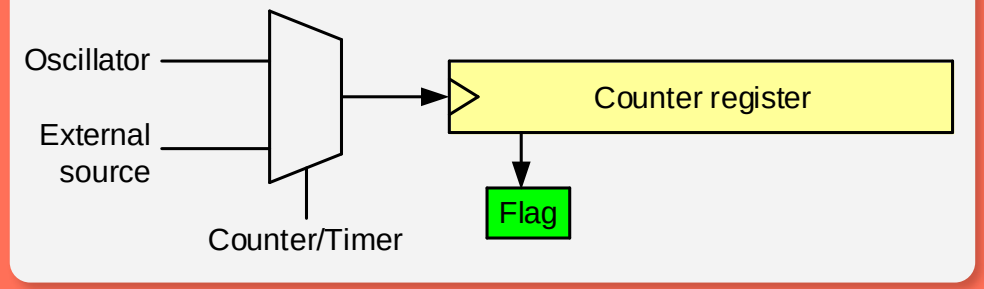
- Delay generating
- Counting
- Wave-form generating
- Capturing



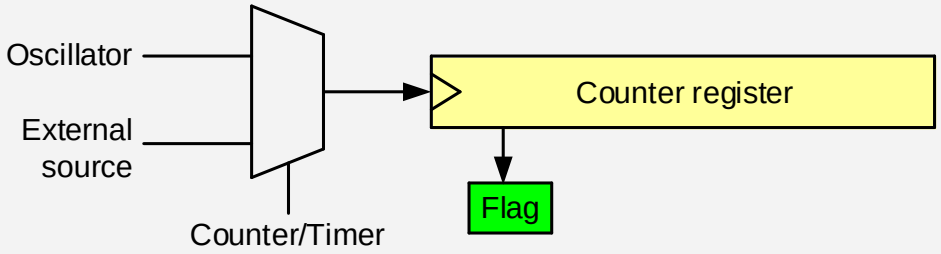
A generic timer/counter

- 1 to 6 timers
  - 3 timers in ATmega328
- 8-bit and 16-bit timers
  - two 8-bit timers and one 16-bit timer in ATmega328

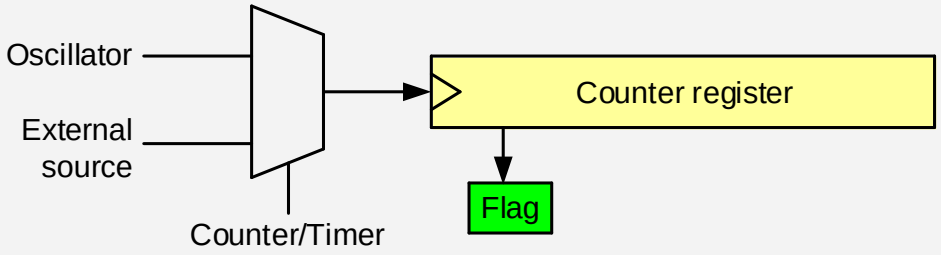
Timer in AVR


Oscillator
External source
Counter/Timer
Counter register
Flag

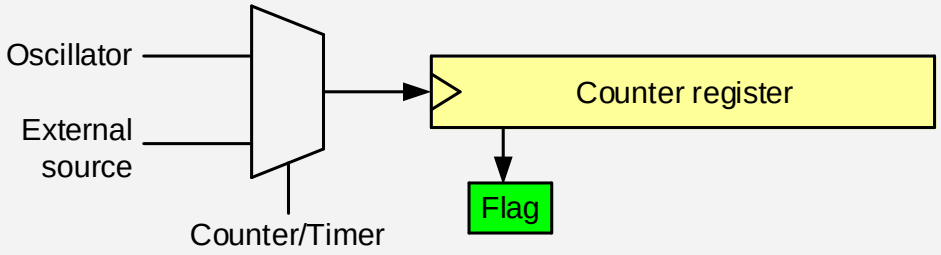
Timer in AVR


Oscillator
External source
Counter/Timer
Counter register
Flag

Timer in AVR

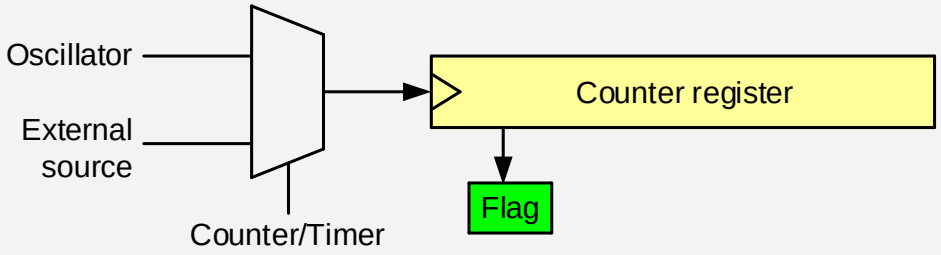

Oscillator
External source
Counter/Timer
Counter register
Flag

Timer in AVR



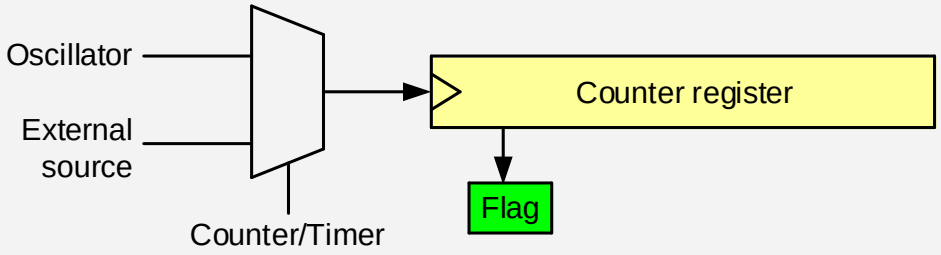
Timer in AVR

Timer in AVR



Timer in AVR

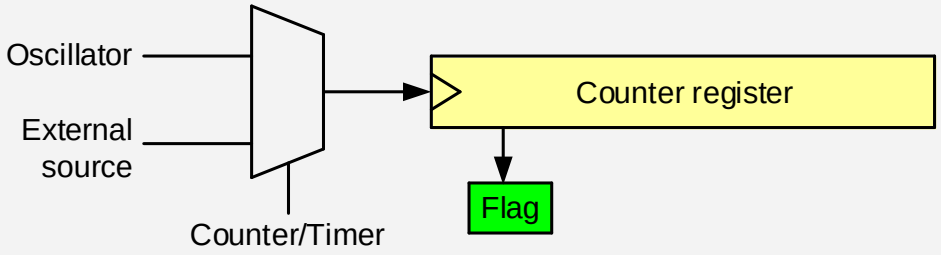
Timer in AVR



note

Timer in AVR

Timer in AVR

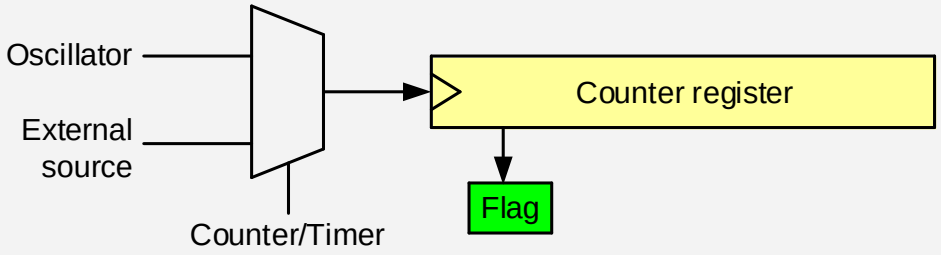


note

Timer in AVR

Timer in AVR

Timer in AVR



note

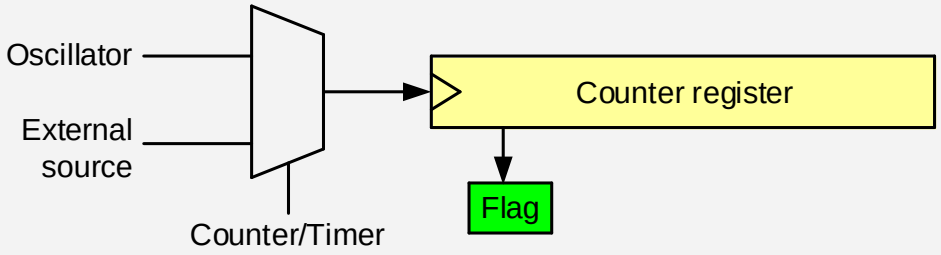
All of the timer registers are byte-addressable registers

Timer in AVR

Timer in AVR

TCNTn

Timer in AVR



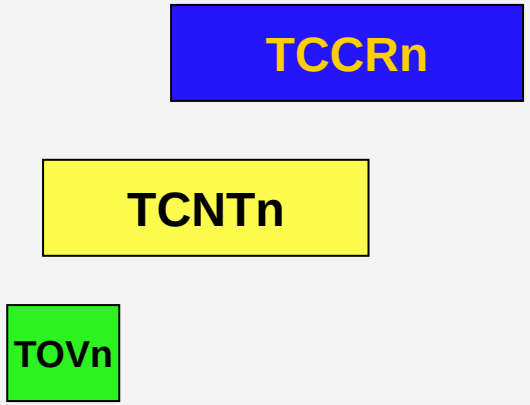
note

All of the timer registers are byte-addressable registers

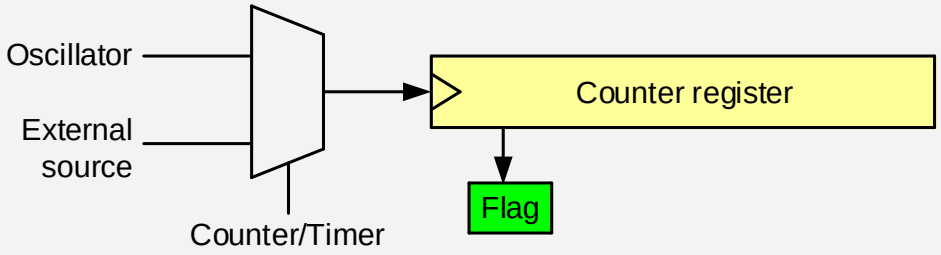
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



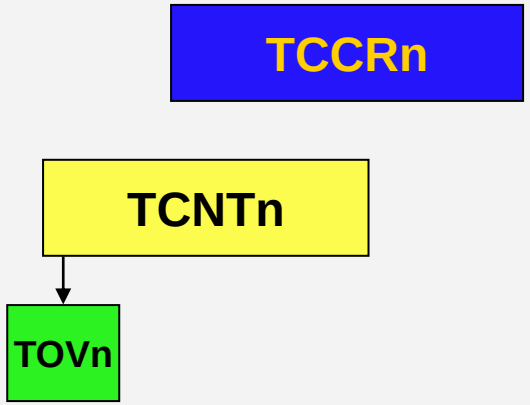
note

All of the timer registers are byte-addressable registers

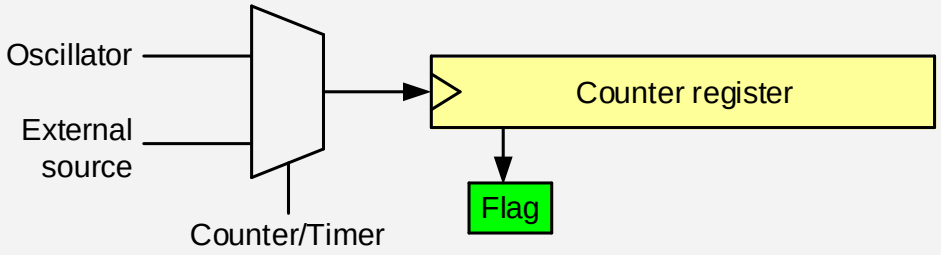
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



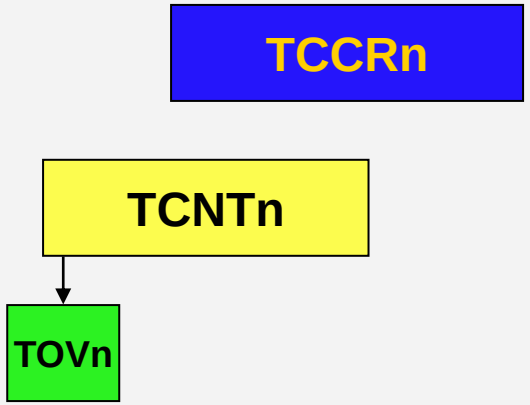
note

All of the timer registers are byte-addressable registers

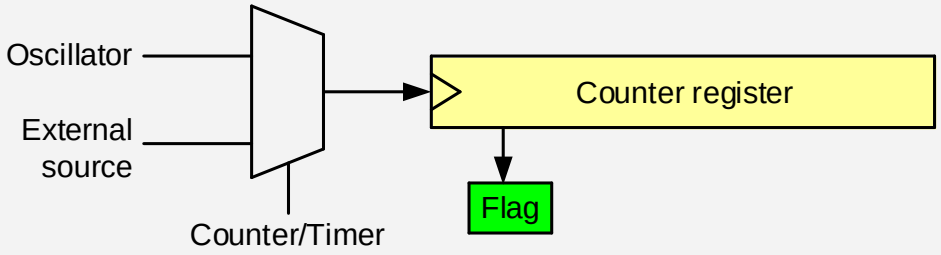
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



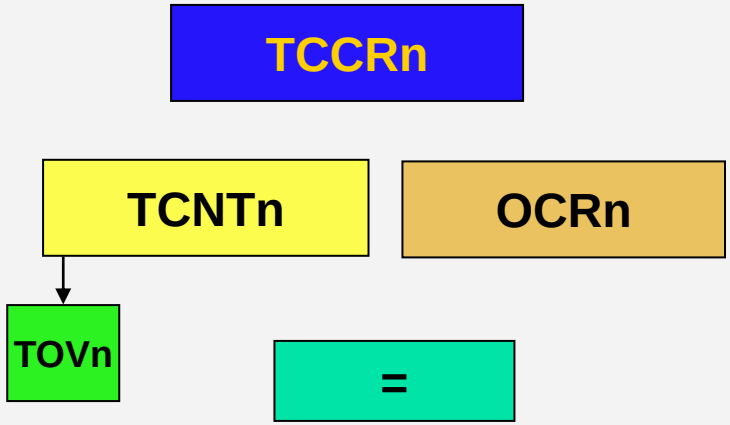
note

All of the timer registers are byte-addressable registers

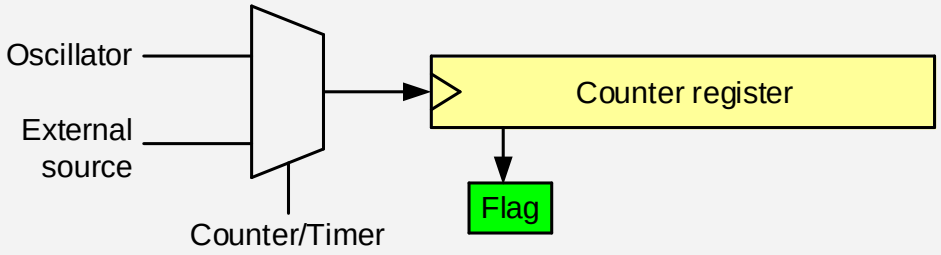
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



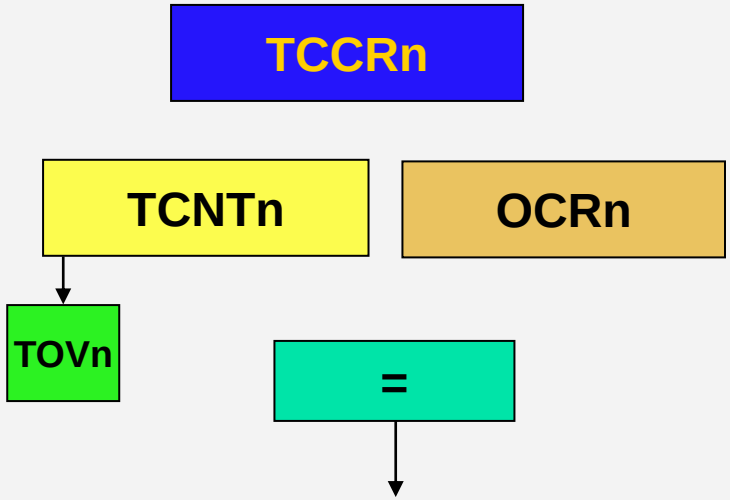
note

All of the timer registers are byte-addressable registers

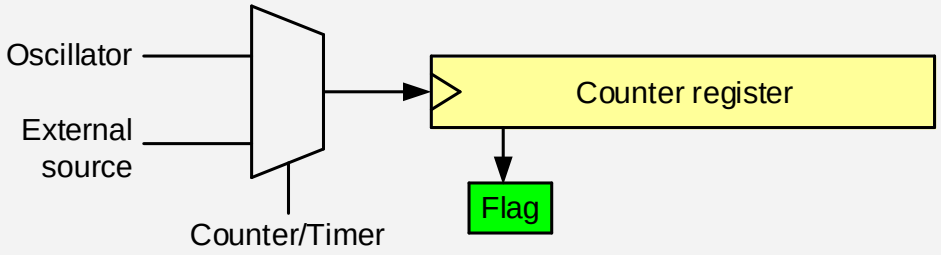
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



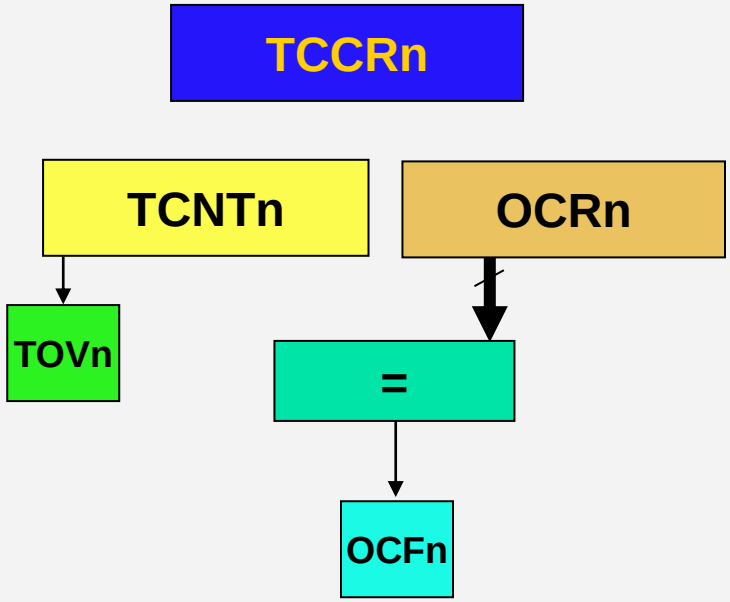
note

All of the timer registers are byte-addressable registers

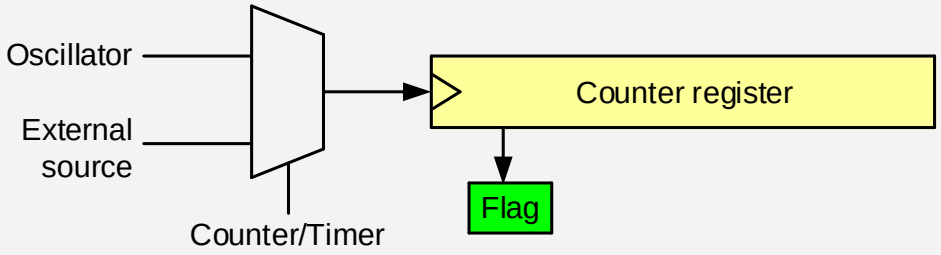
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



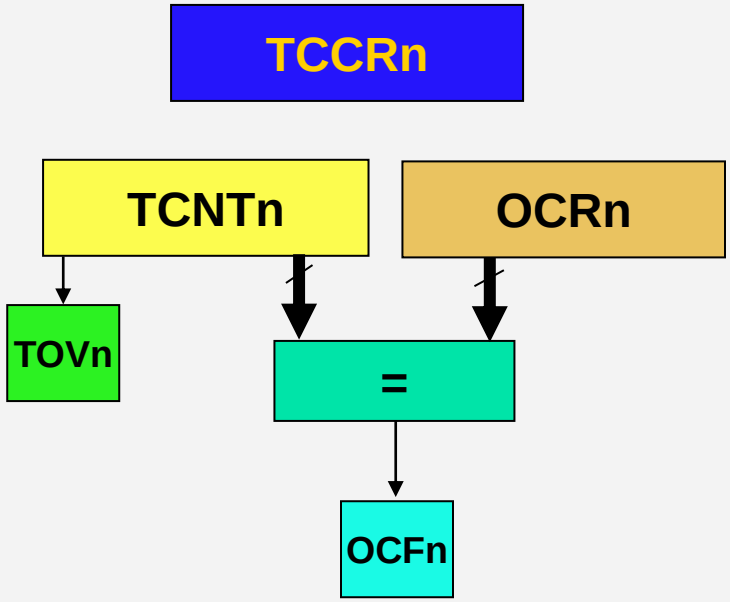
note

All of the timer registers are byte-addressable registers

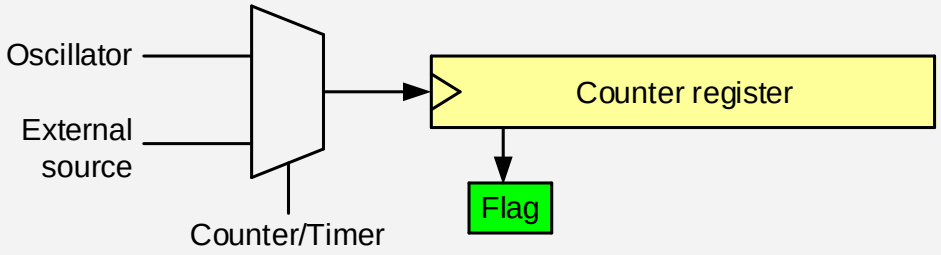
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



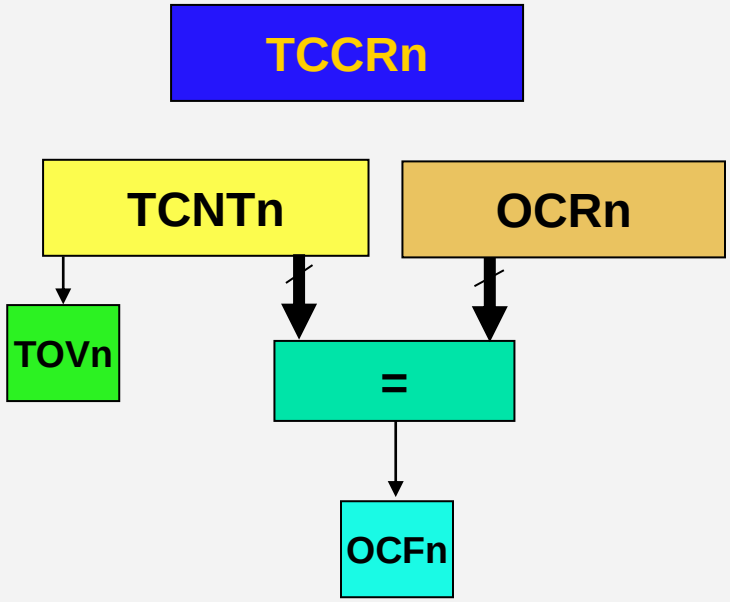
note

All of the timer registers are byte-addressable registers

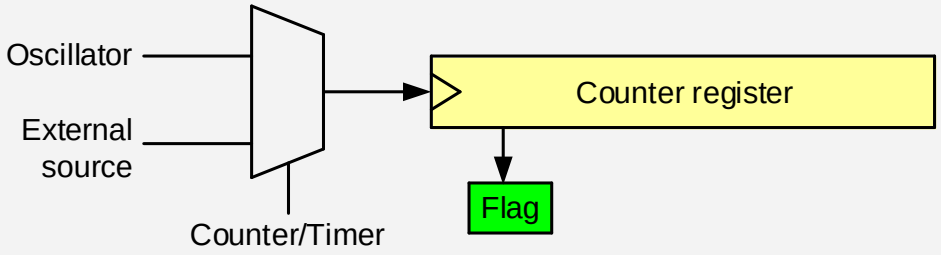
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



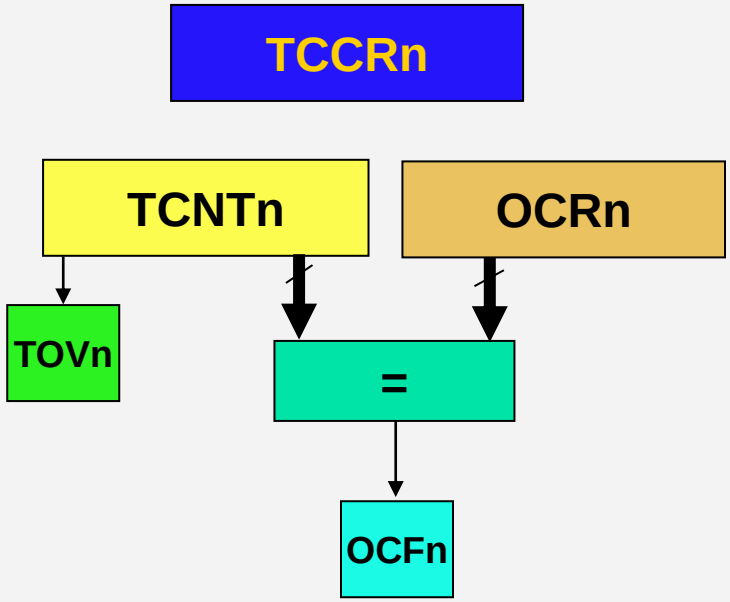
note

All of the timer registers are byte-addressable registers

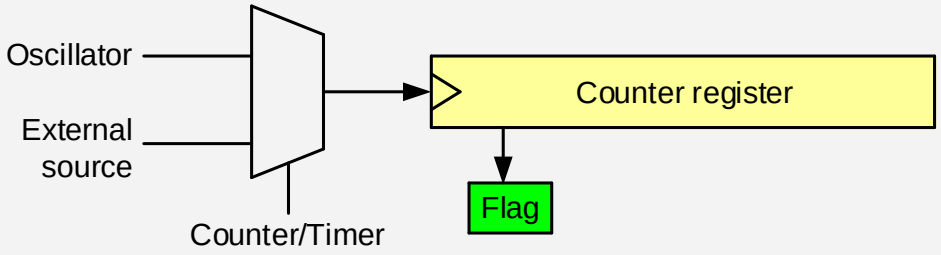
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



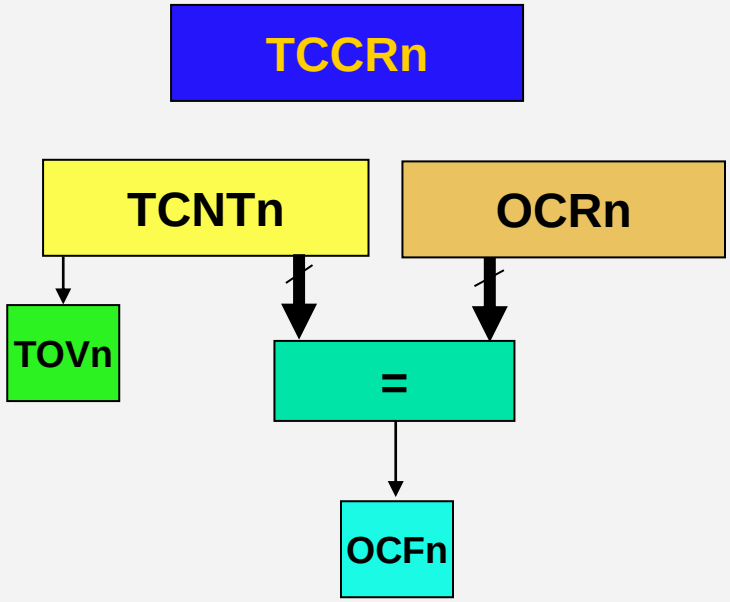
note

All of the timer registers are byte-addressable registers

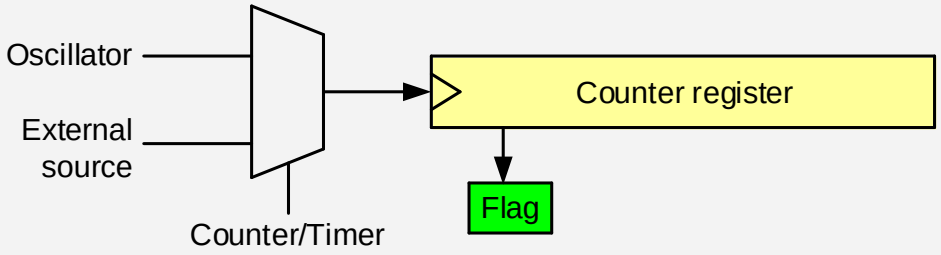
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



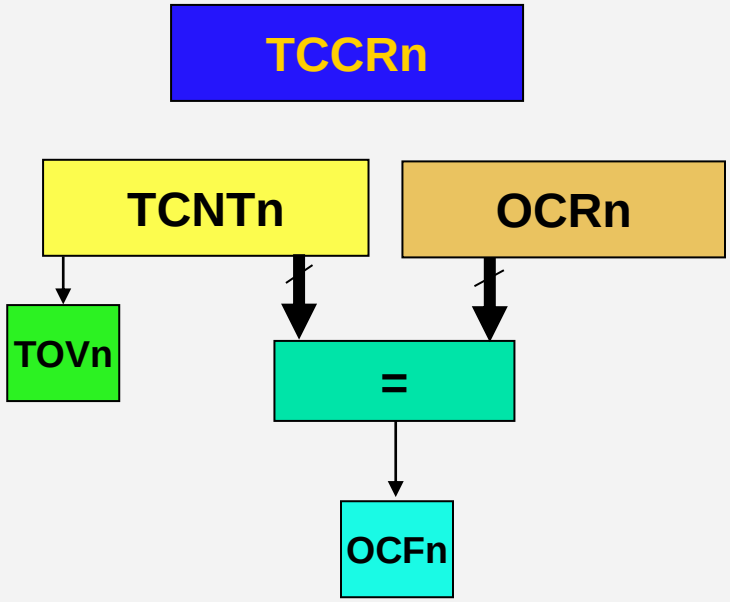
note

All of the timer registers are byte-addressable registers

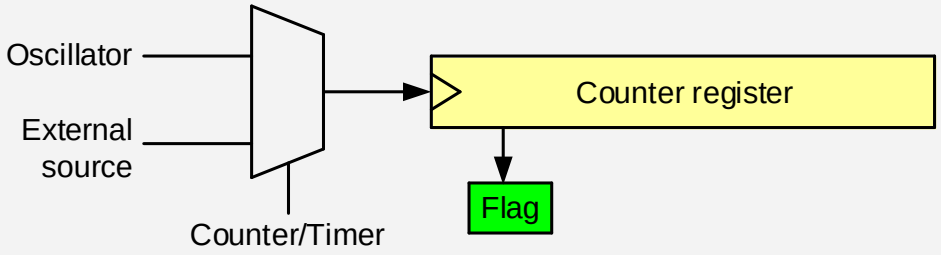
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



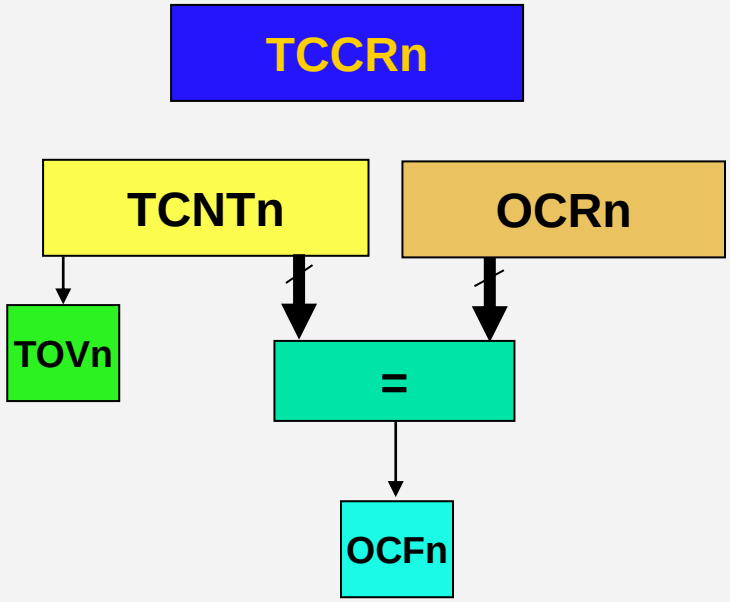
note

All of the timer registers are byte-addressable registers

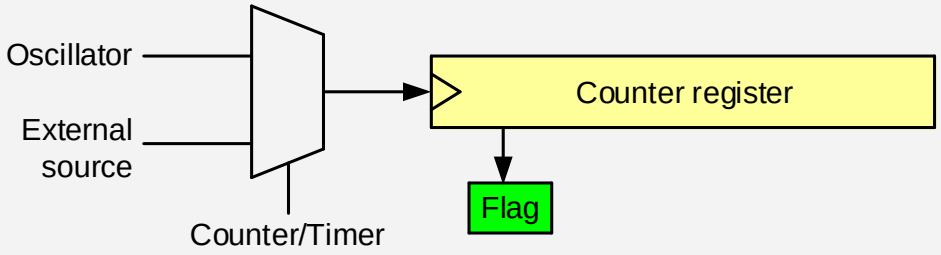
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



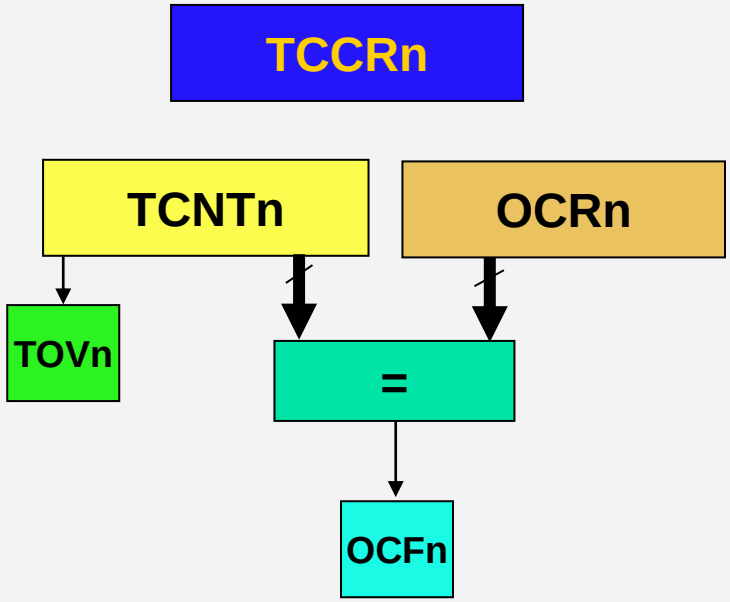
note

All of the timer registers are byte-addressable registers

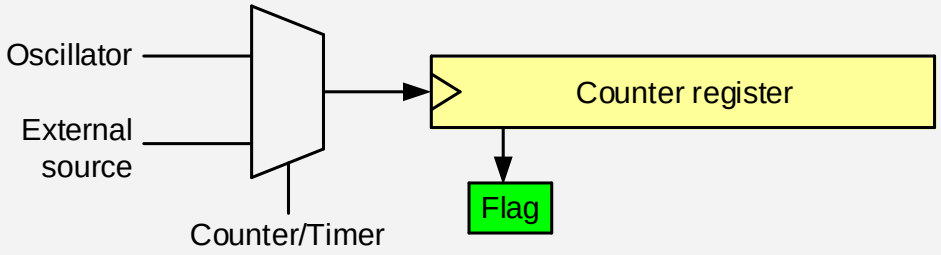
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



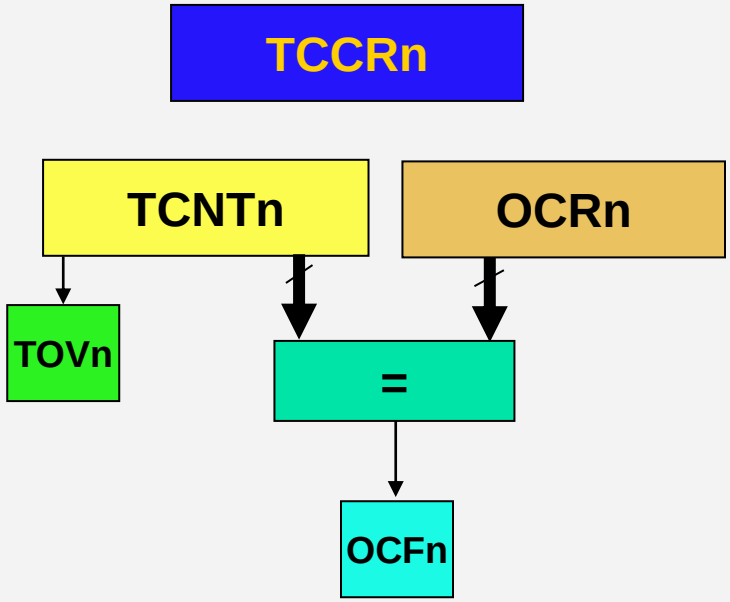
note

All of the timer registers are byte-addressable registers

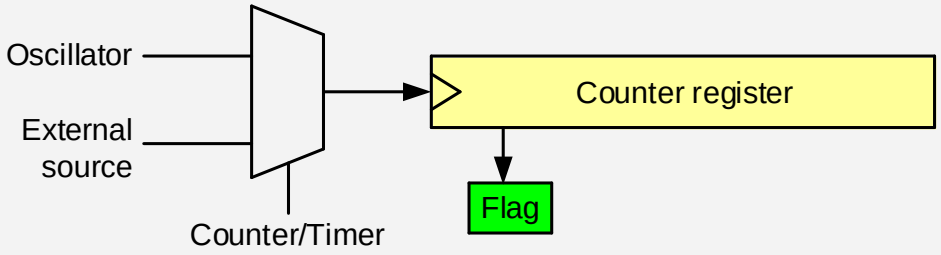
Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer in AVR



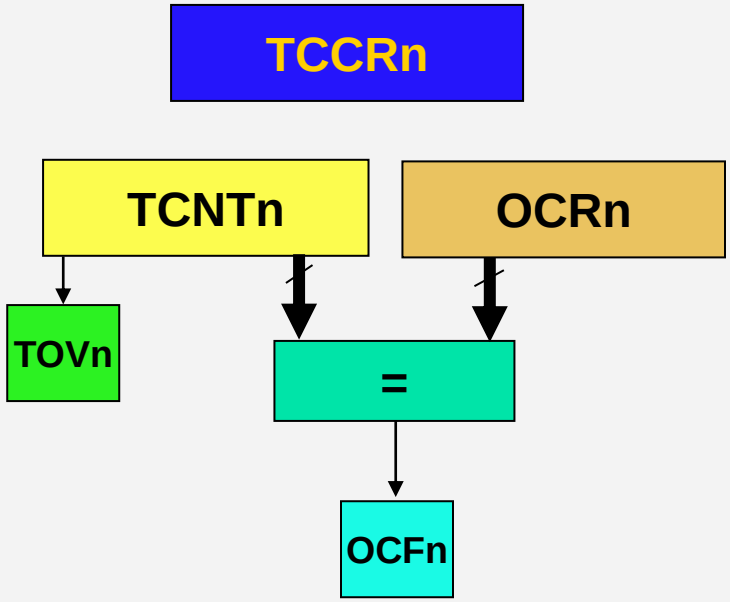
note

All of the timer registers are byte-addressable registers

Timer in AVR

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

Timer in AVR



Timer 0 (8-bit timer)

–	–	–	–	–	OCIE0B	OCIE0A	TOIE0

TIMSK0

Timer 0 (8-bit timer)

TCNT0

-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)

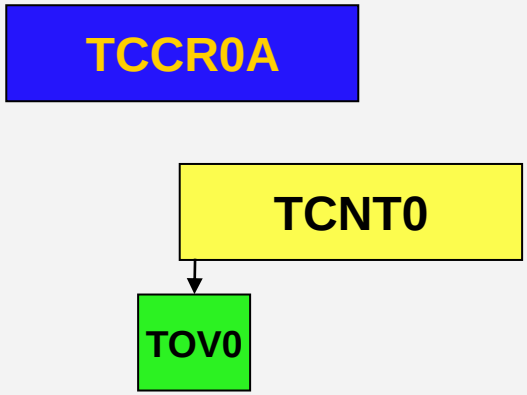
TCCR0A

TCNT0

TOV0

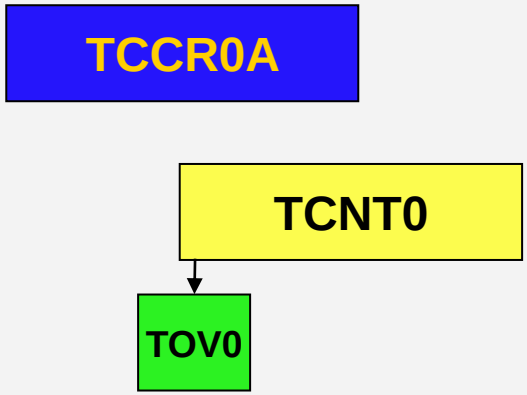
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



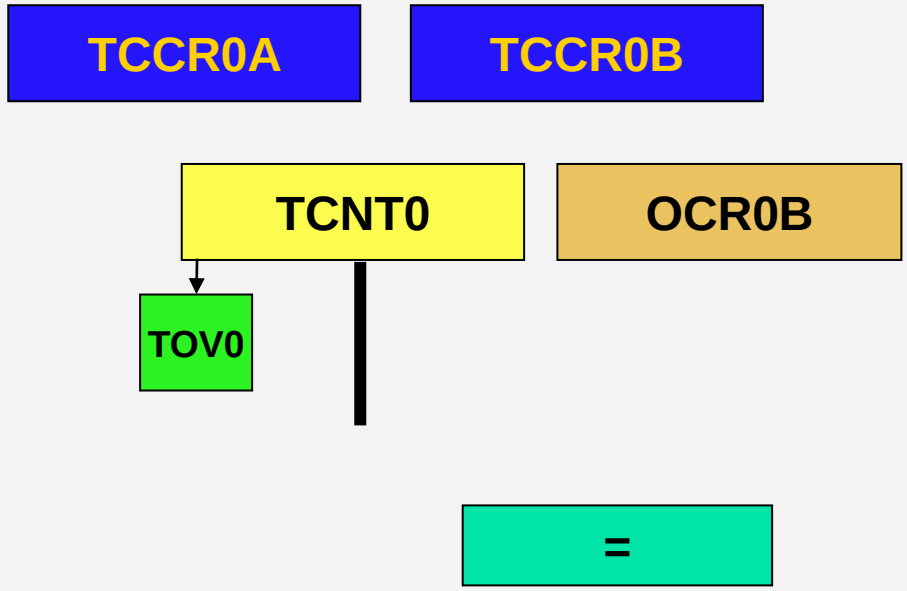
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



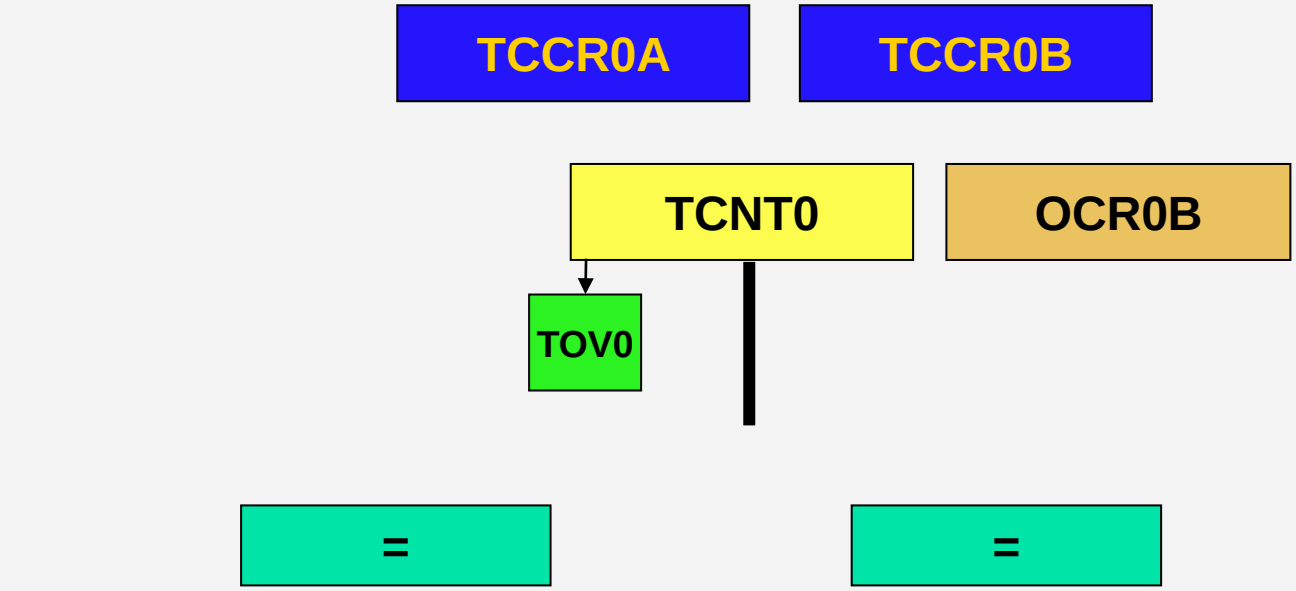
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



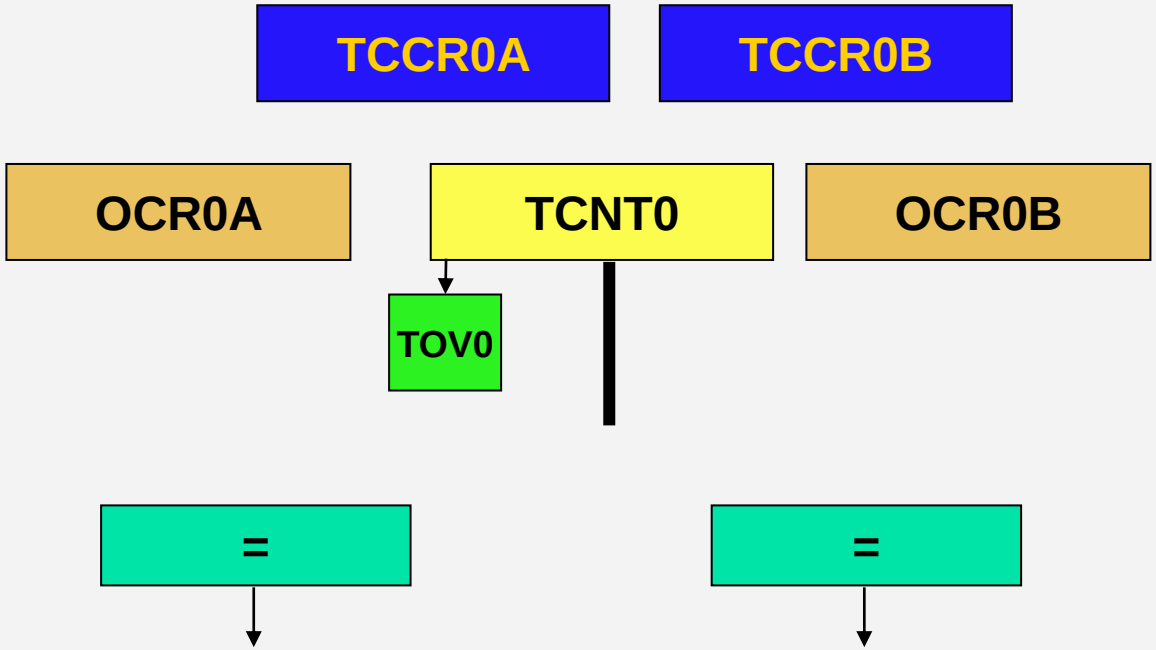
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



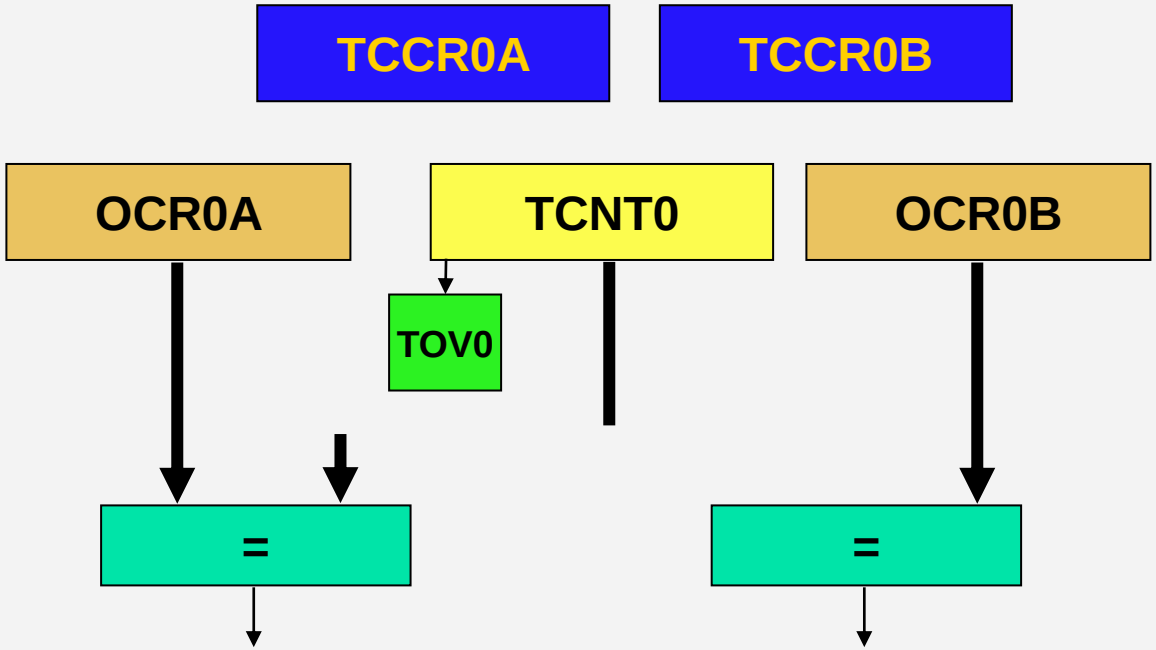
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



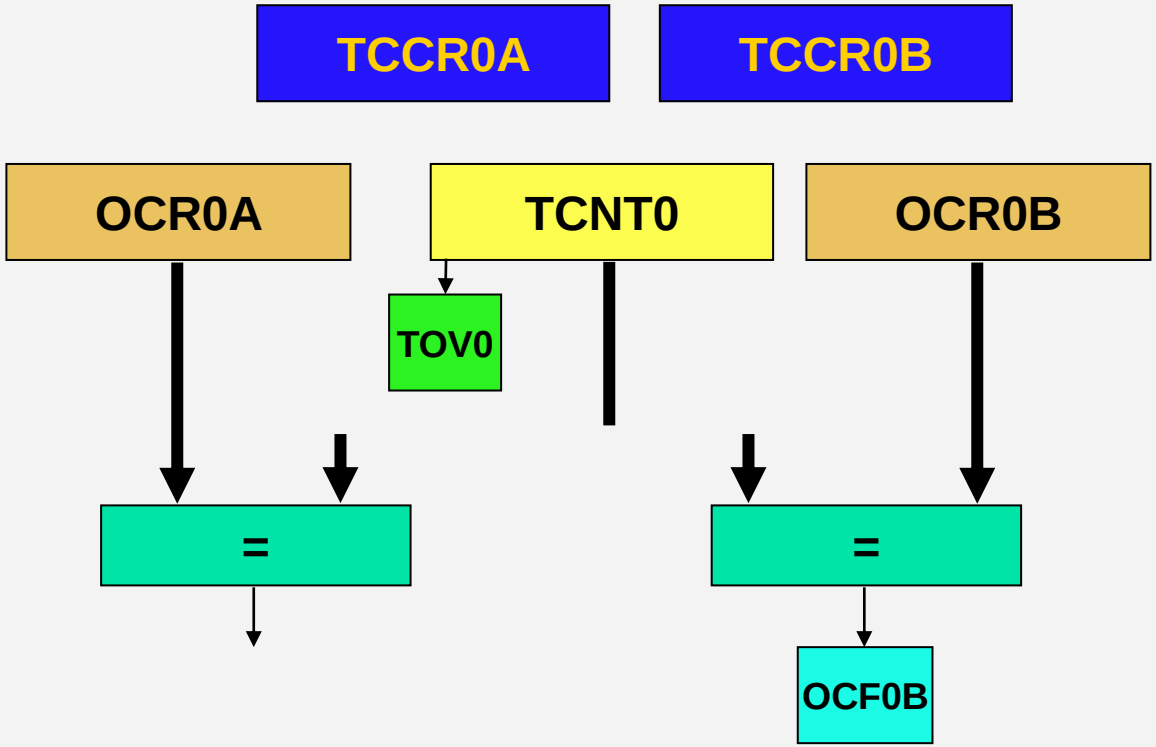
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



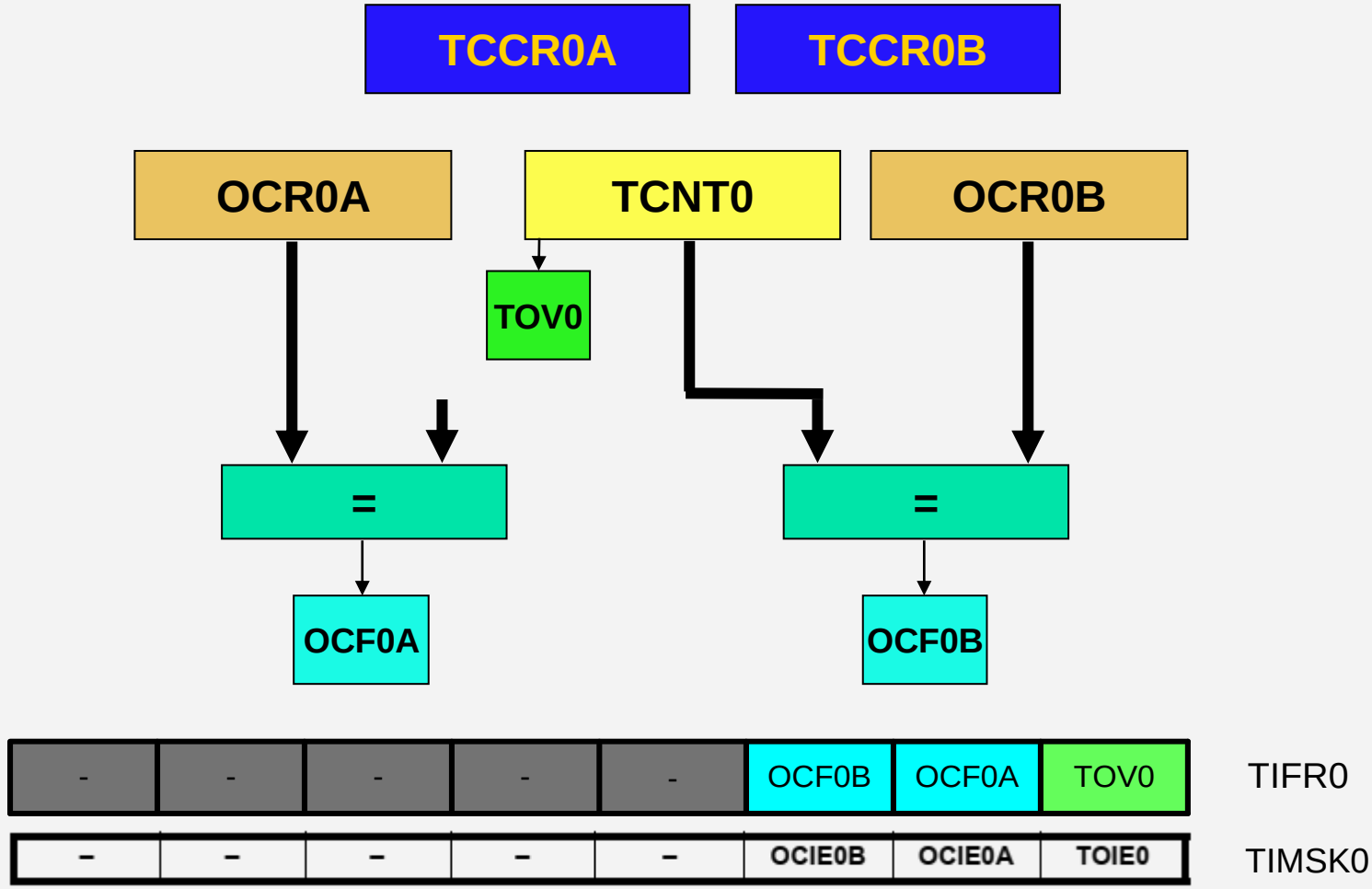
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)

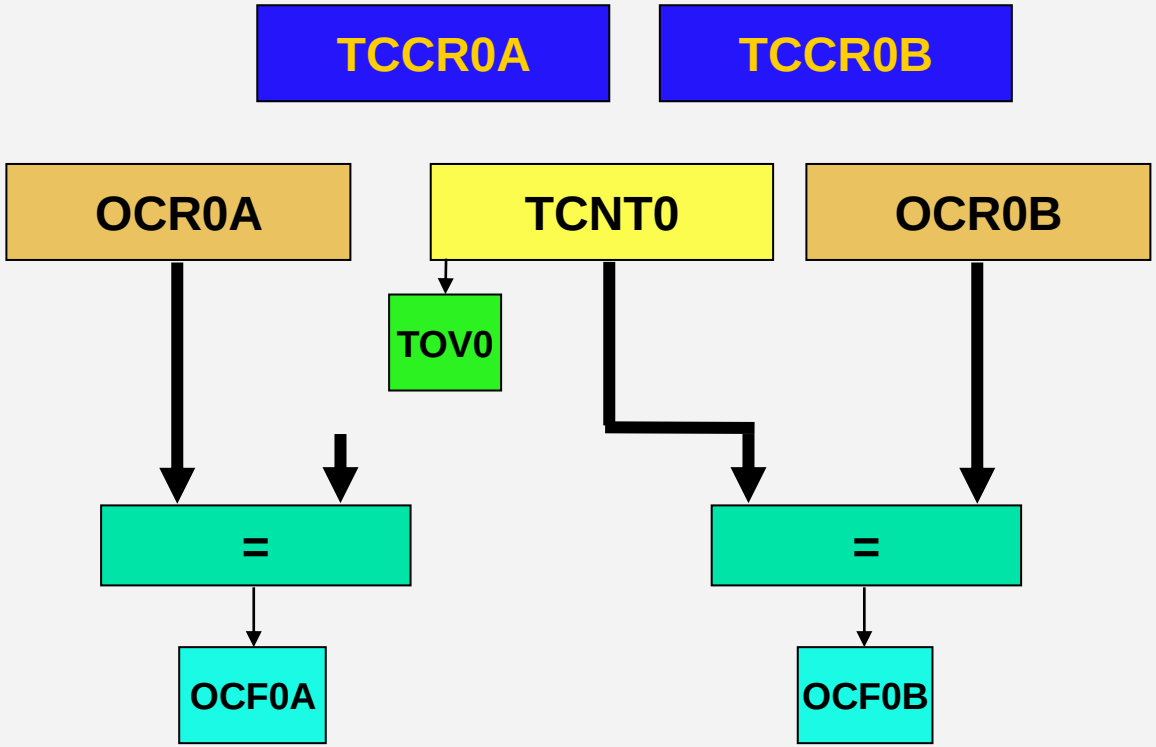


-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)

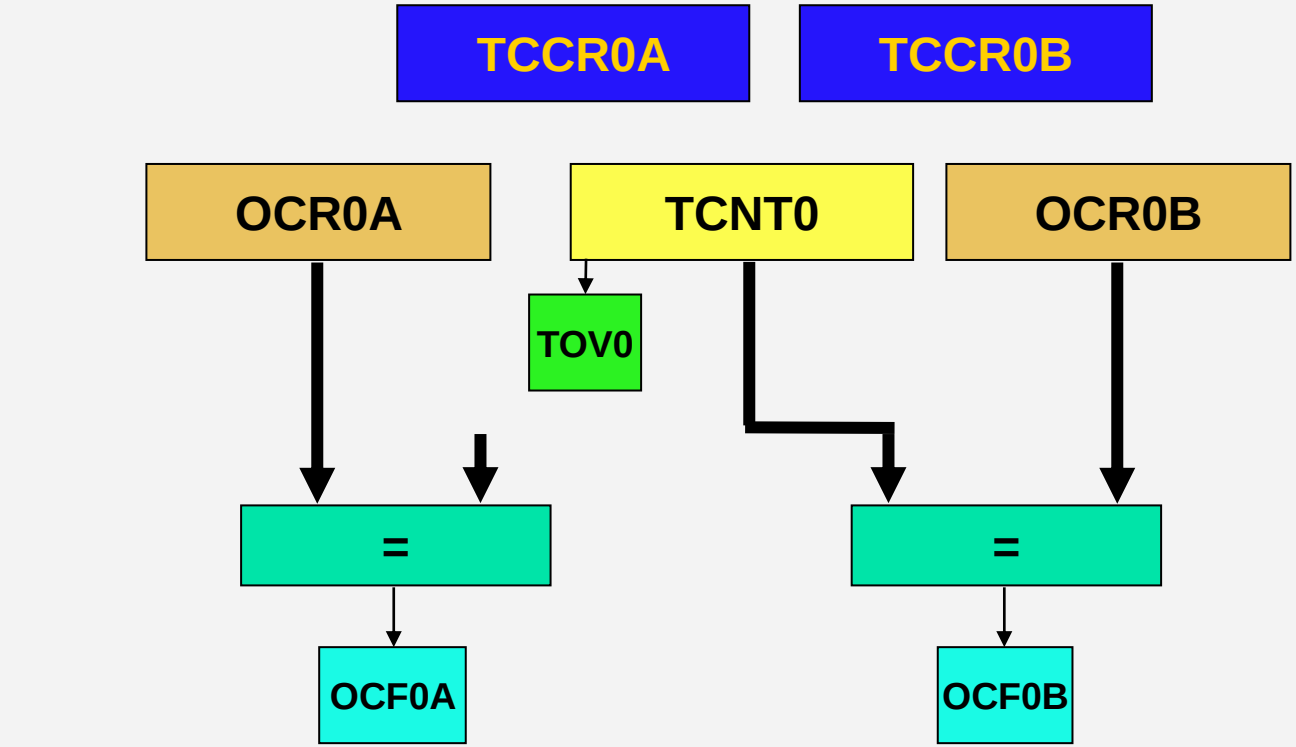


Timer 0 (8-bit timer)



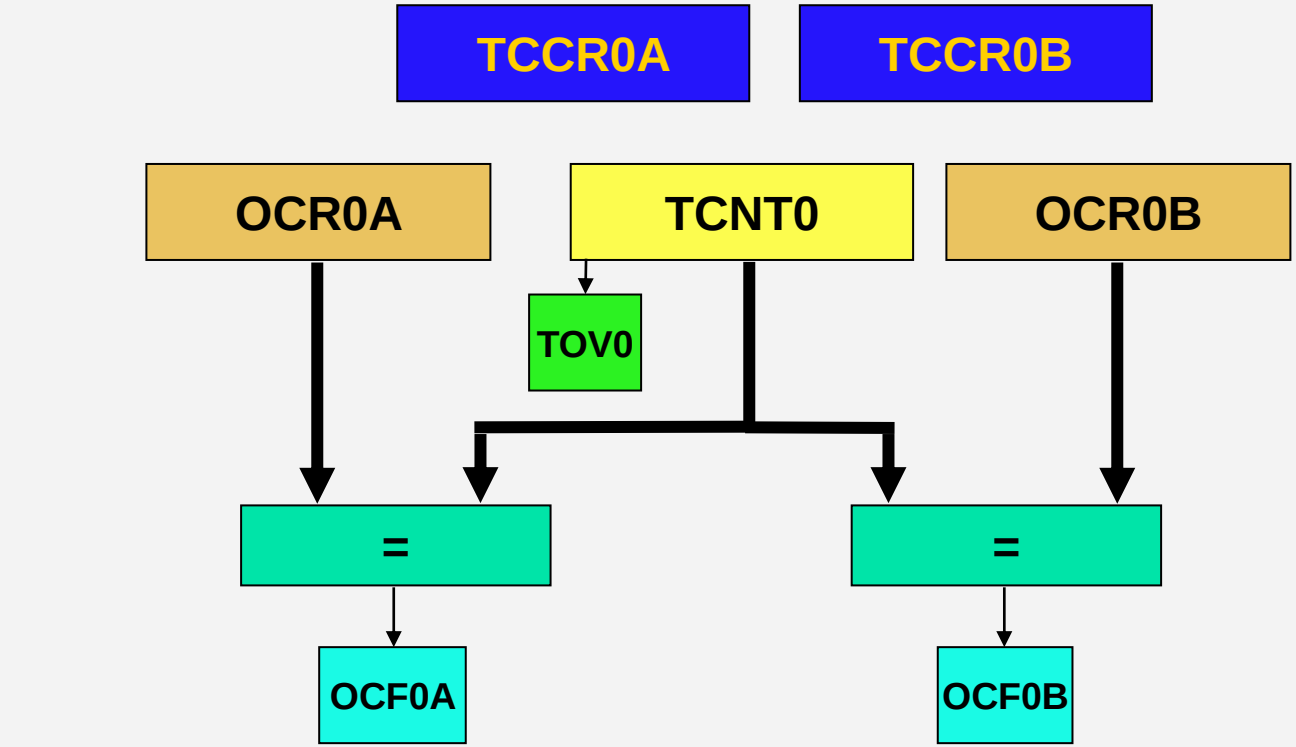
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



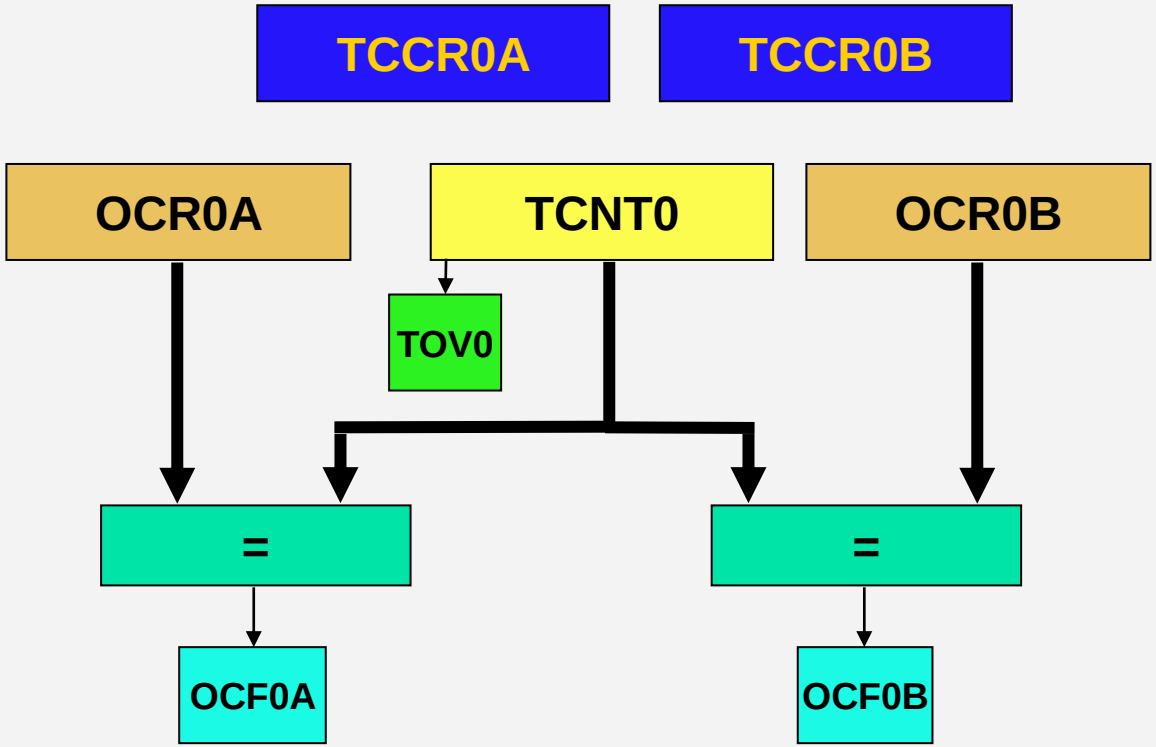
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



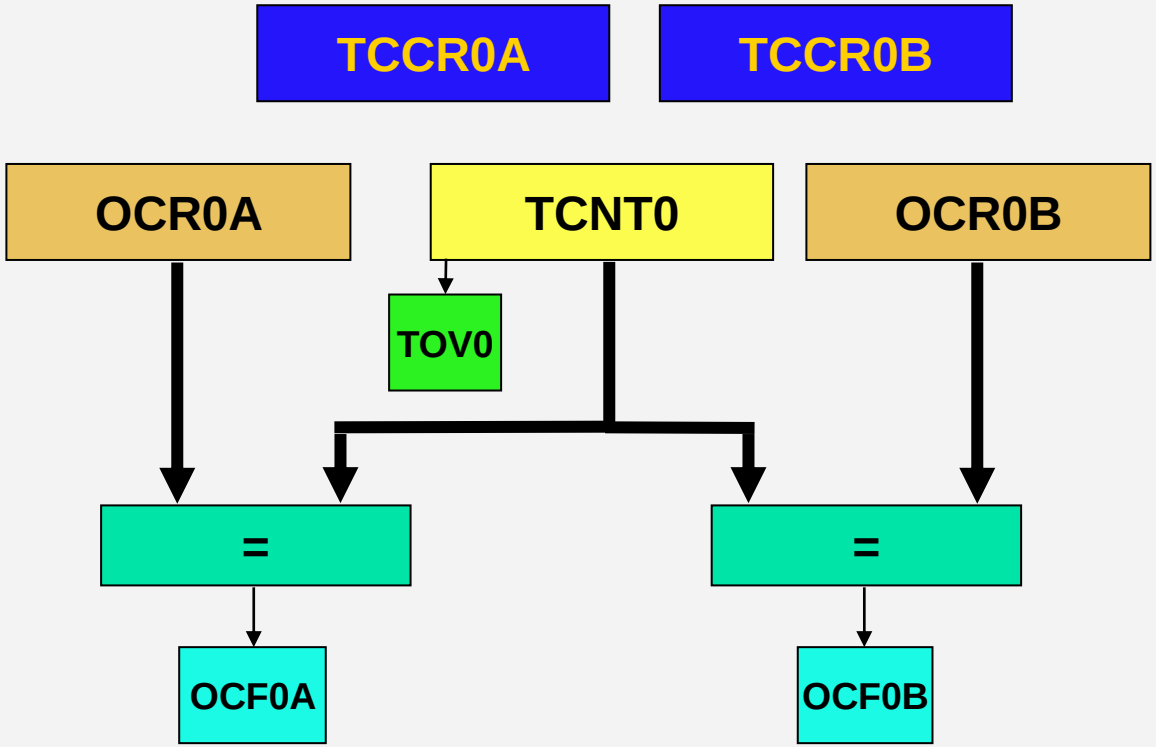
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



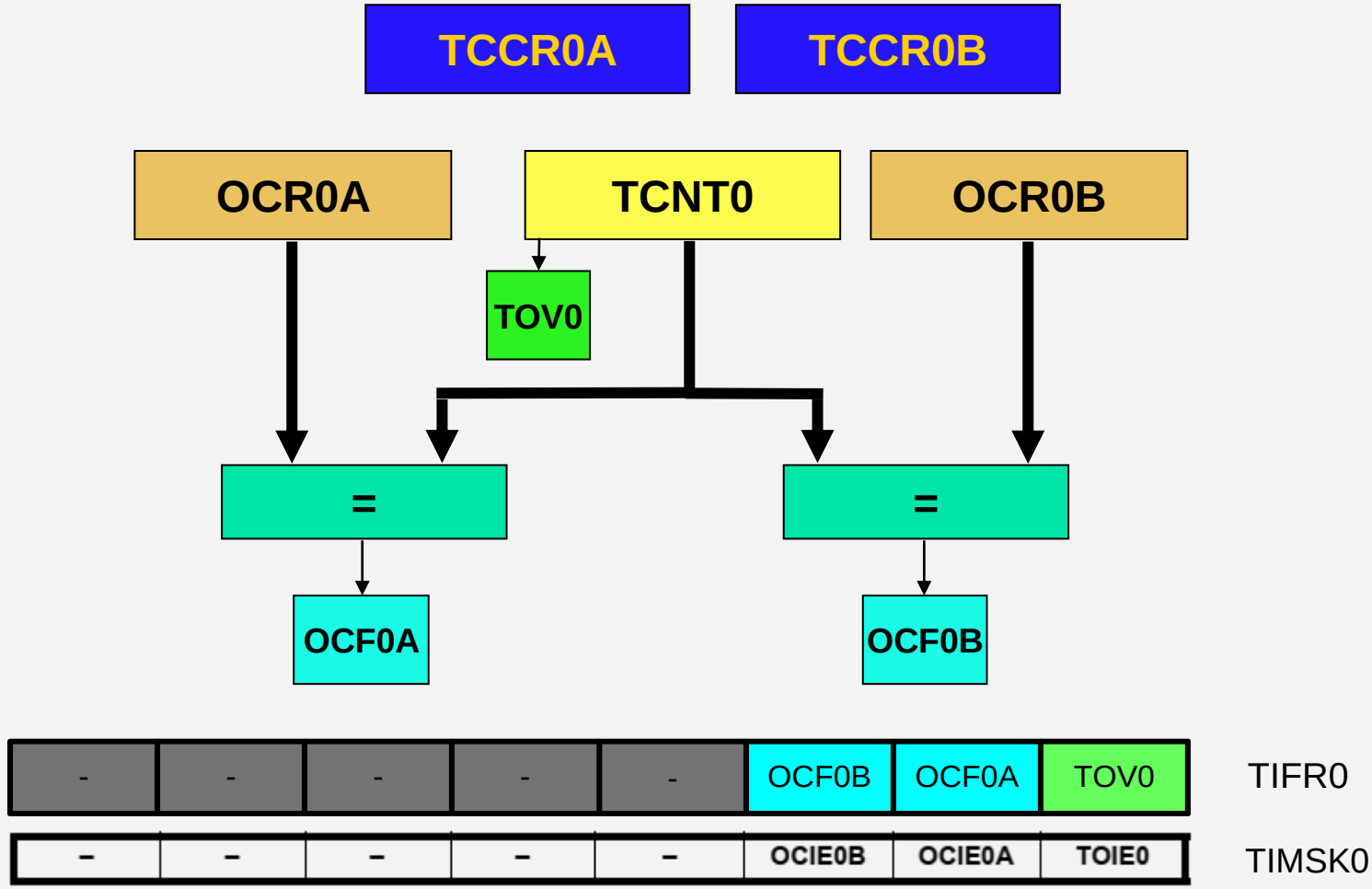
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)

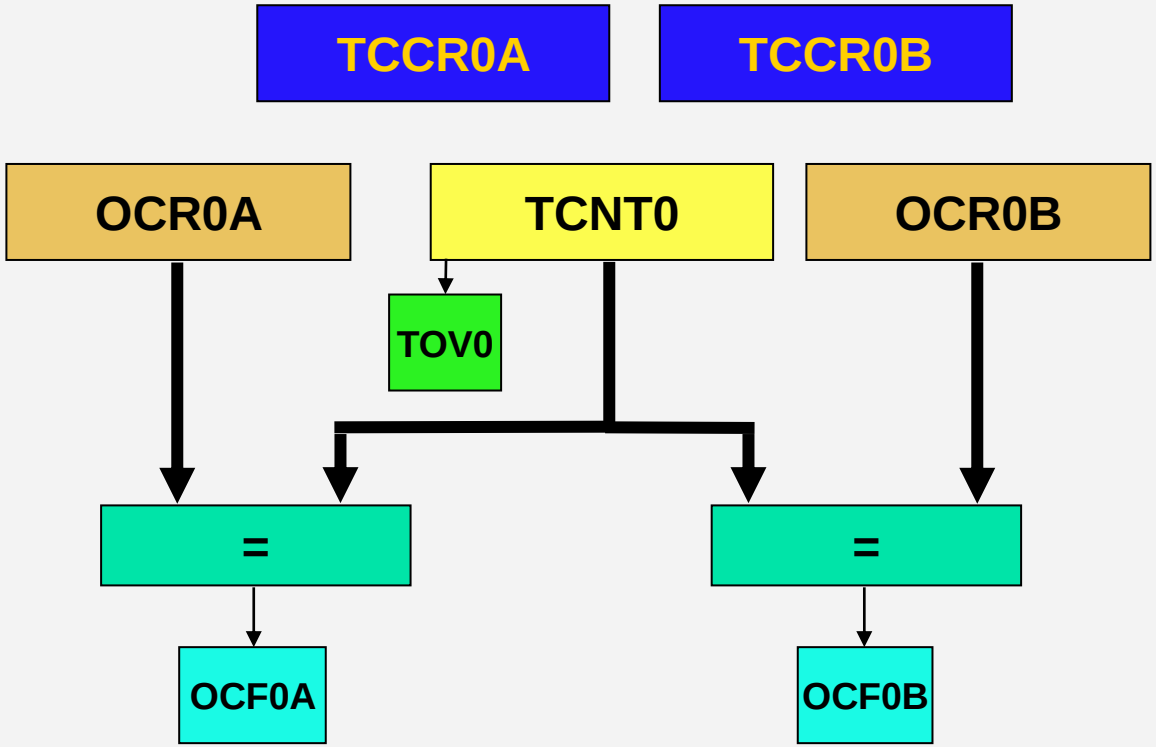


-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)

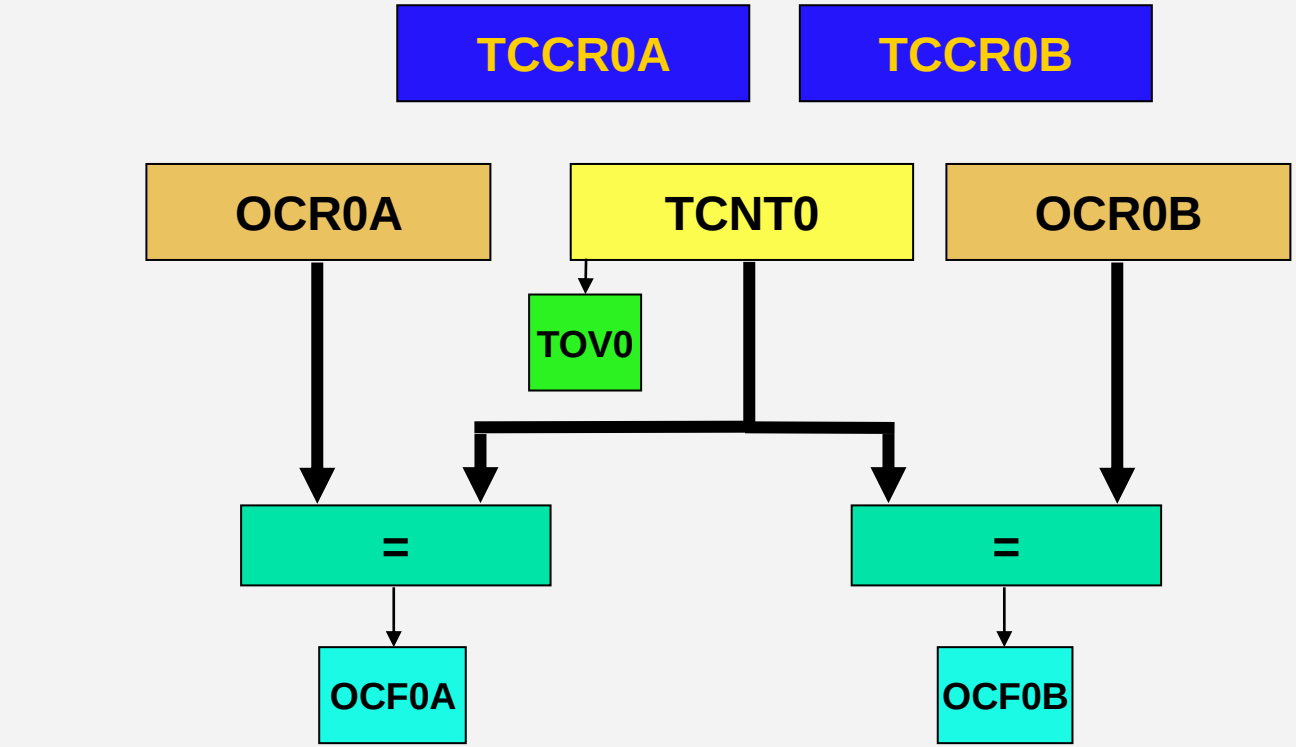


Timer 0 (8-bit timer)



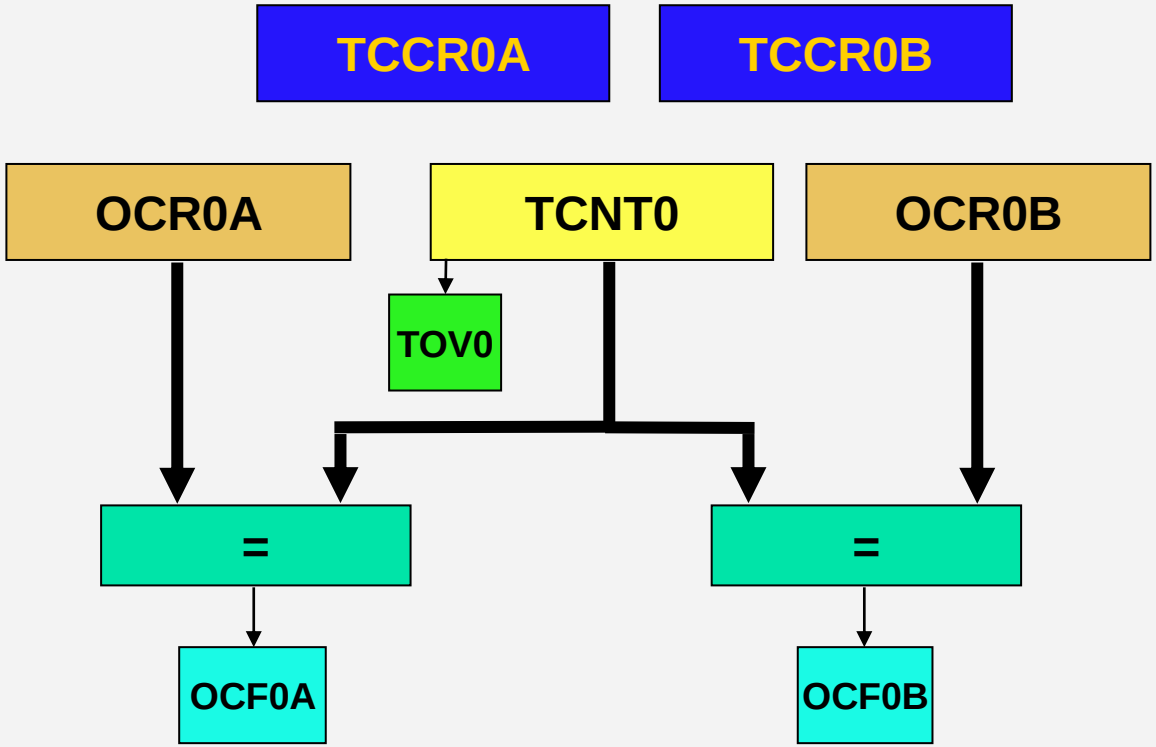
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



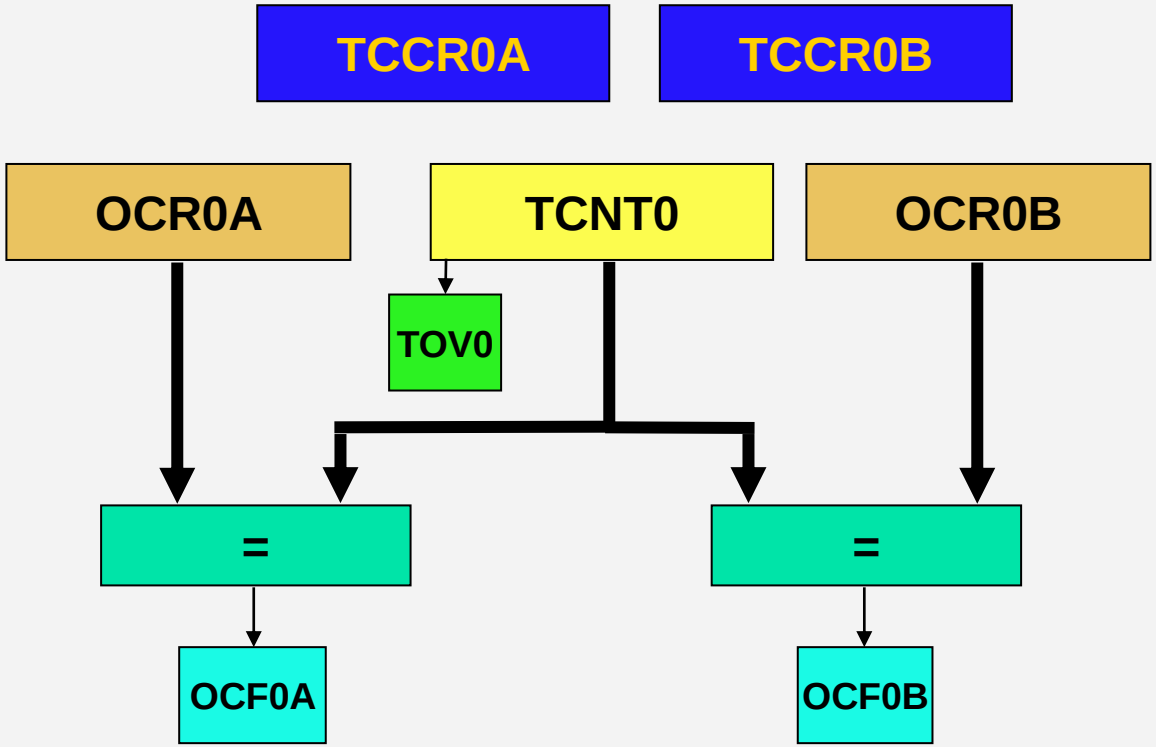
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



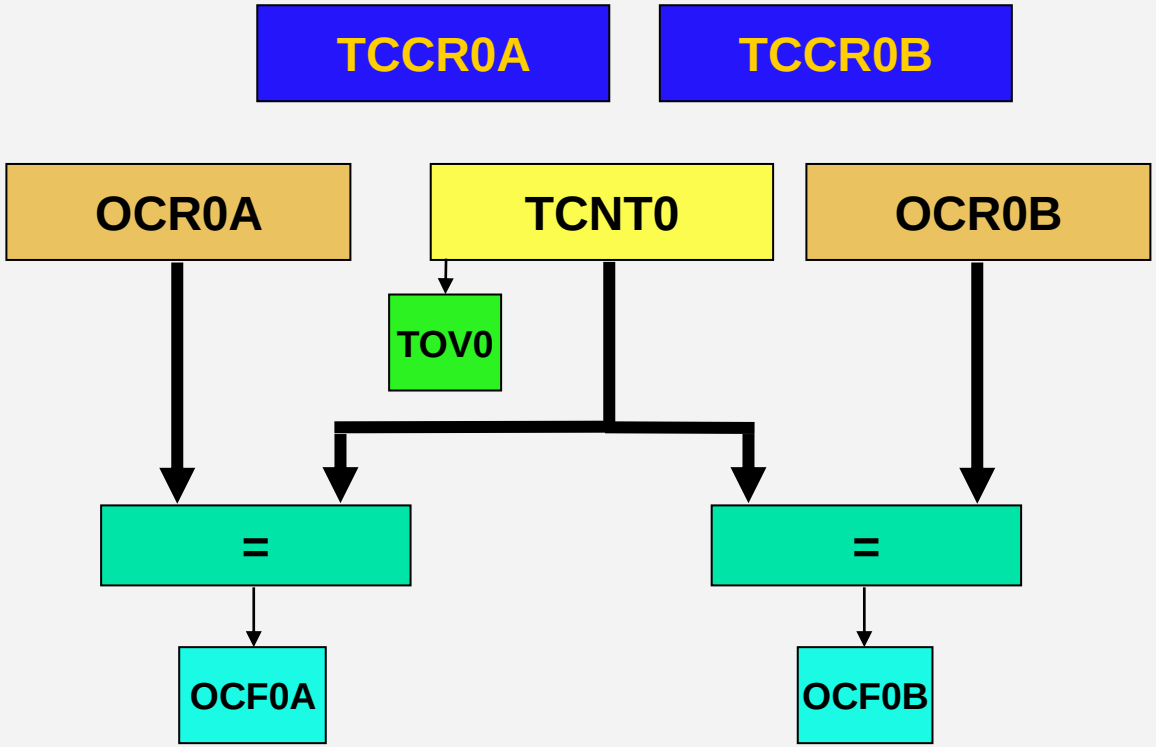
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



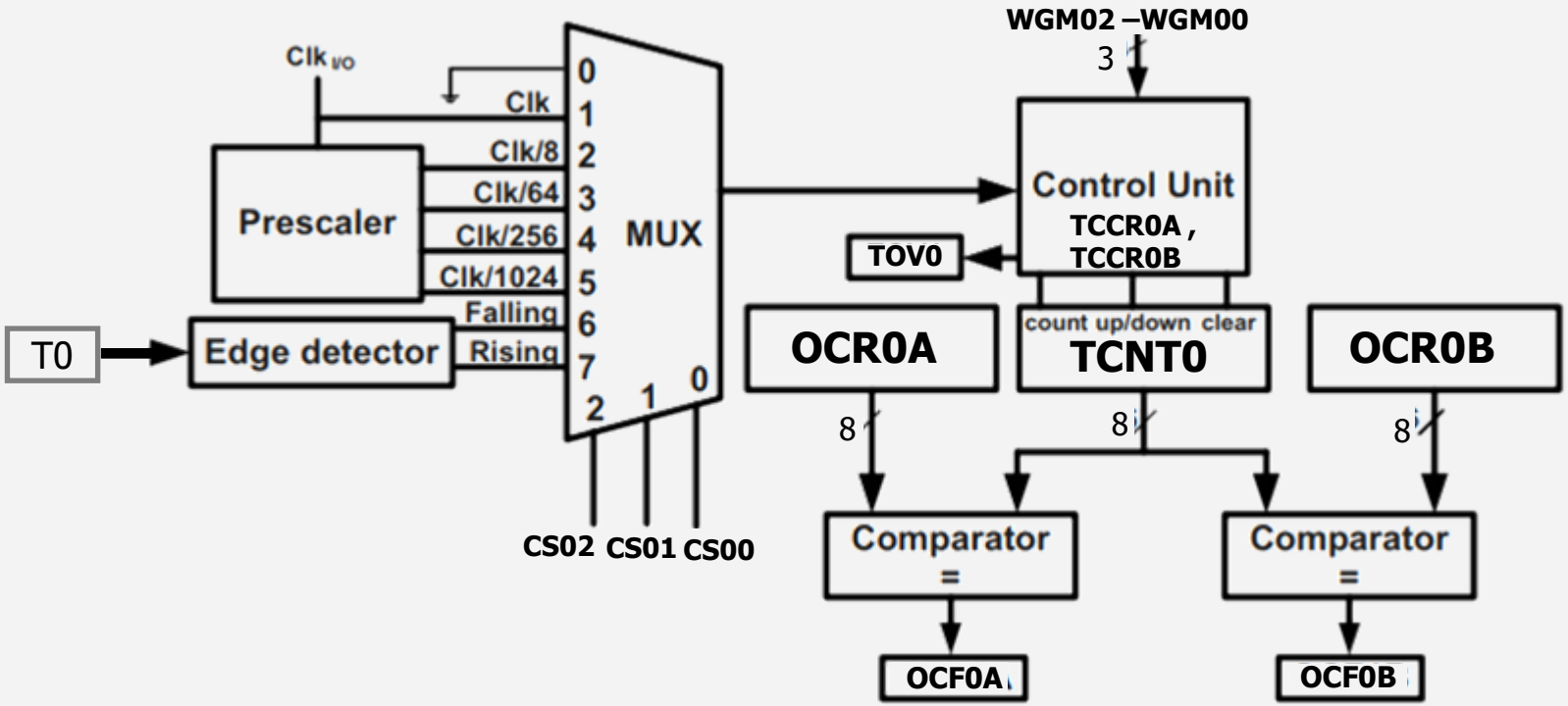
-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)

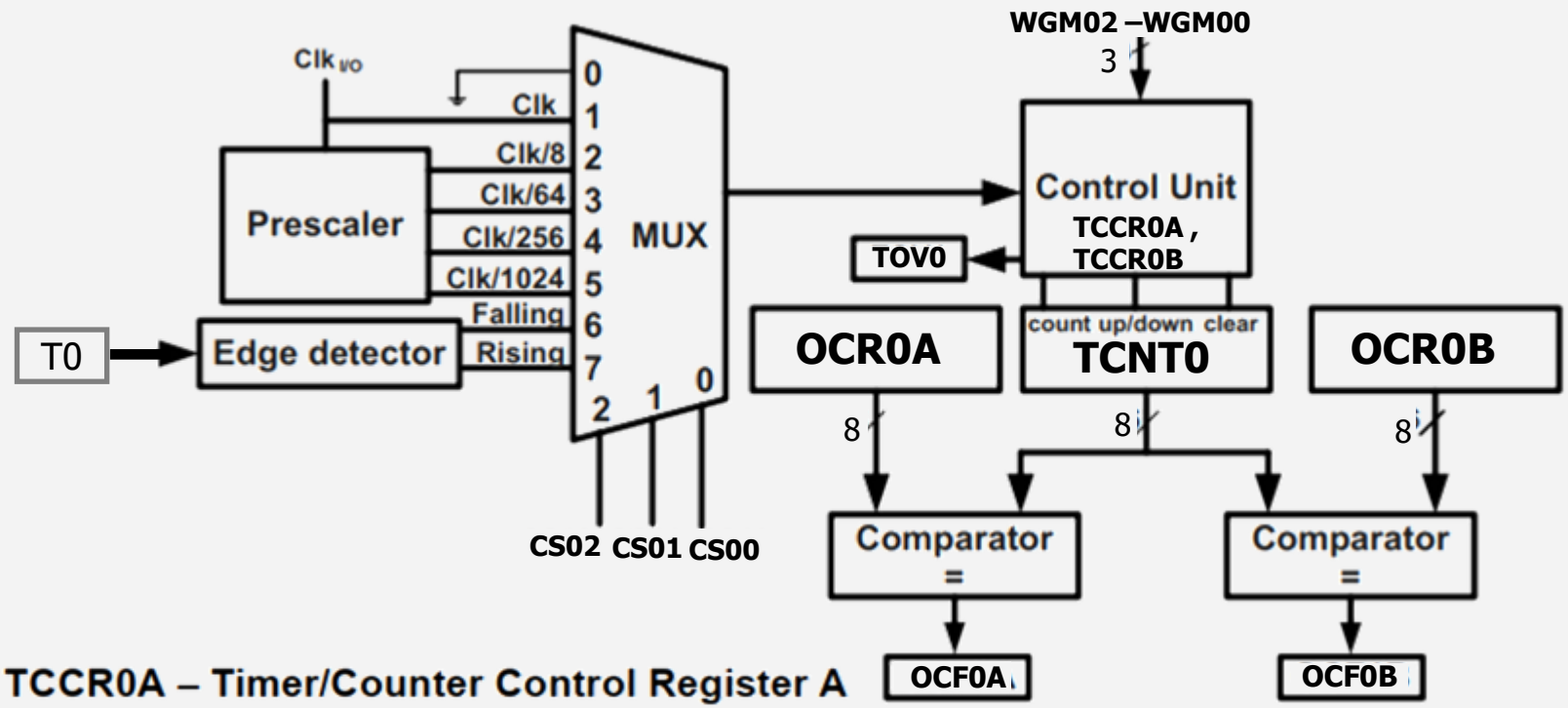


-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0

Timer 0 (8-bit timer)



Timer 0 (8-bit timer)



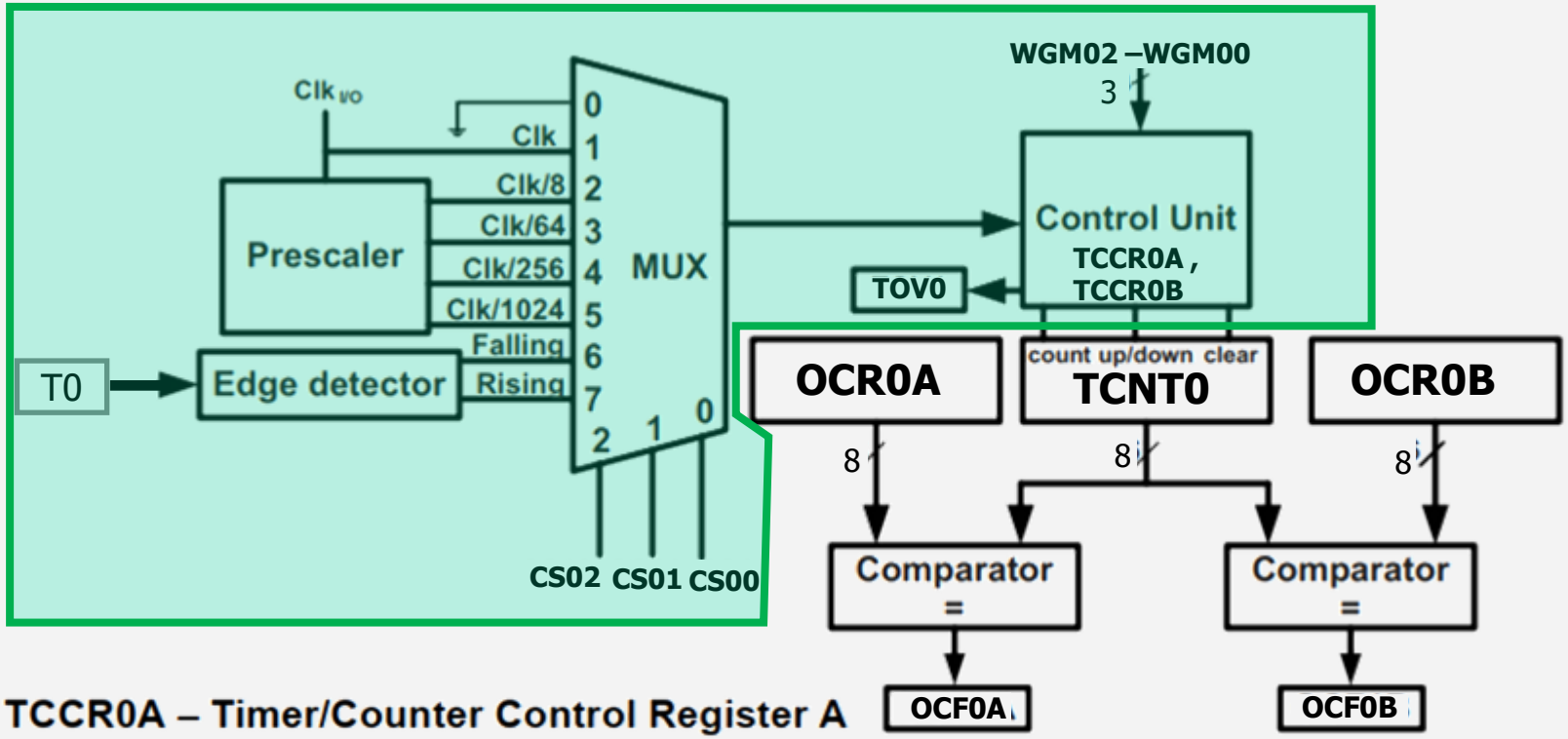
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



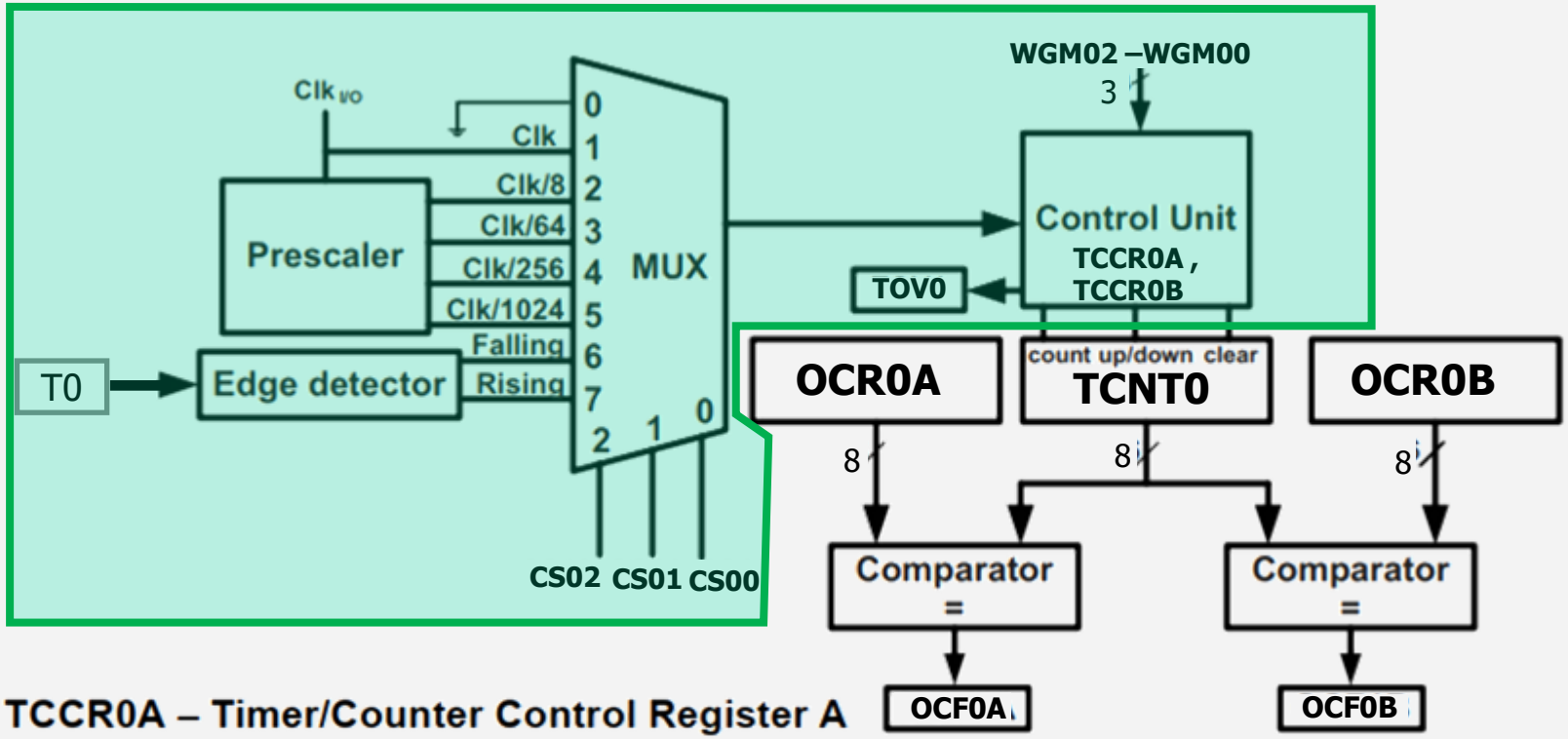
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



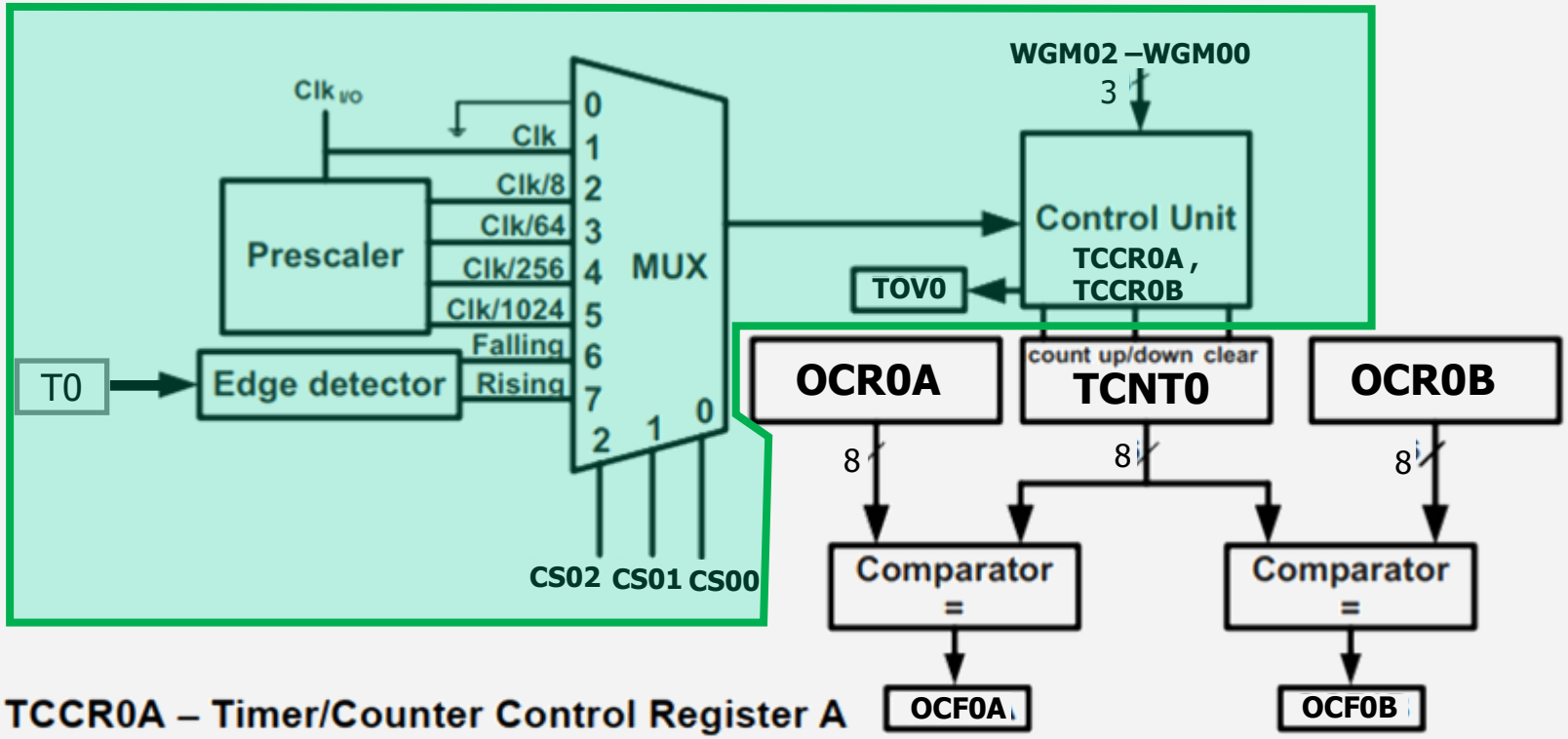
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



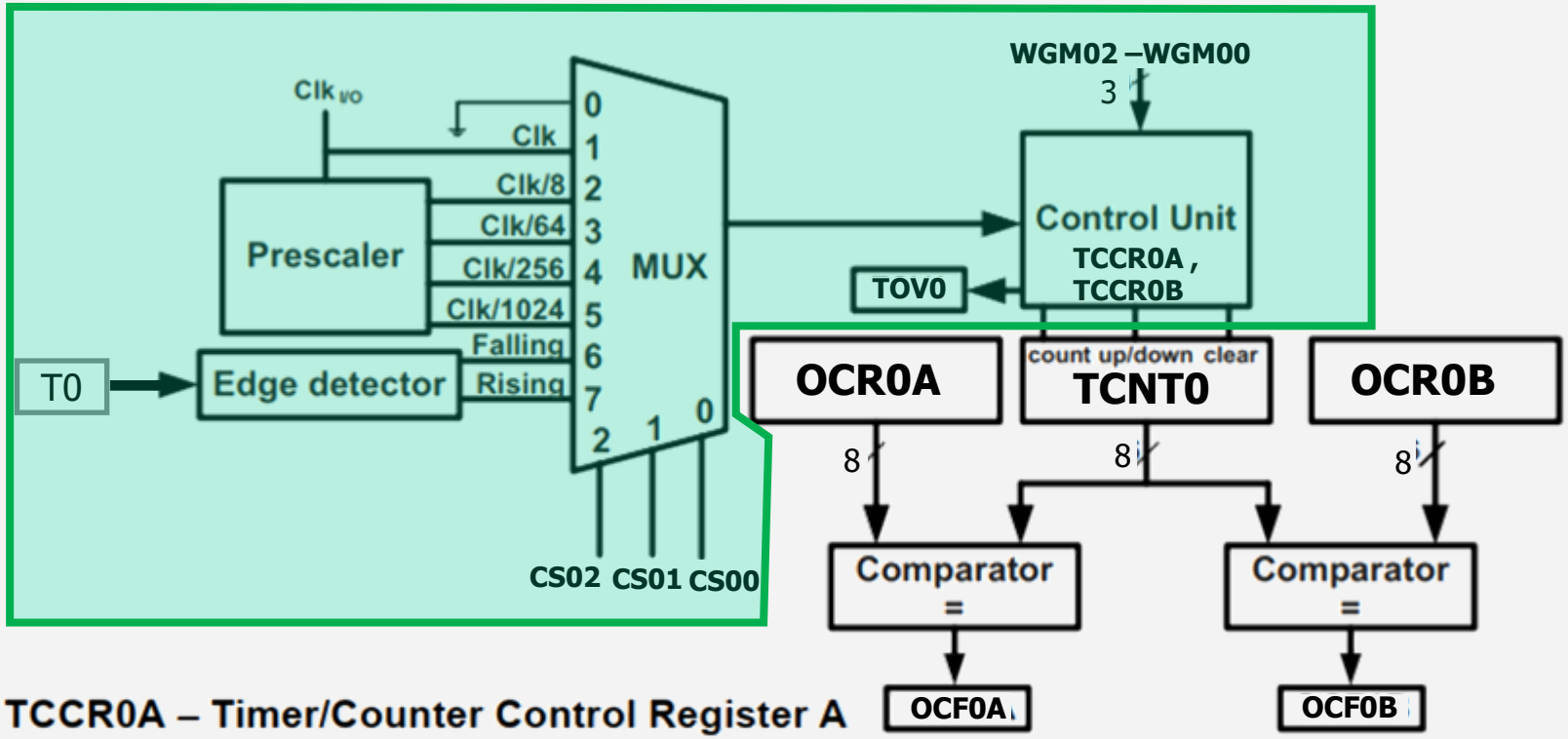
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



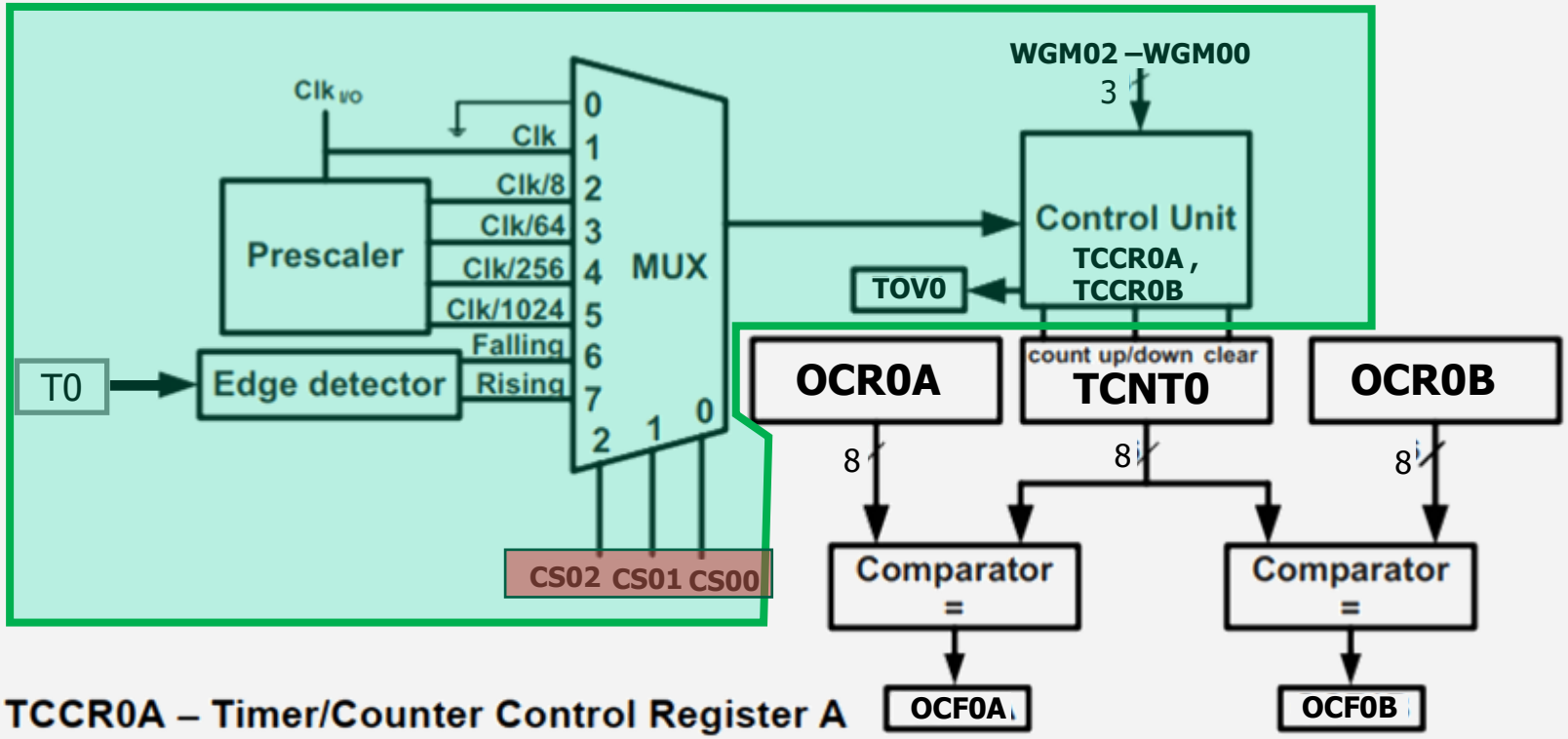
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



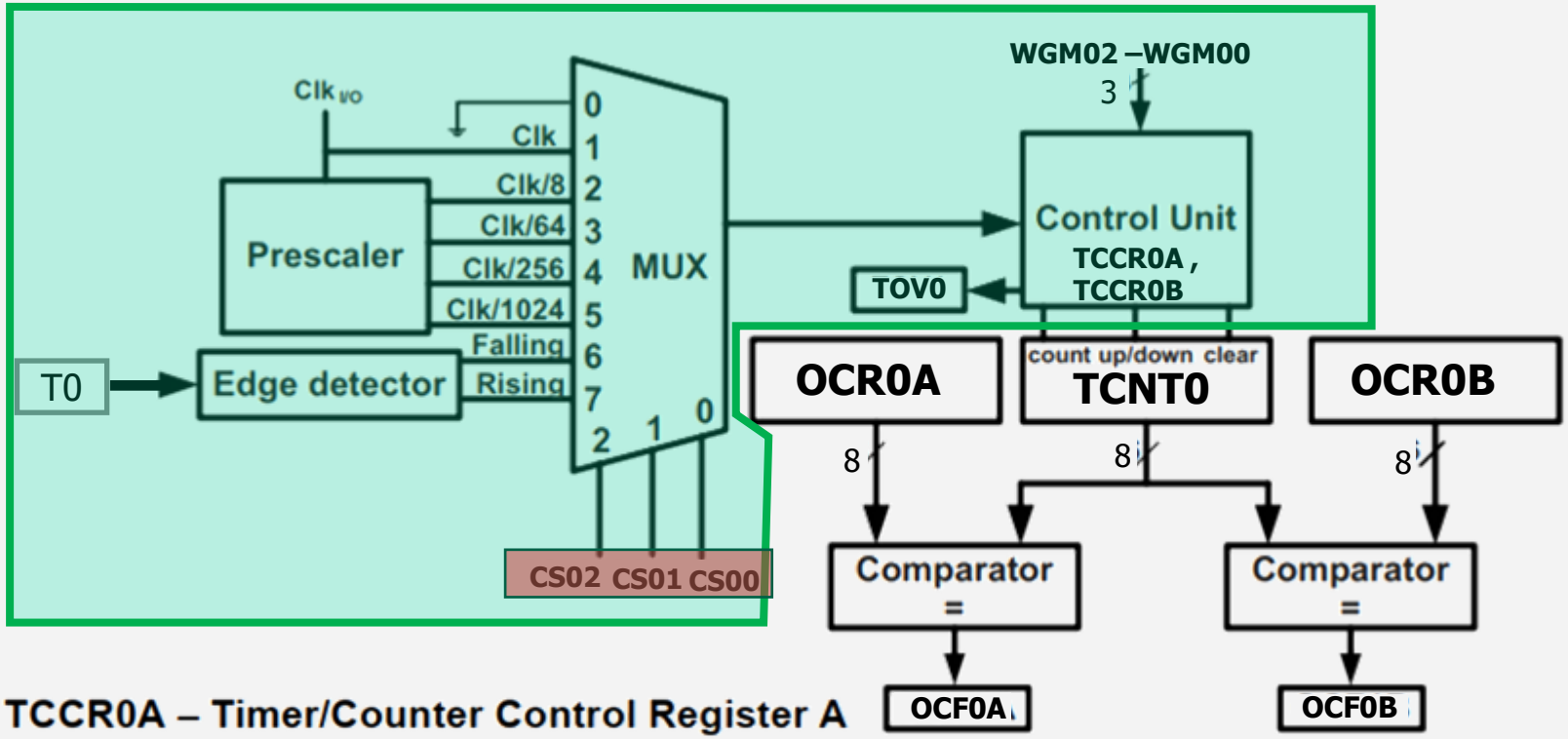
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



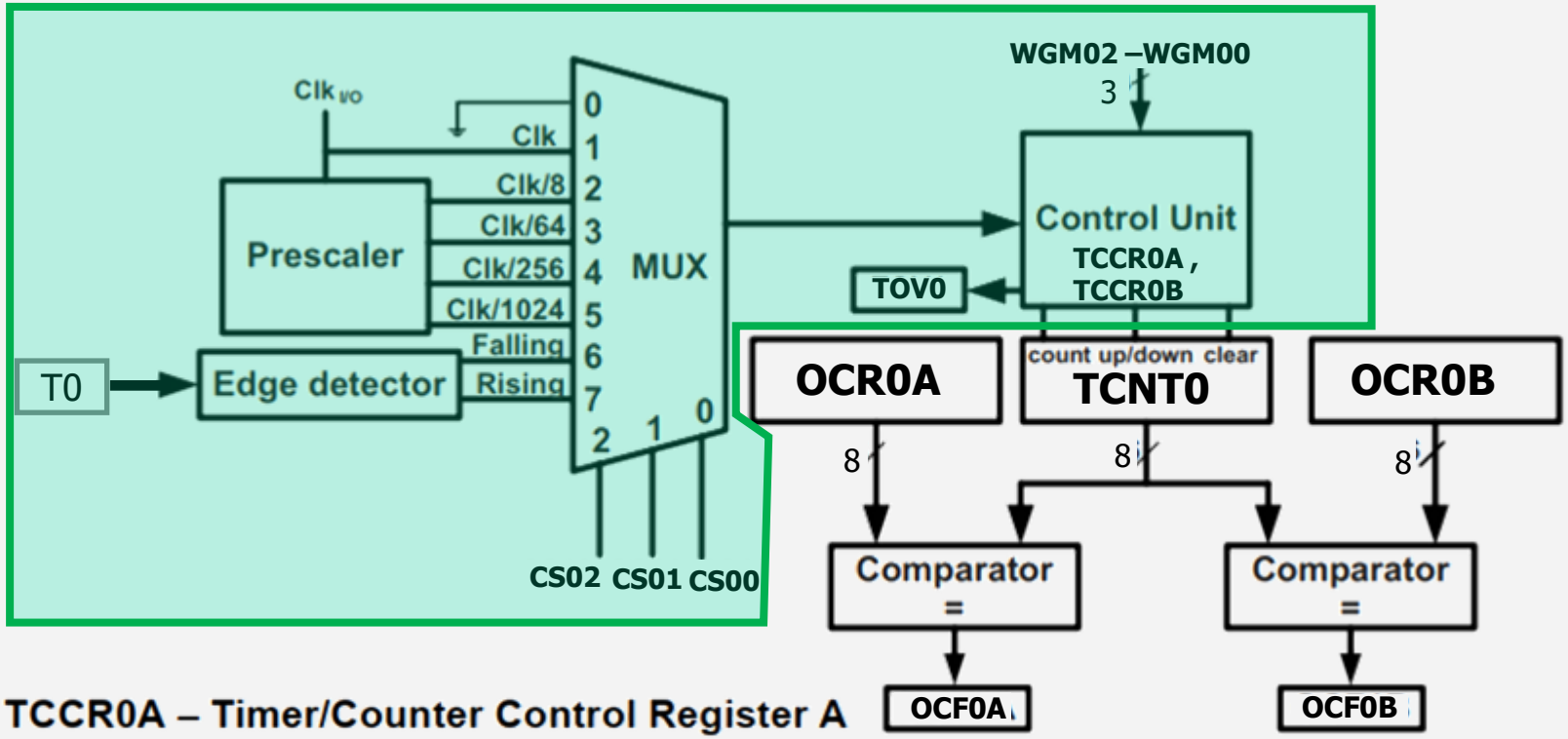
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



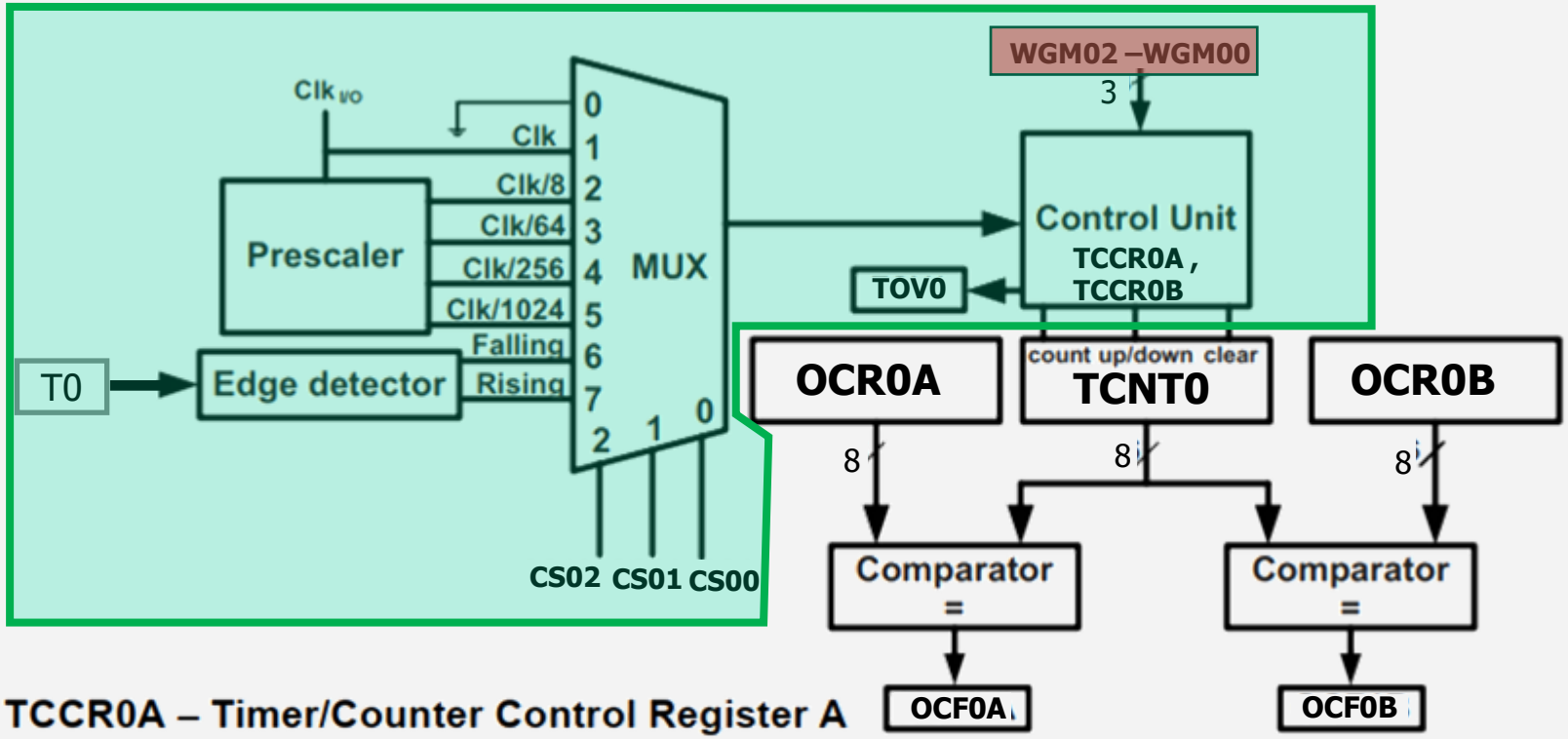
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



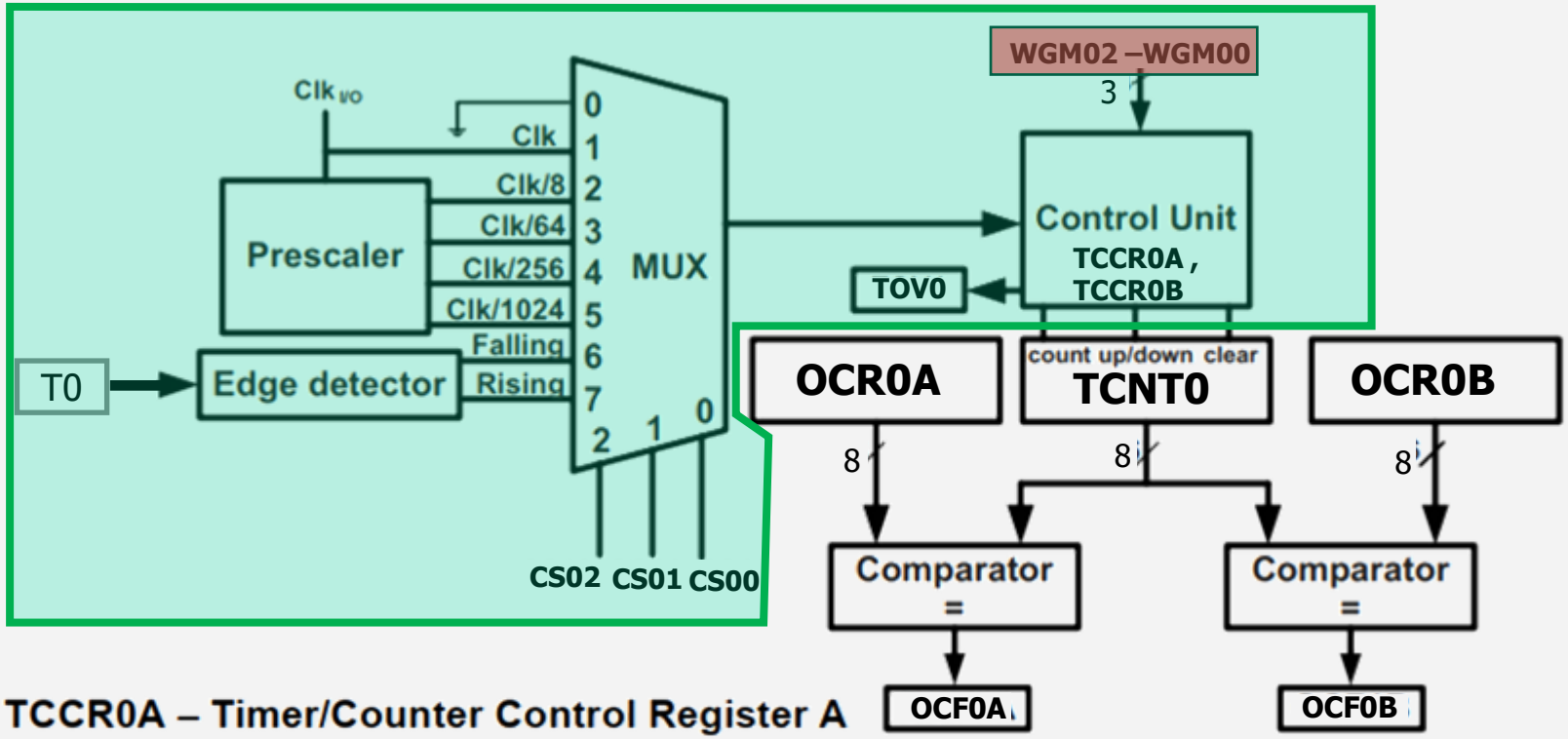
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



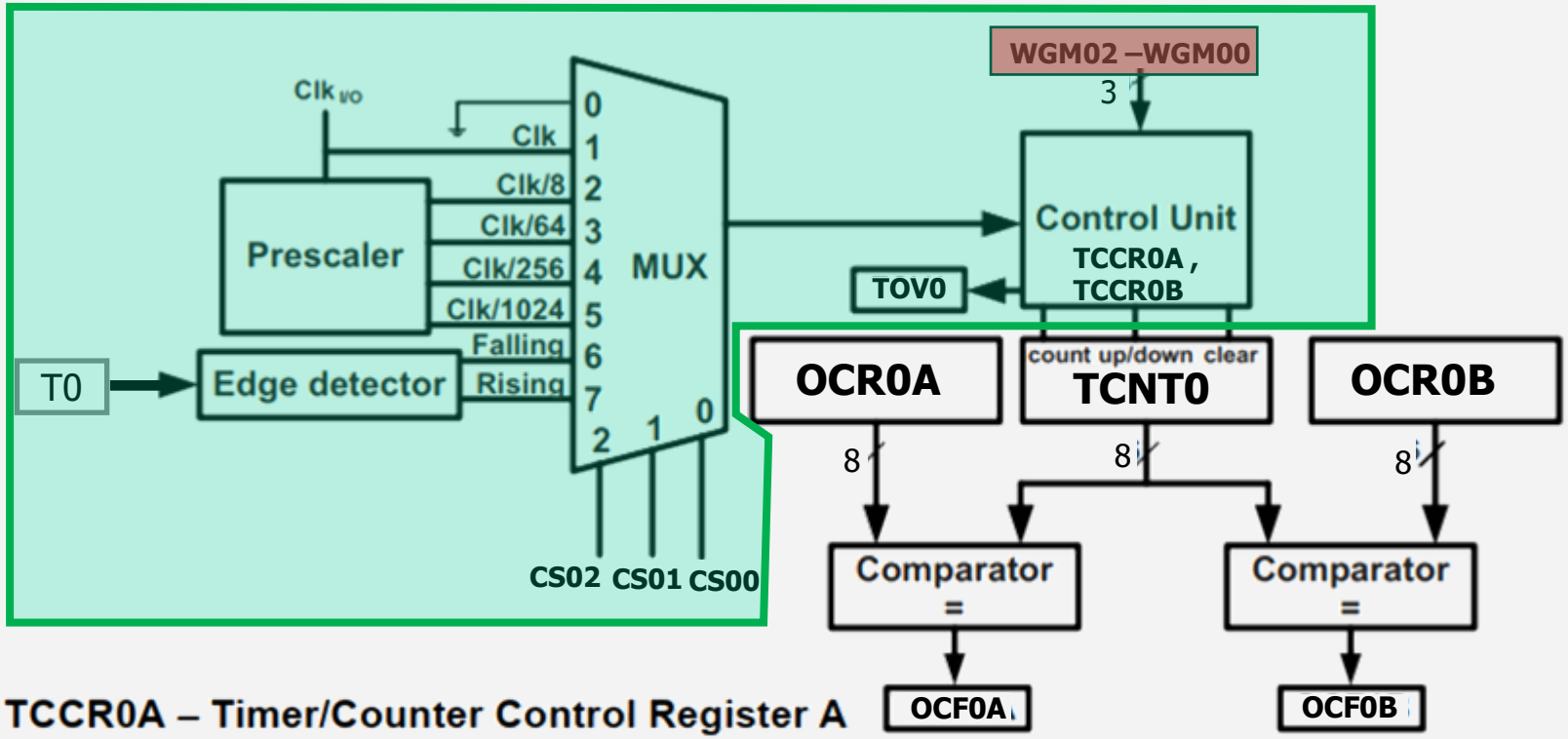
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



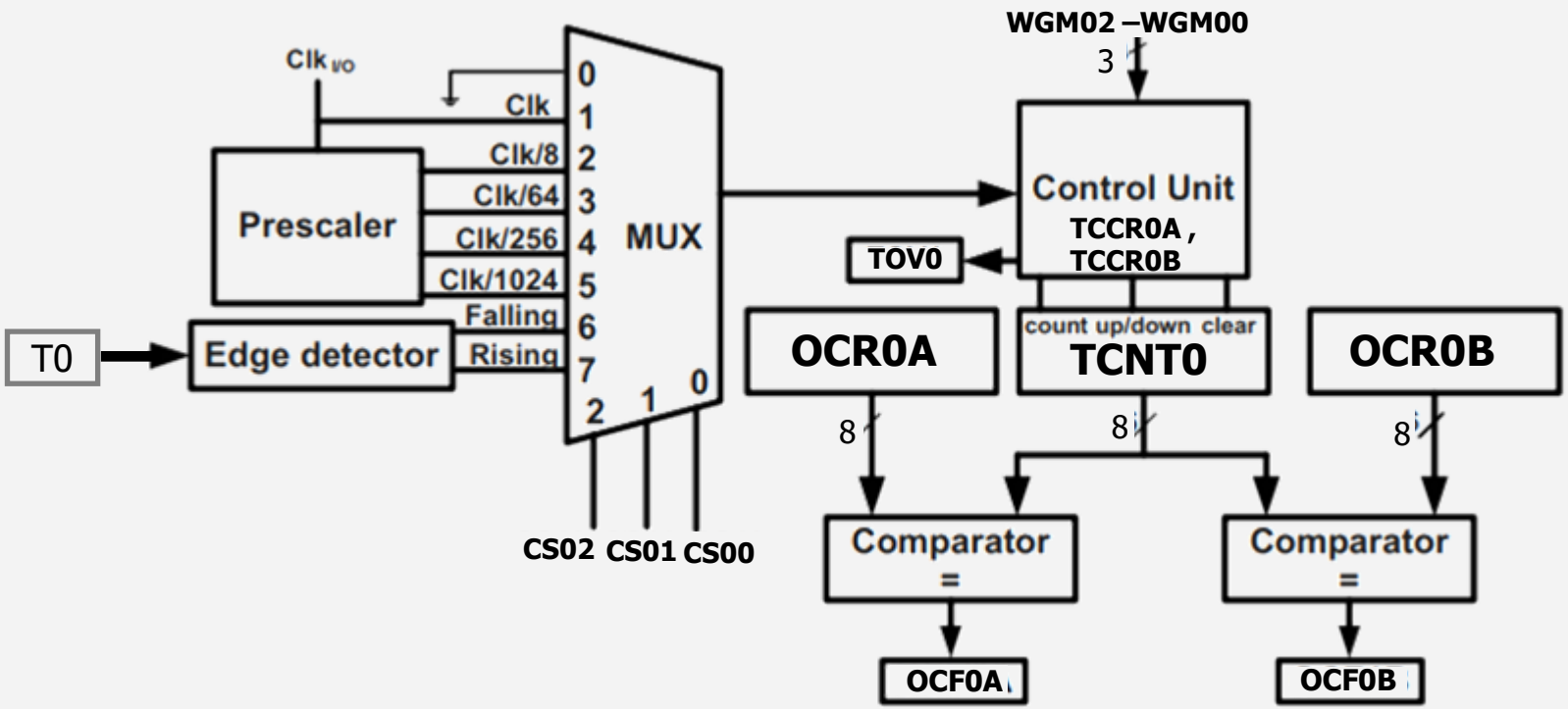
TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W

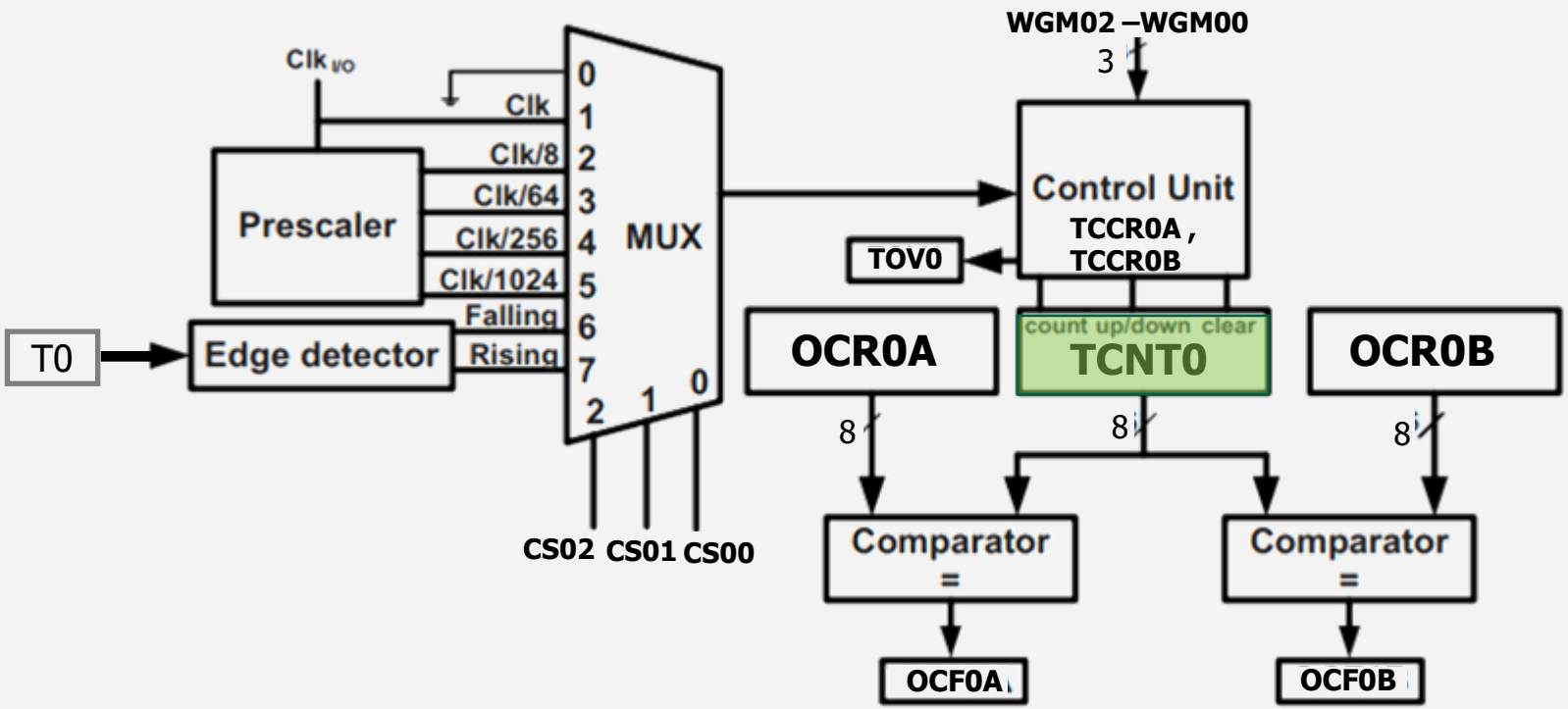
Timer 0 (8-bit timer)



TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0
0x26 (0x46)	TCNT0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

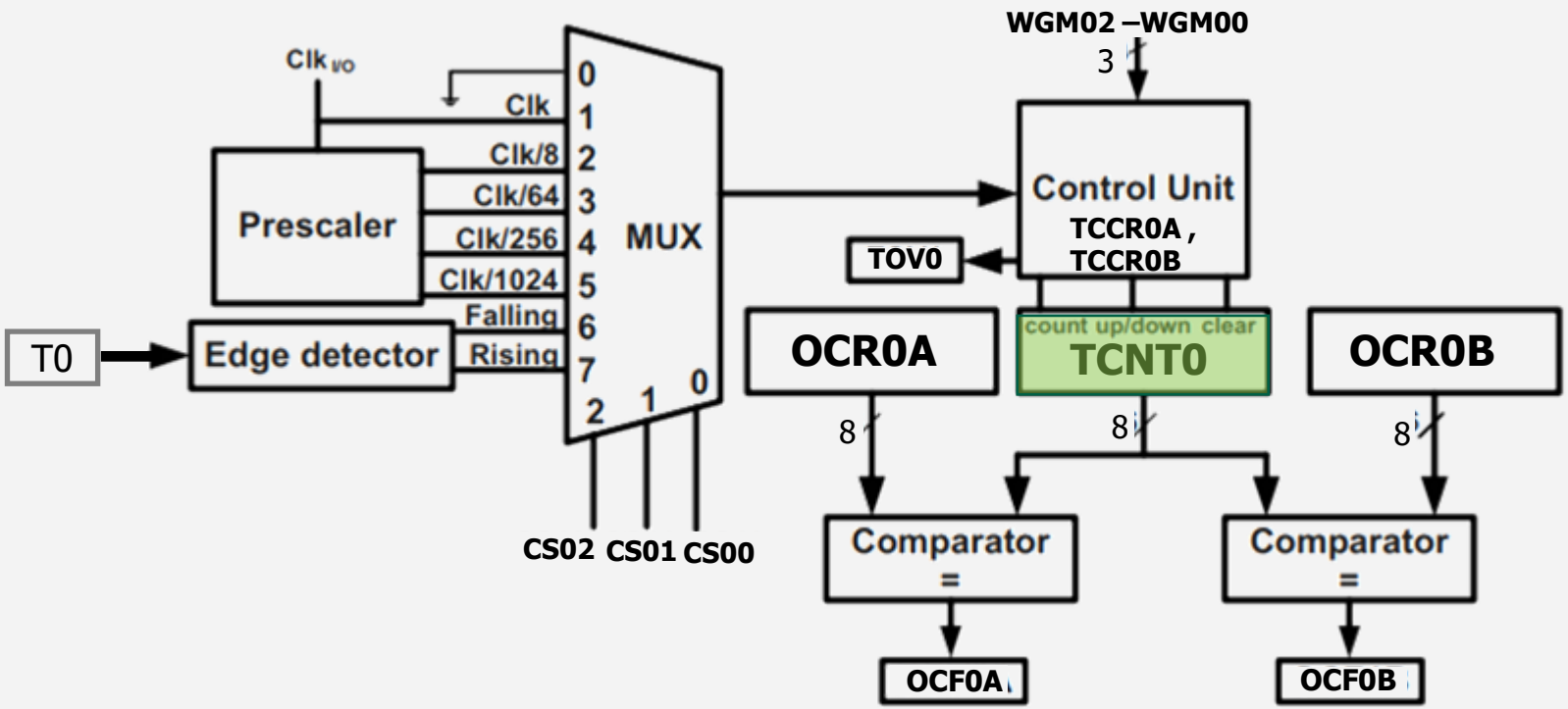
Timer 0 (8-bit timer)



TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0
0x26 (0x46)	TCNT0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

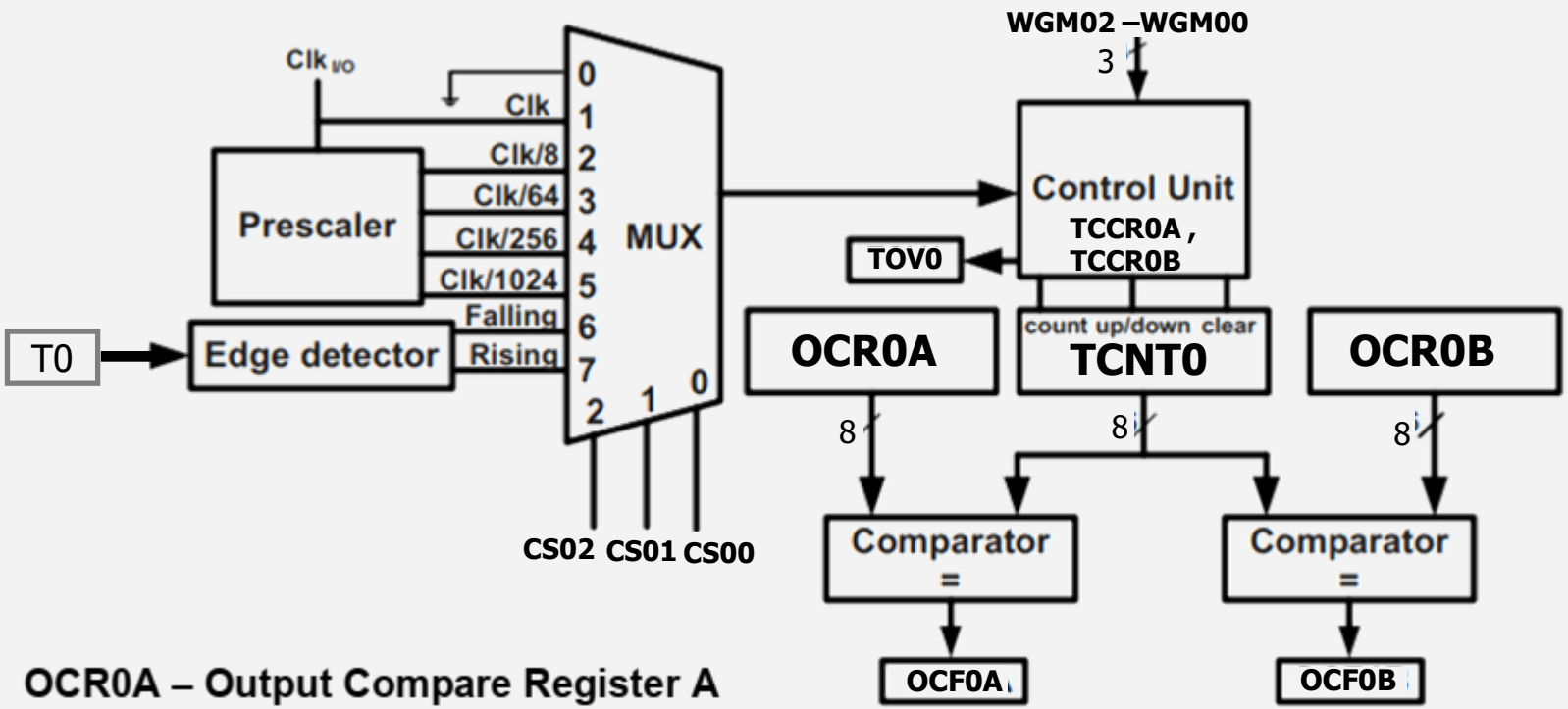
Timer 0 (8-bit timer)



TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0
0x26 (0x46)	TCNT0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



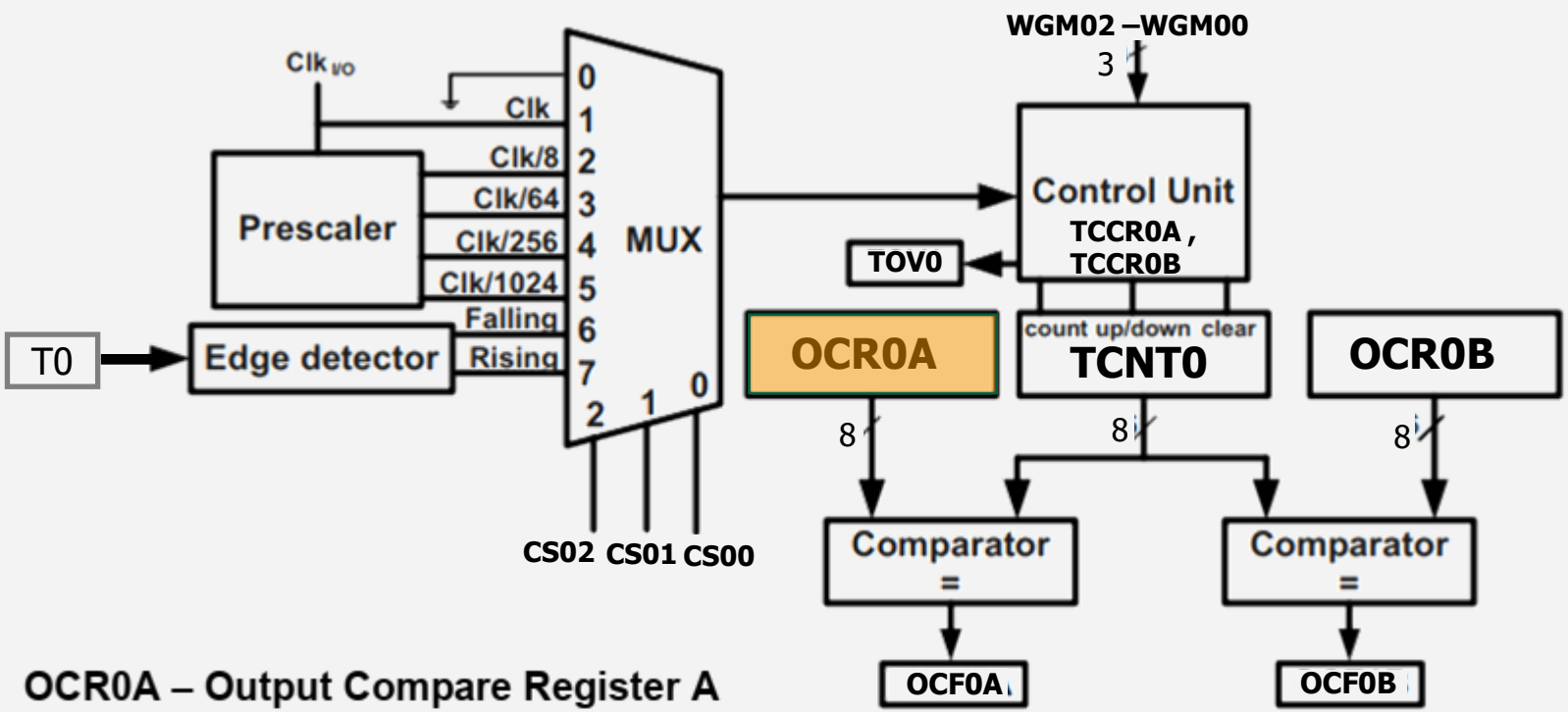
OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0
0x27 (0x47)	OCR0A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0
0x28 (0x48)	OCR0B[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



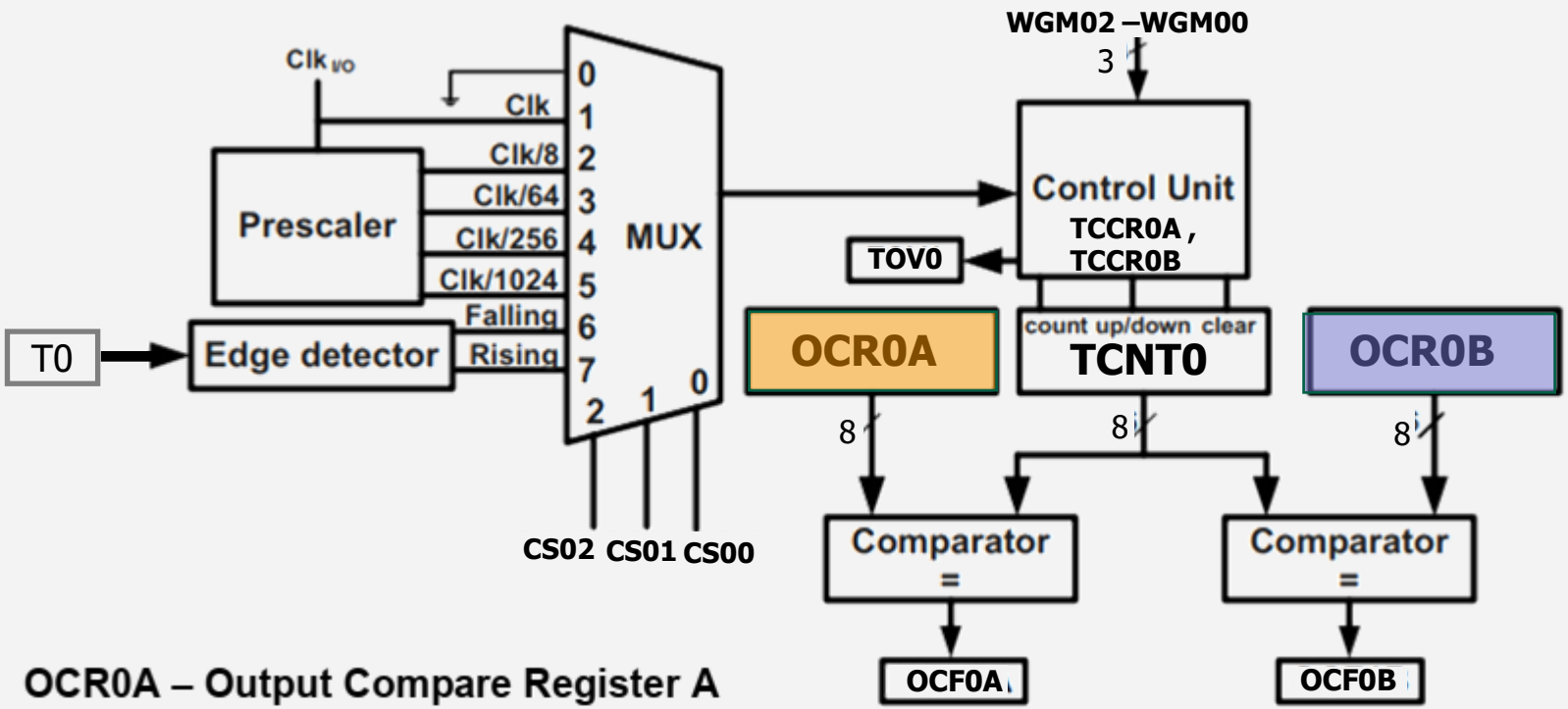
OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0
0x27 (0x47)	OCR0A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0
0x28 (0x48)	OCR0B[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



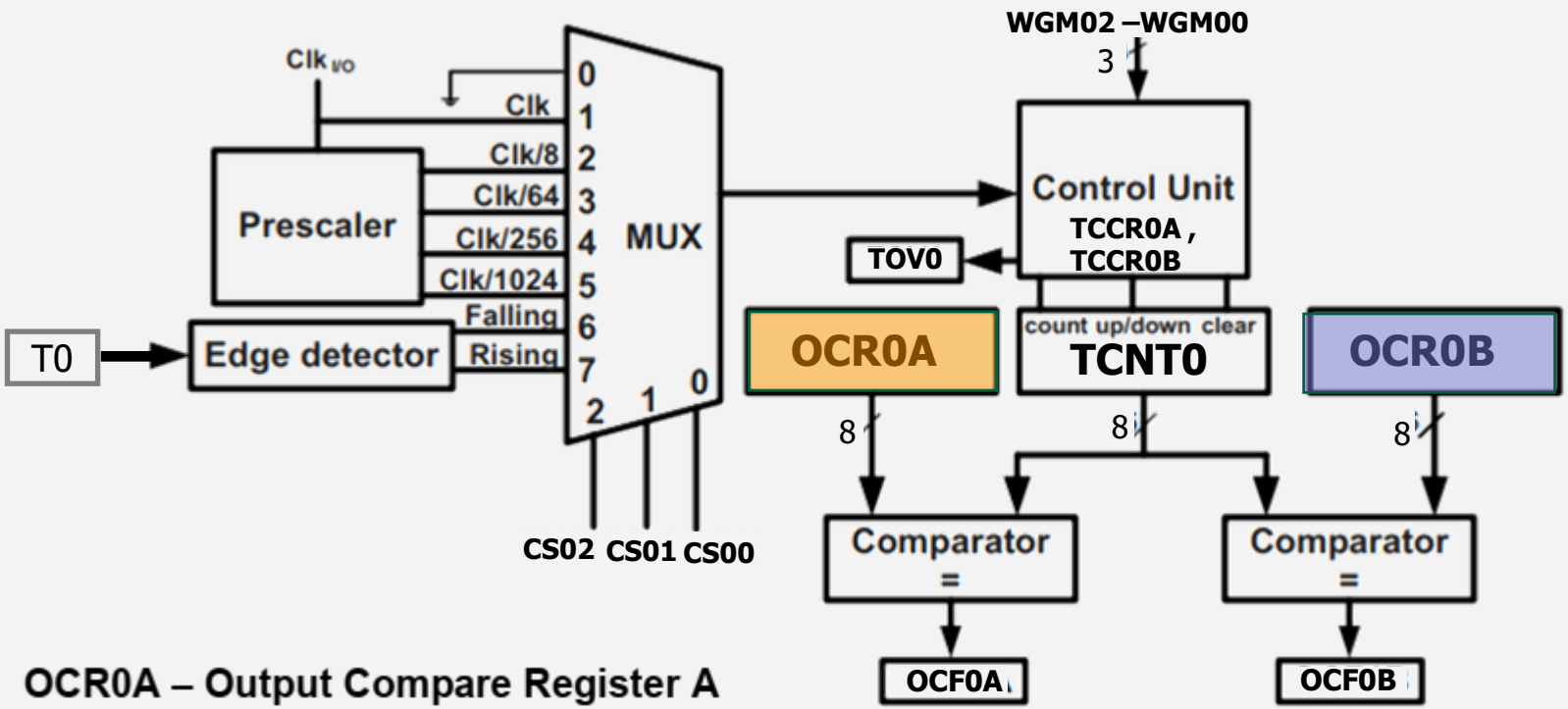
OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0
0x27 (0x47)	OCR0A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0
0x28 (0x48)	OCR0B[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



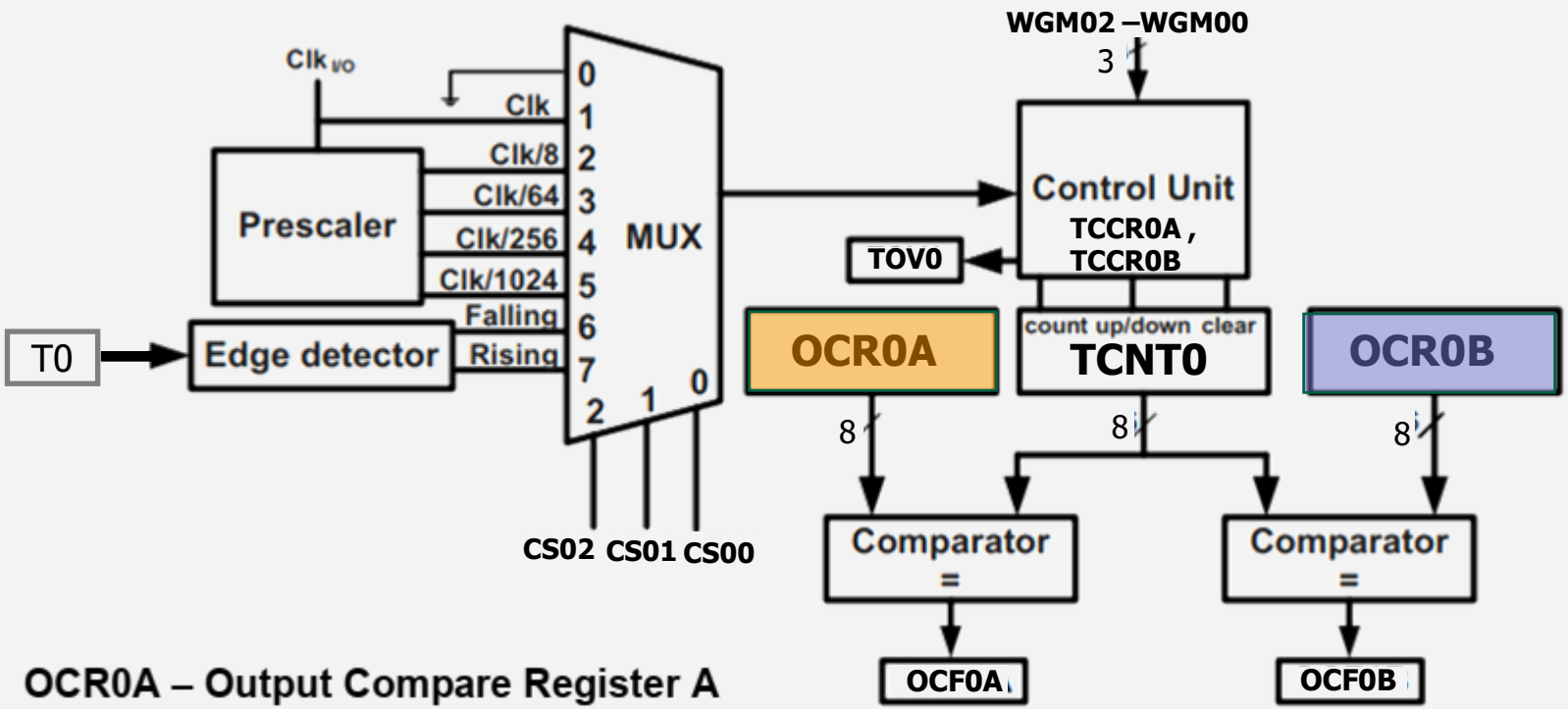
OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0
0x27 (0x47)	OCR0A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0
0x28 (0x48)	OCR0B[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Timer 0 (8-bit timer)



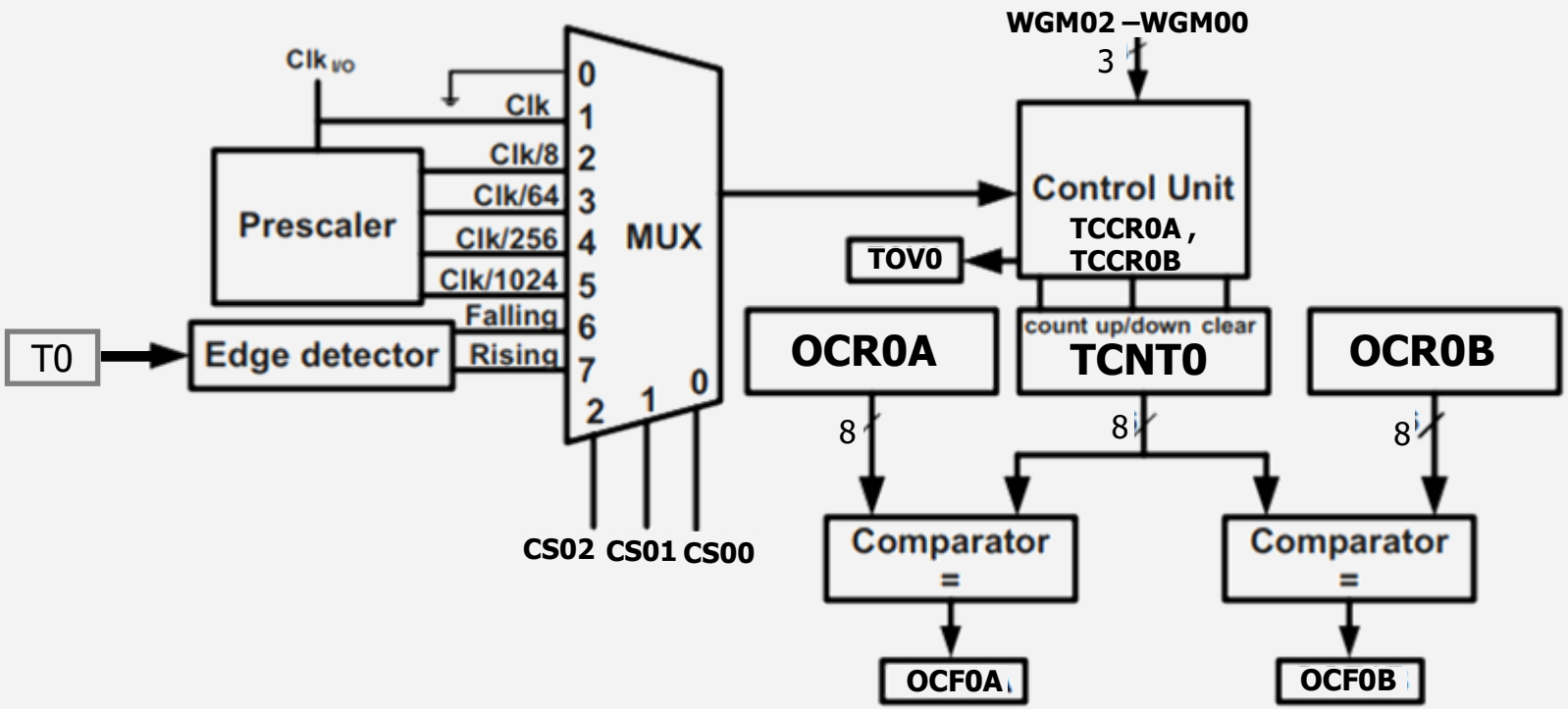
OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0
0x27 (0x47)	OCR0A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0
0x28 (0x48)	OCR0B[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

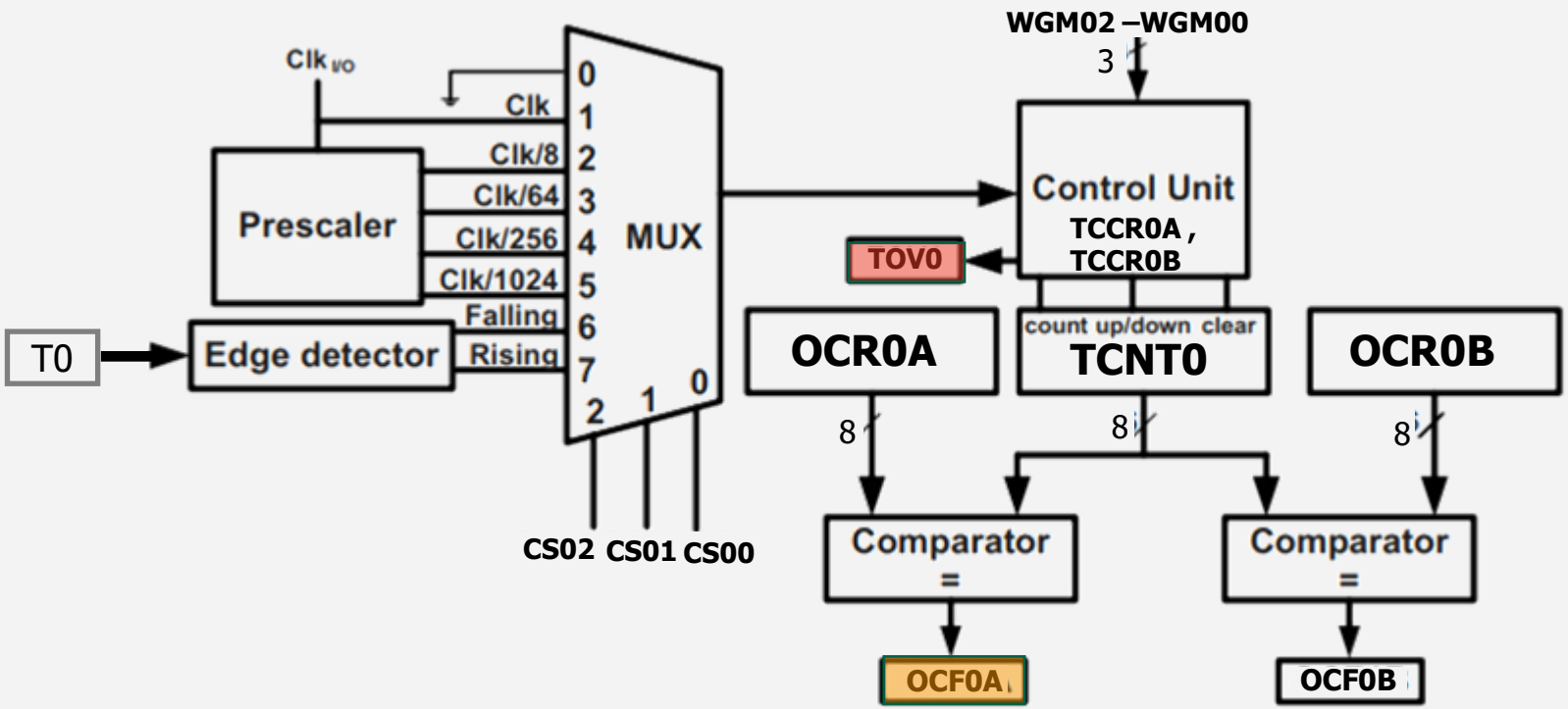
Timer 0 (8-bit timer)



TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

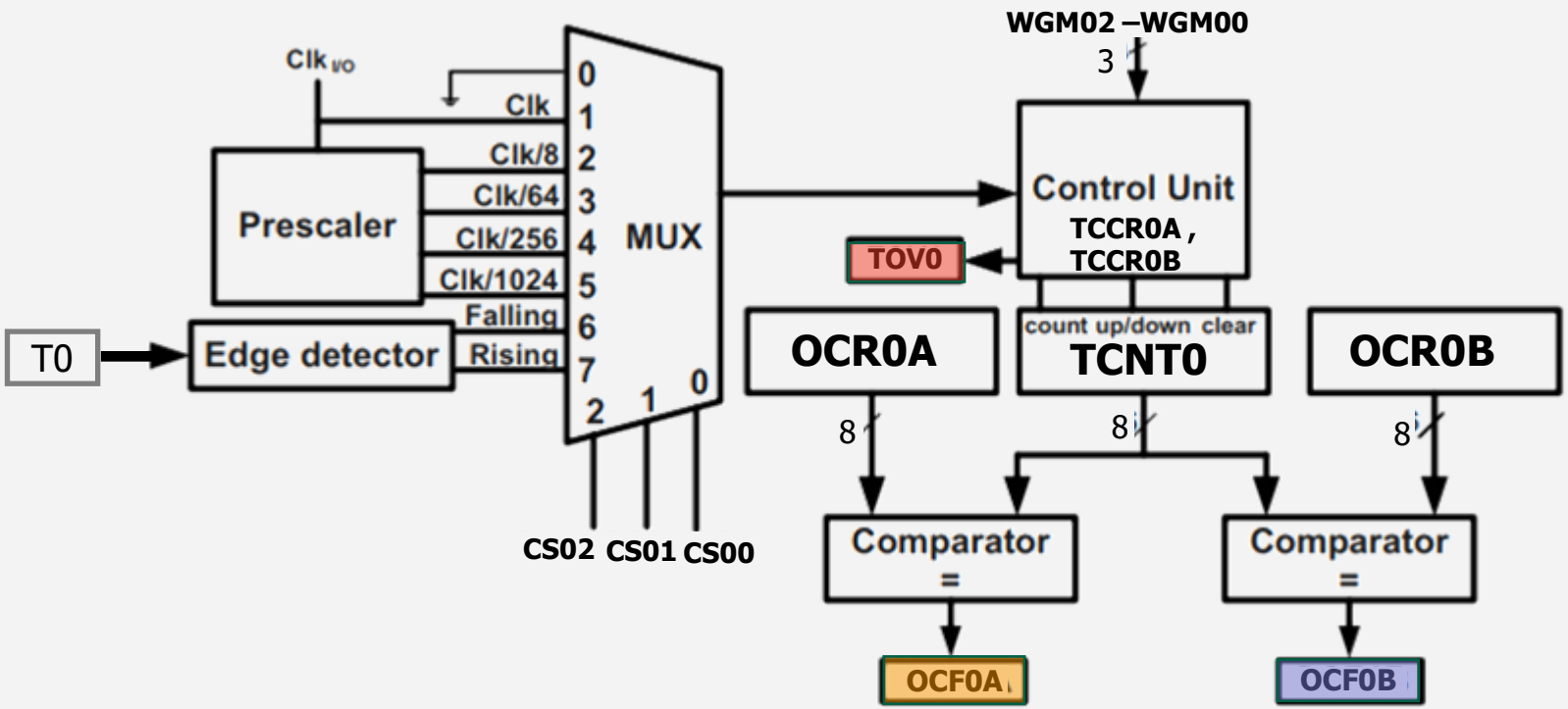
Timer 0 (8-bit timer)



TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

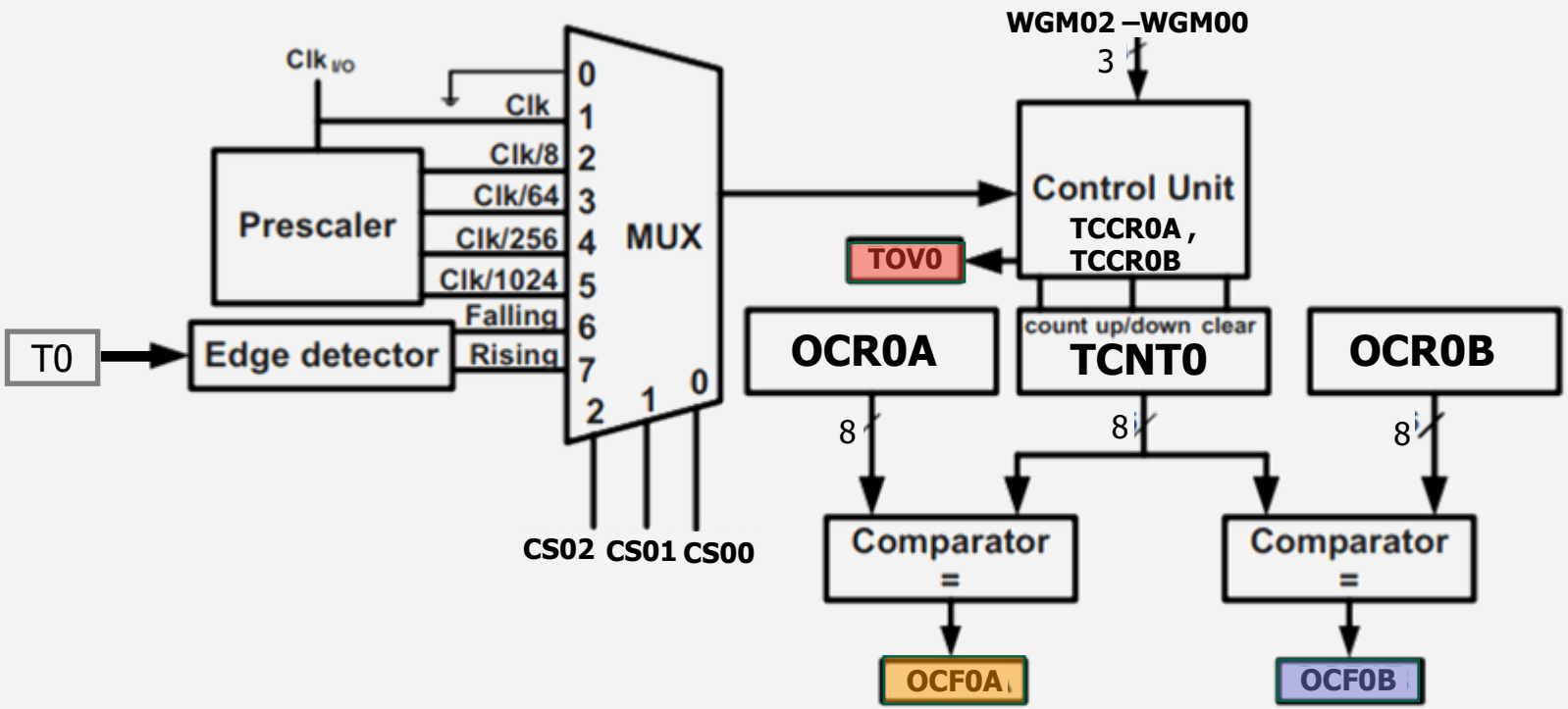
Timer 0 (8-bit timer)



TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

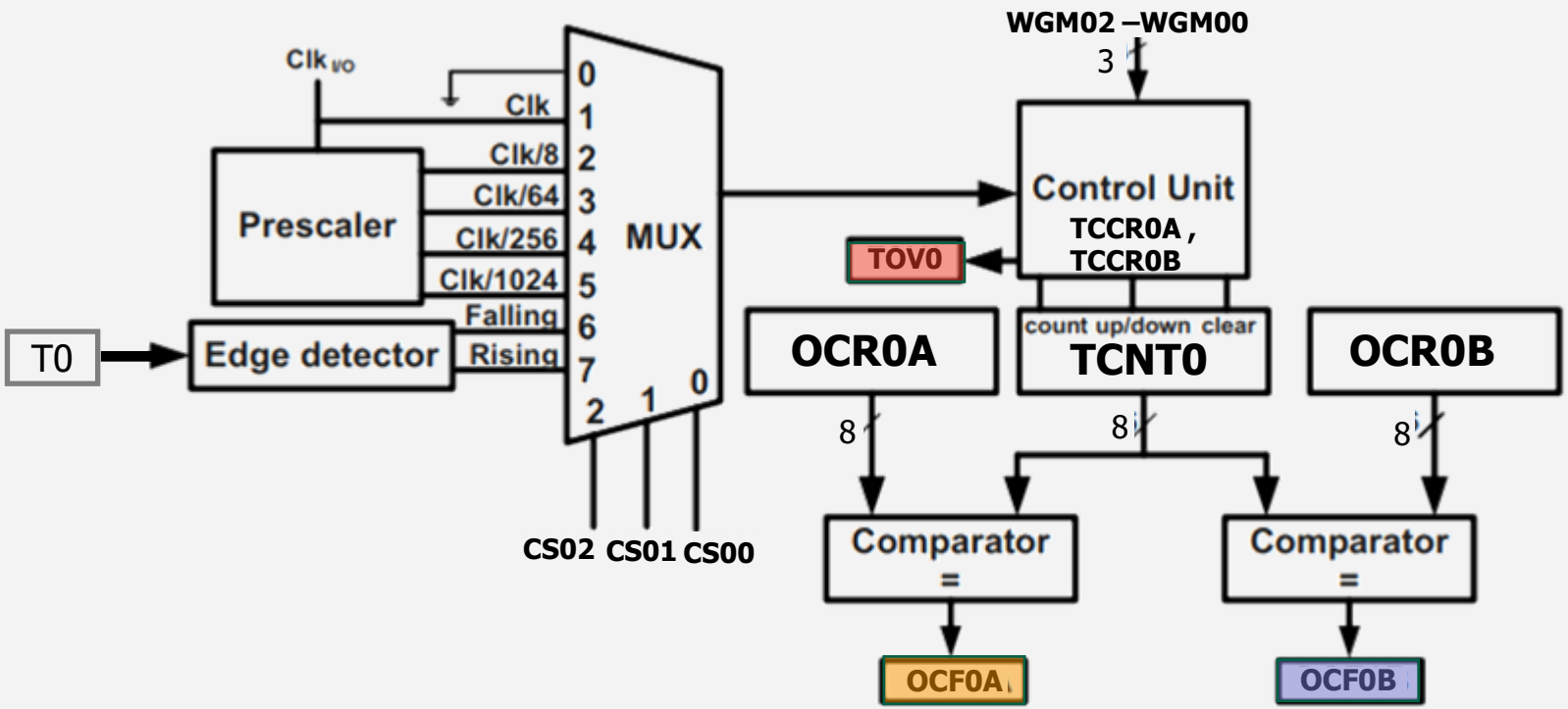
Timer 0 (8-bit timer)



TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

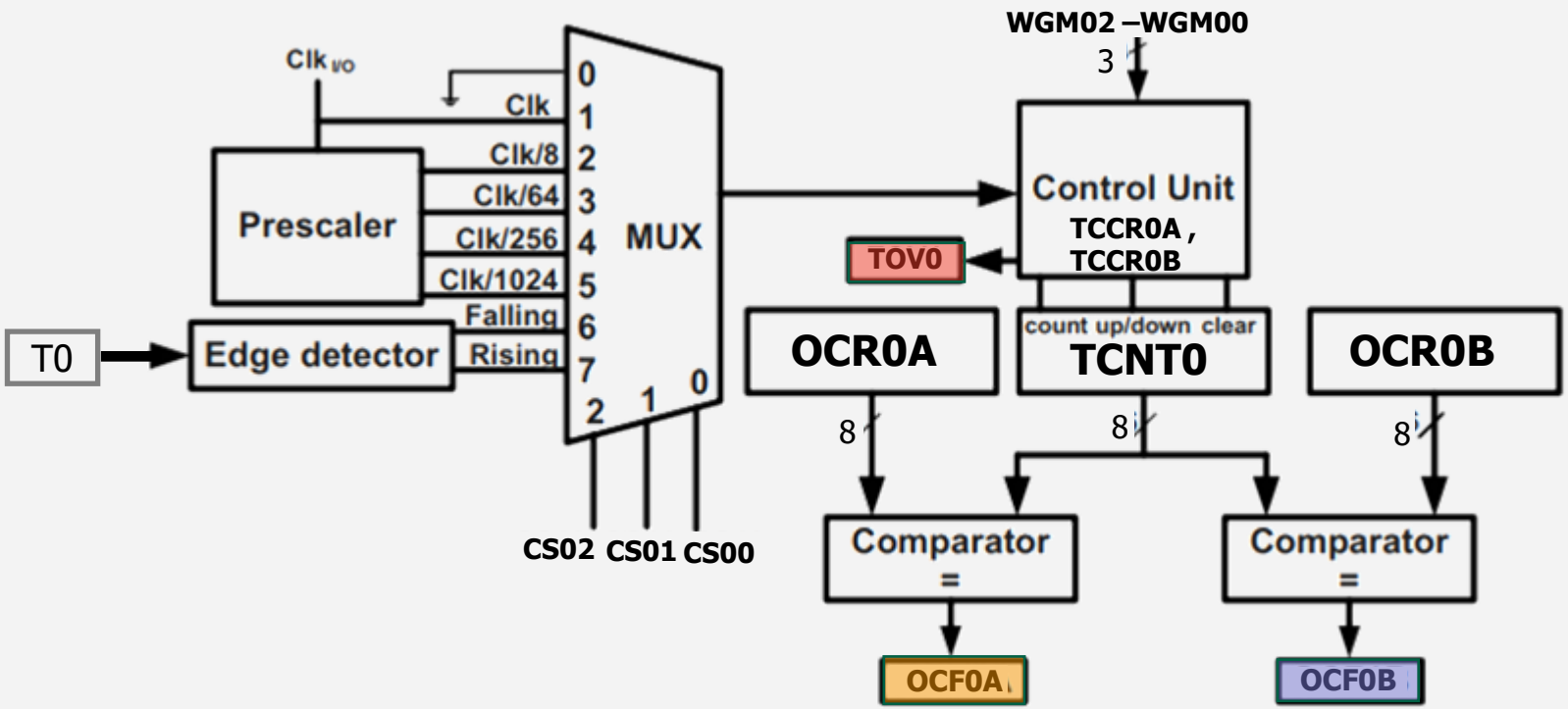
Timer 0 (8-bit timer)



TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

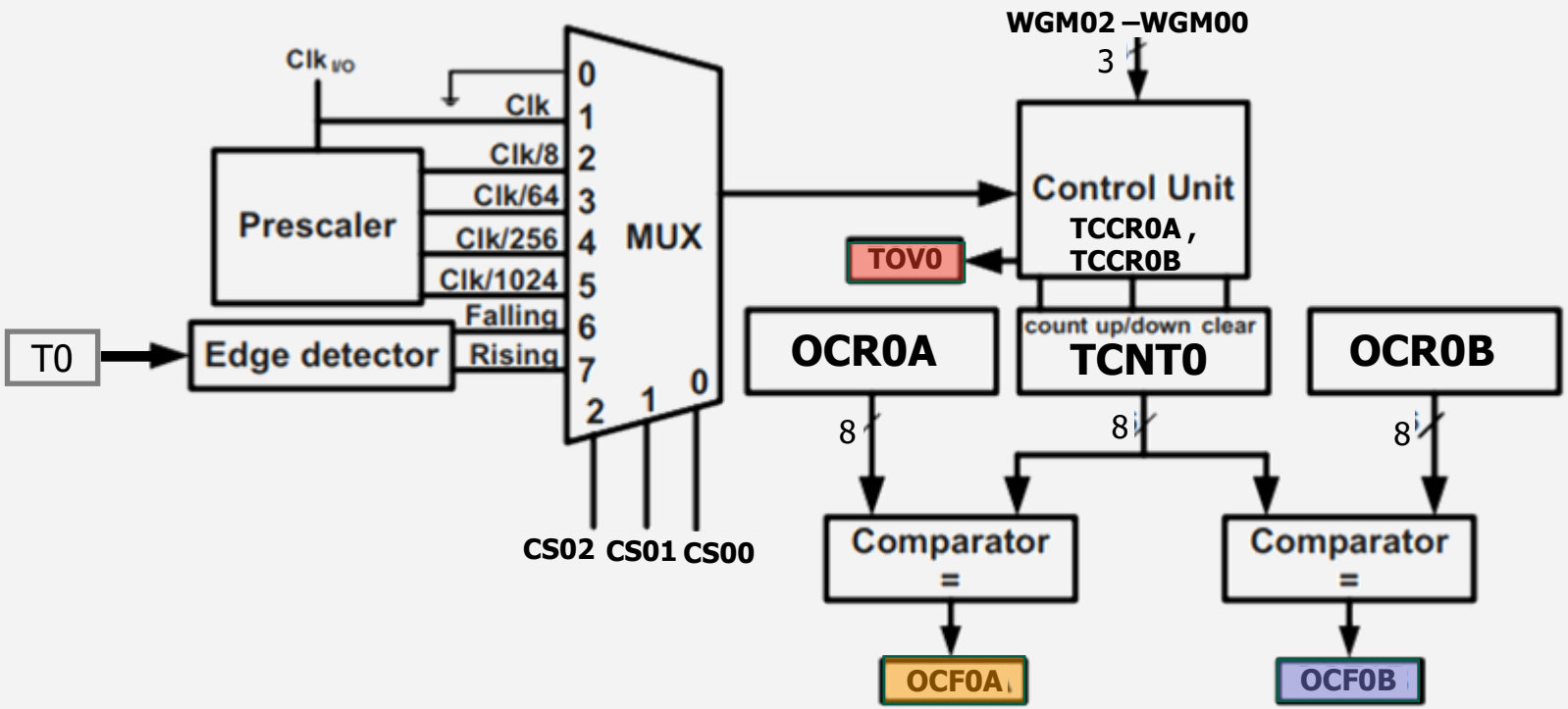
Timer 0 (8-bit timer)



TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

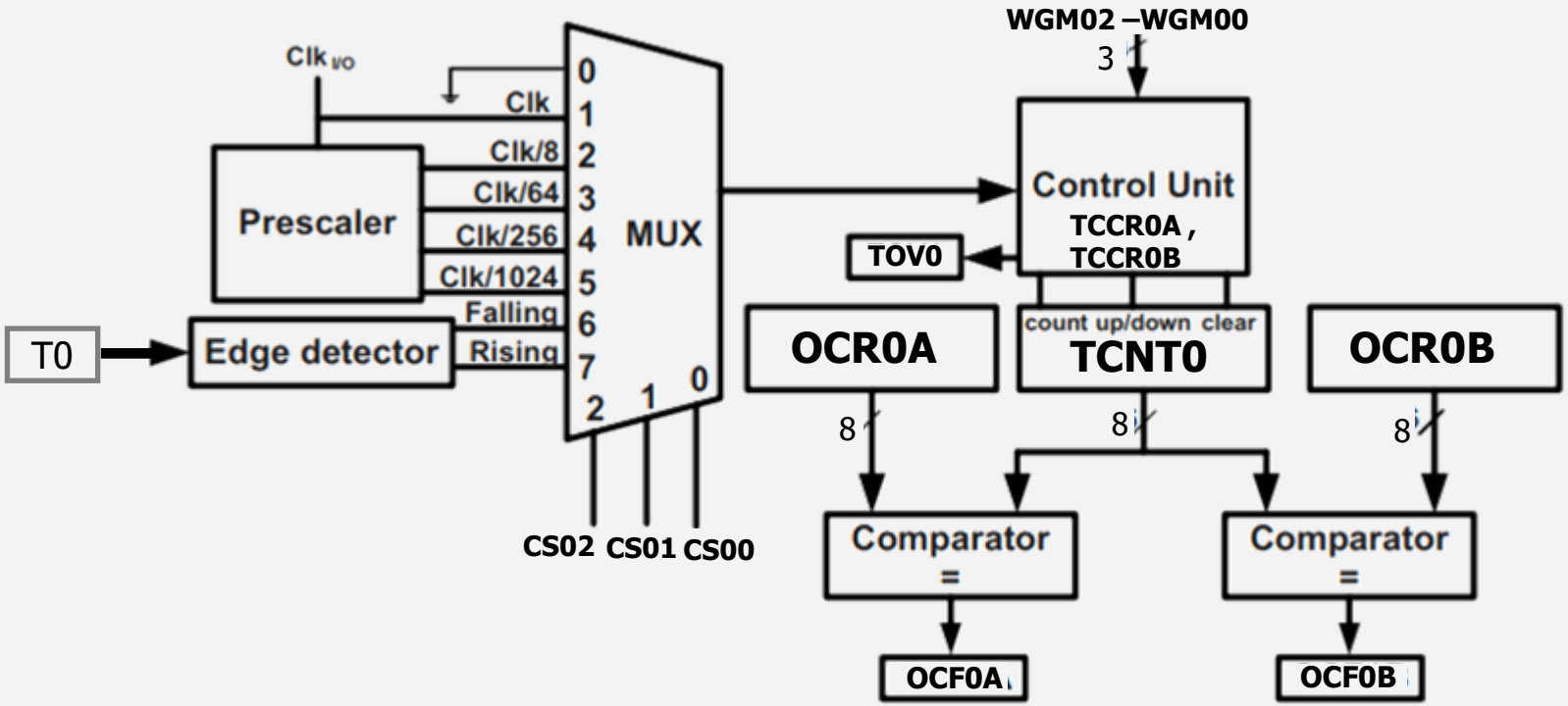
Timer 0 (8-bit timer)



TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

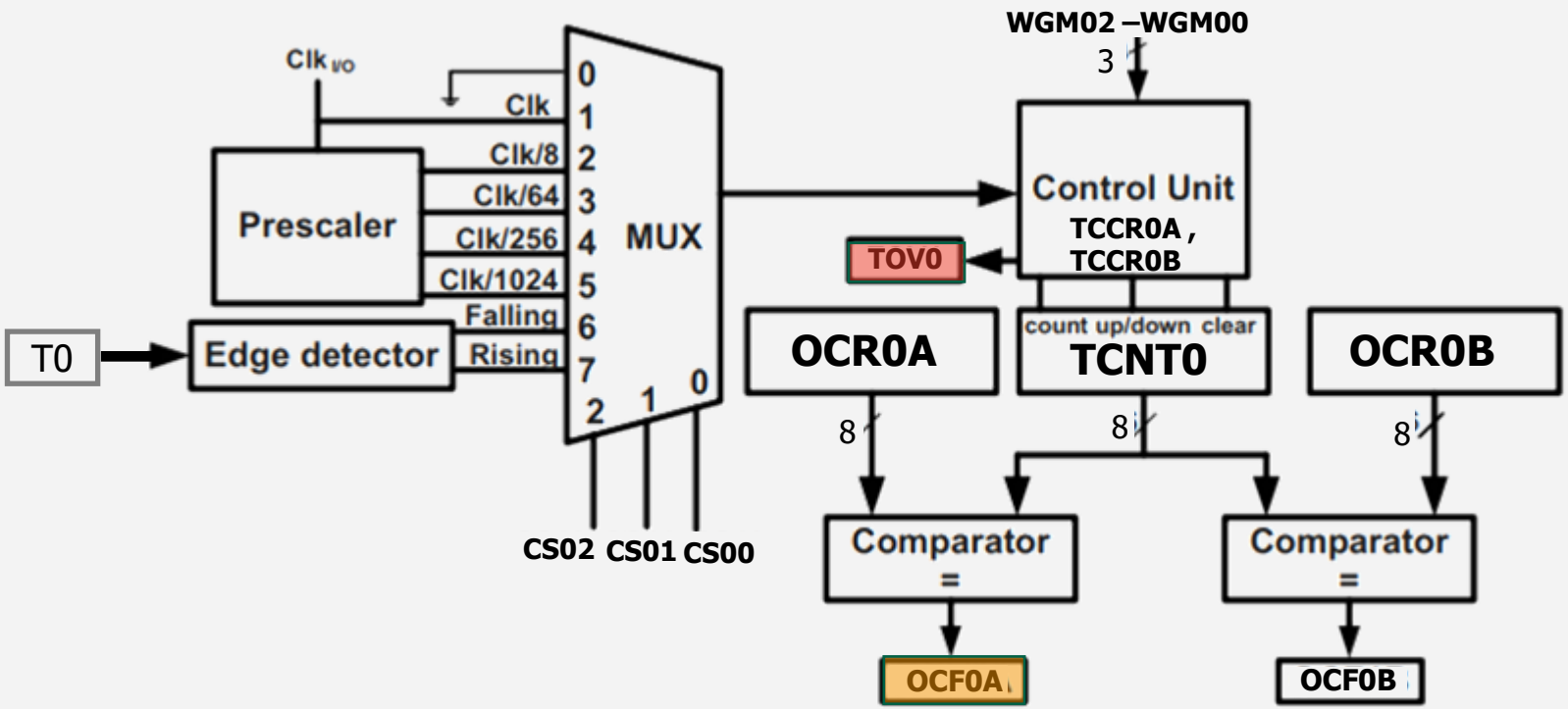
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

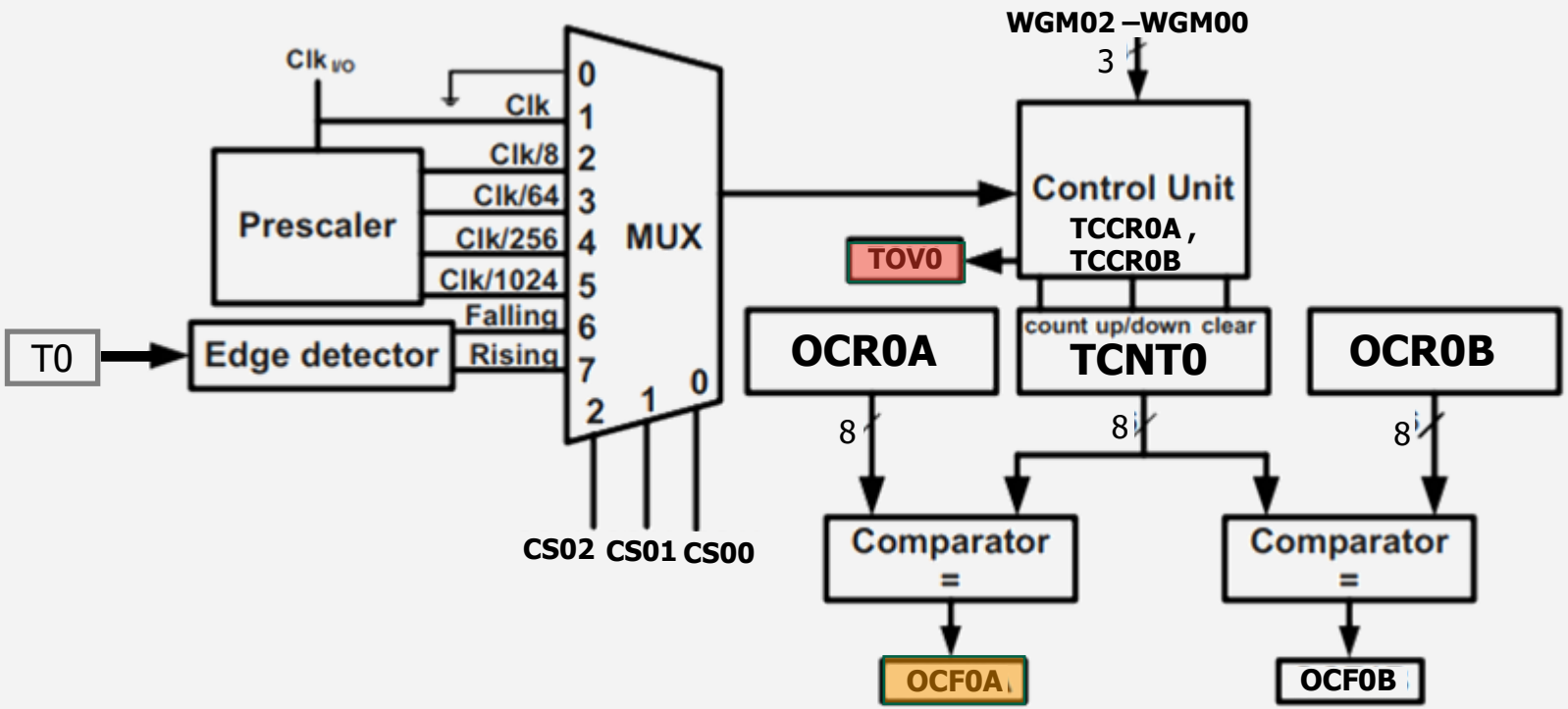
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

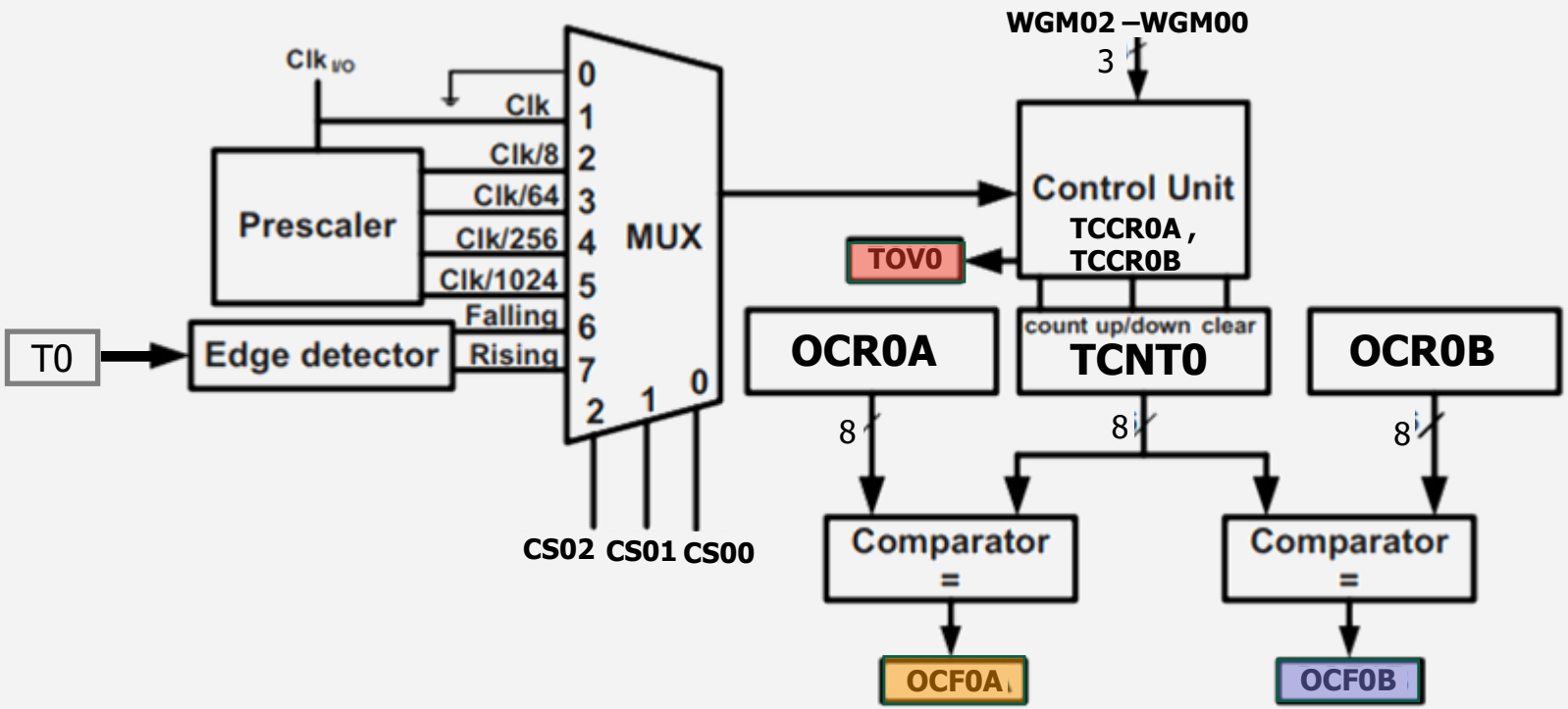
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

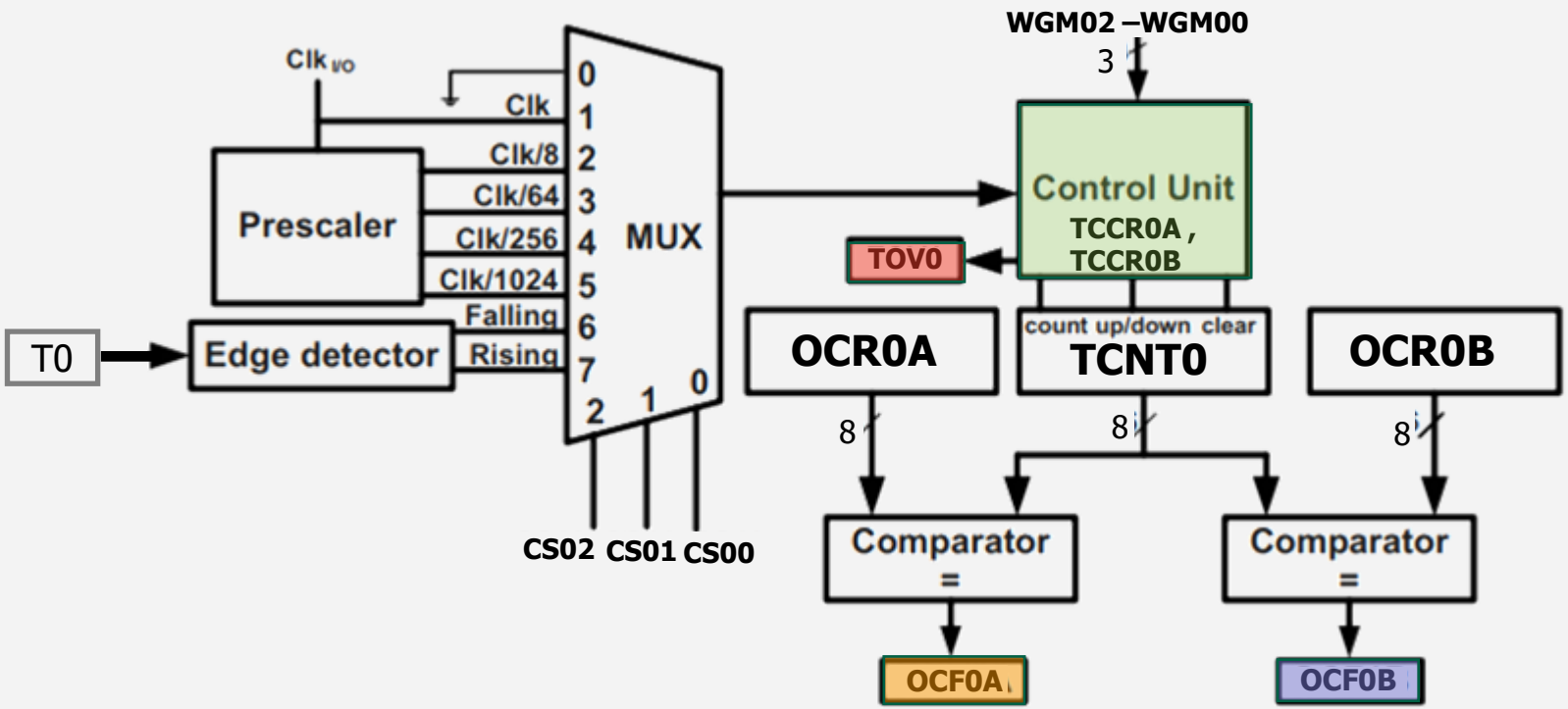
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

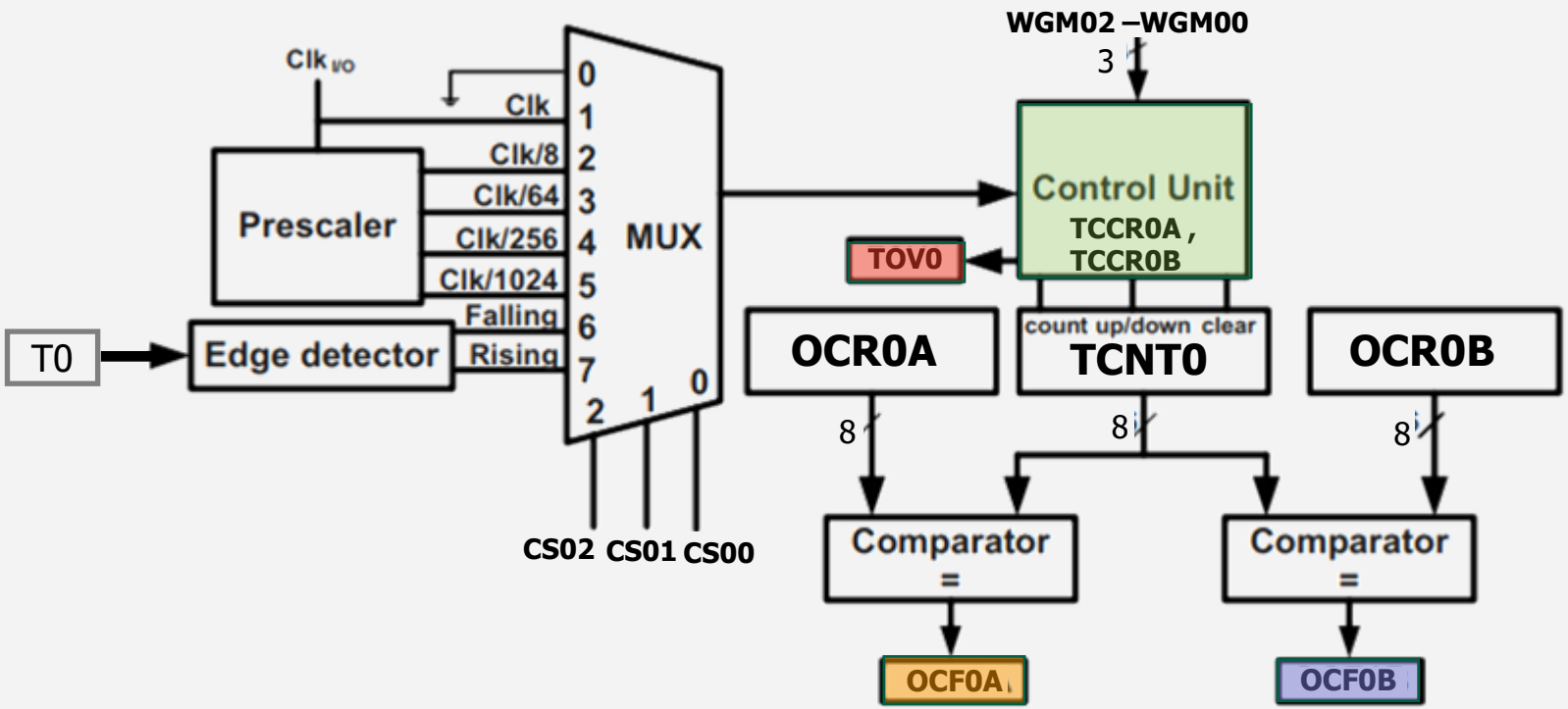
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

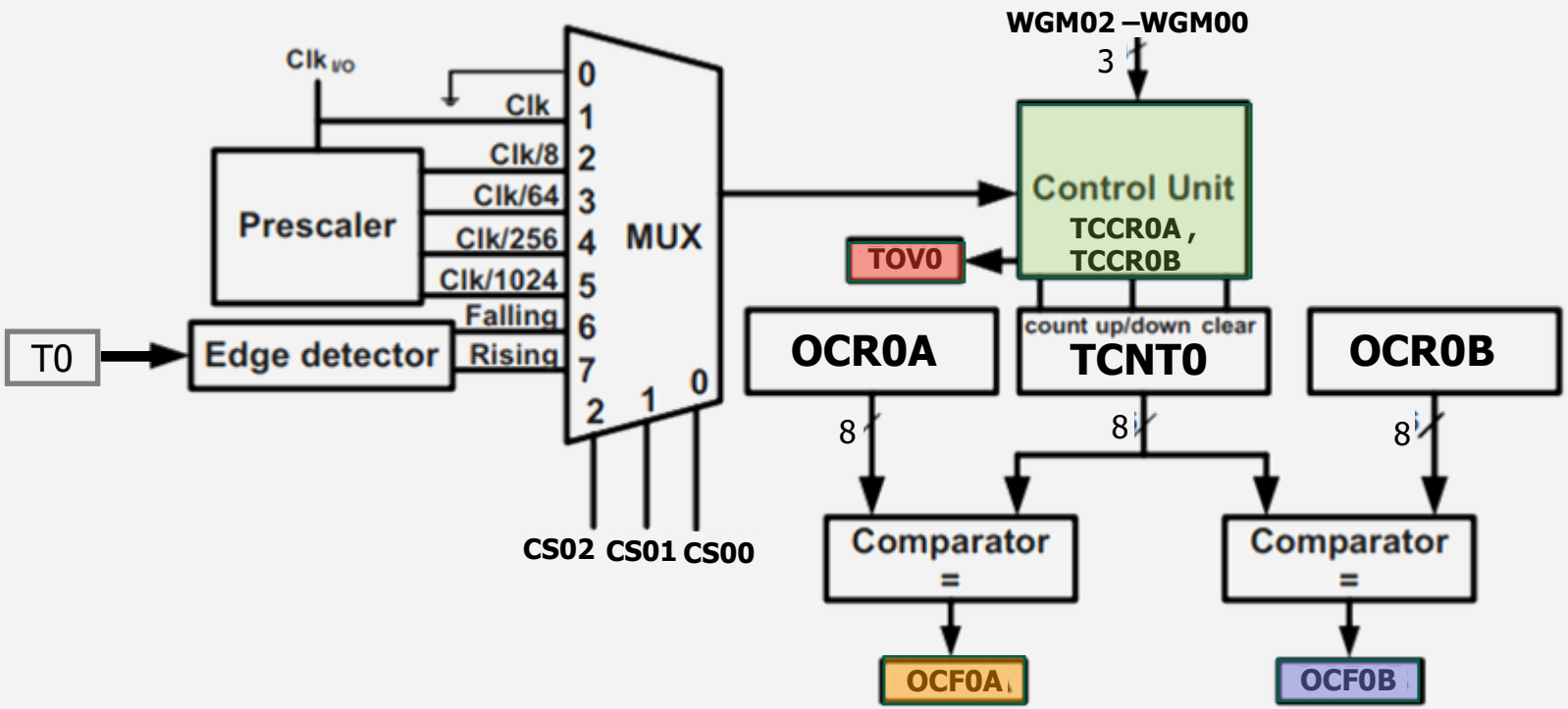
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

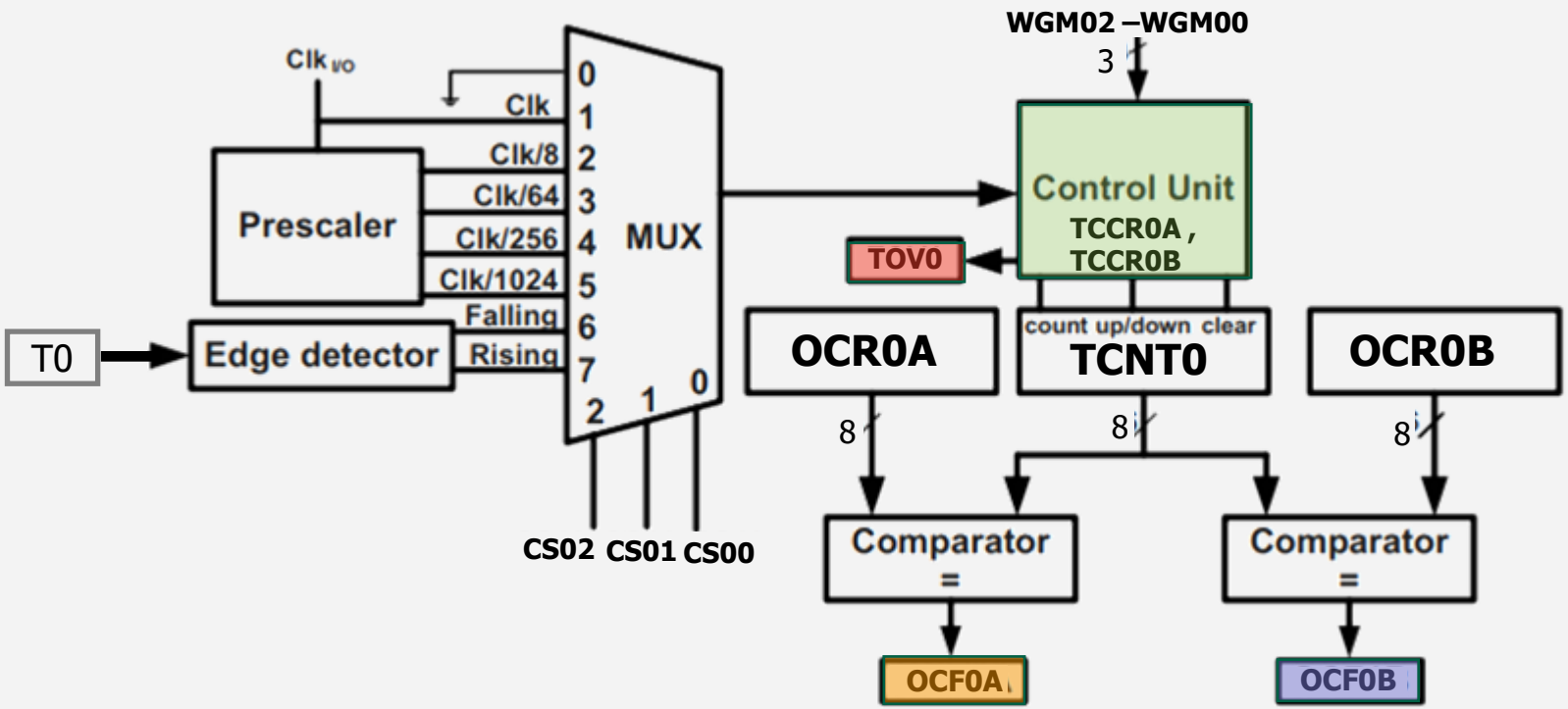
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

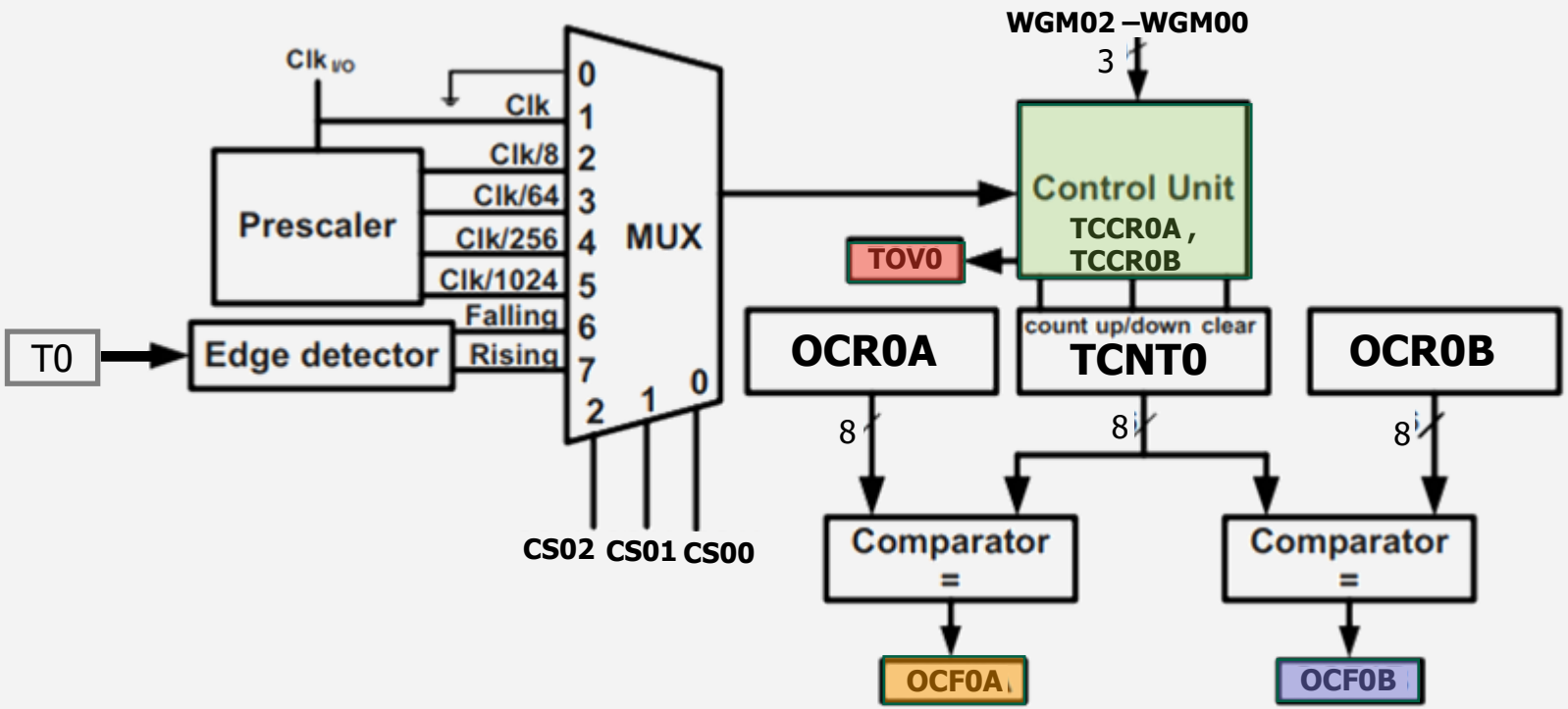
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

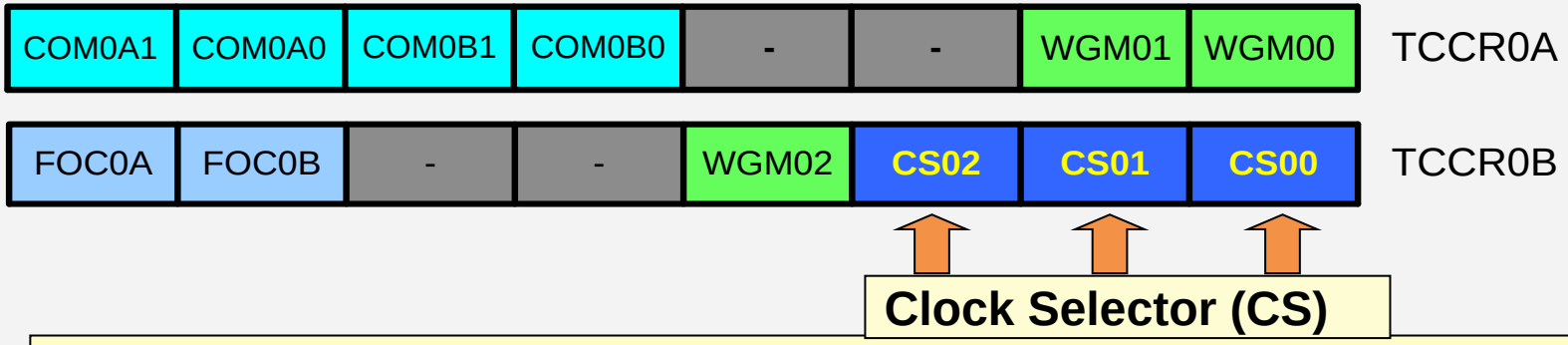
Timer 0 (8-bit timer)



TIMSK0 – Timer/Counter Interrupt Mask Register

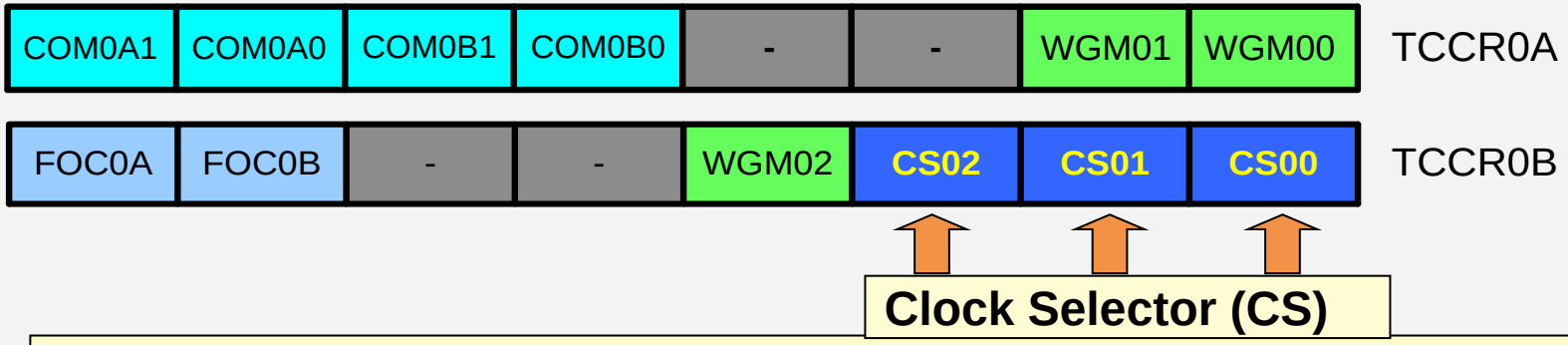
Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

Timer 0 (8-bit timer)



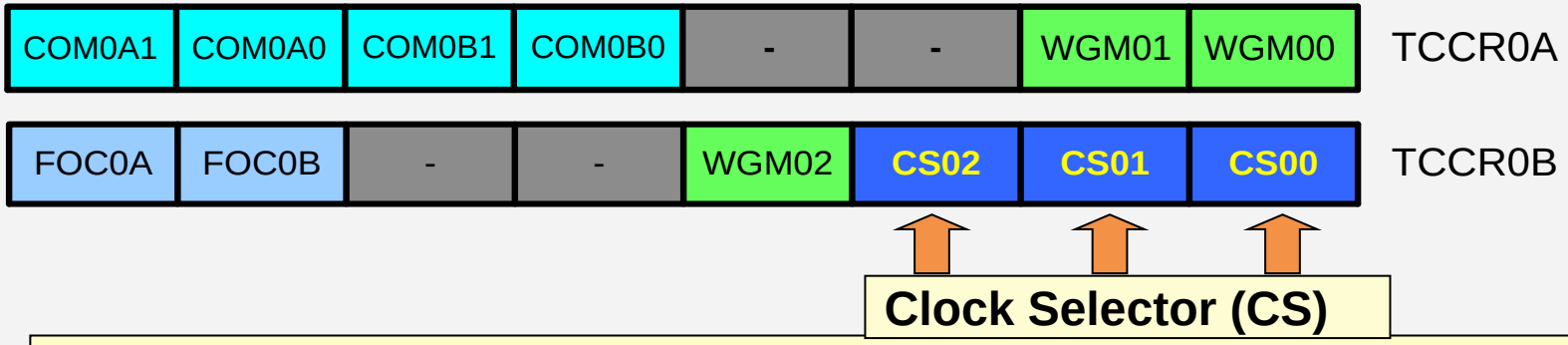
CS02	CS01	CS00	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Timer 0 (8-bit timer)



CS02	CS01	CS00	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

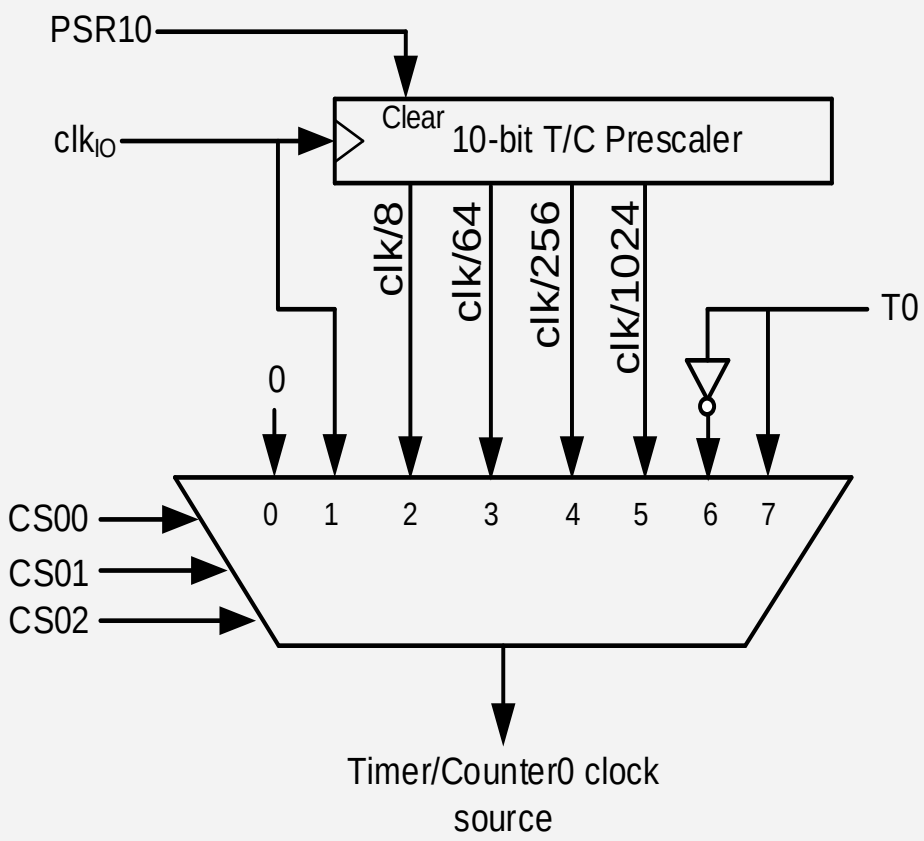
Timer 0 (8-bit timer)



CS02	CS01	CS00	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Timer 0 (8-bit timer)

COM0A1	COM0							TCCR0A
FOC0A	FOC							TCCR0B



CS02	CS01
0	0
0	0
0	1
0	1
1	0
1	0
1	1
1	1

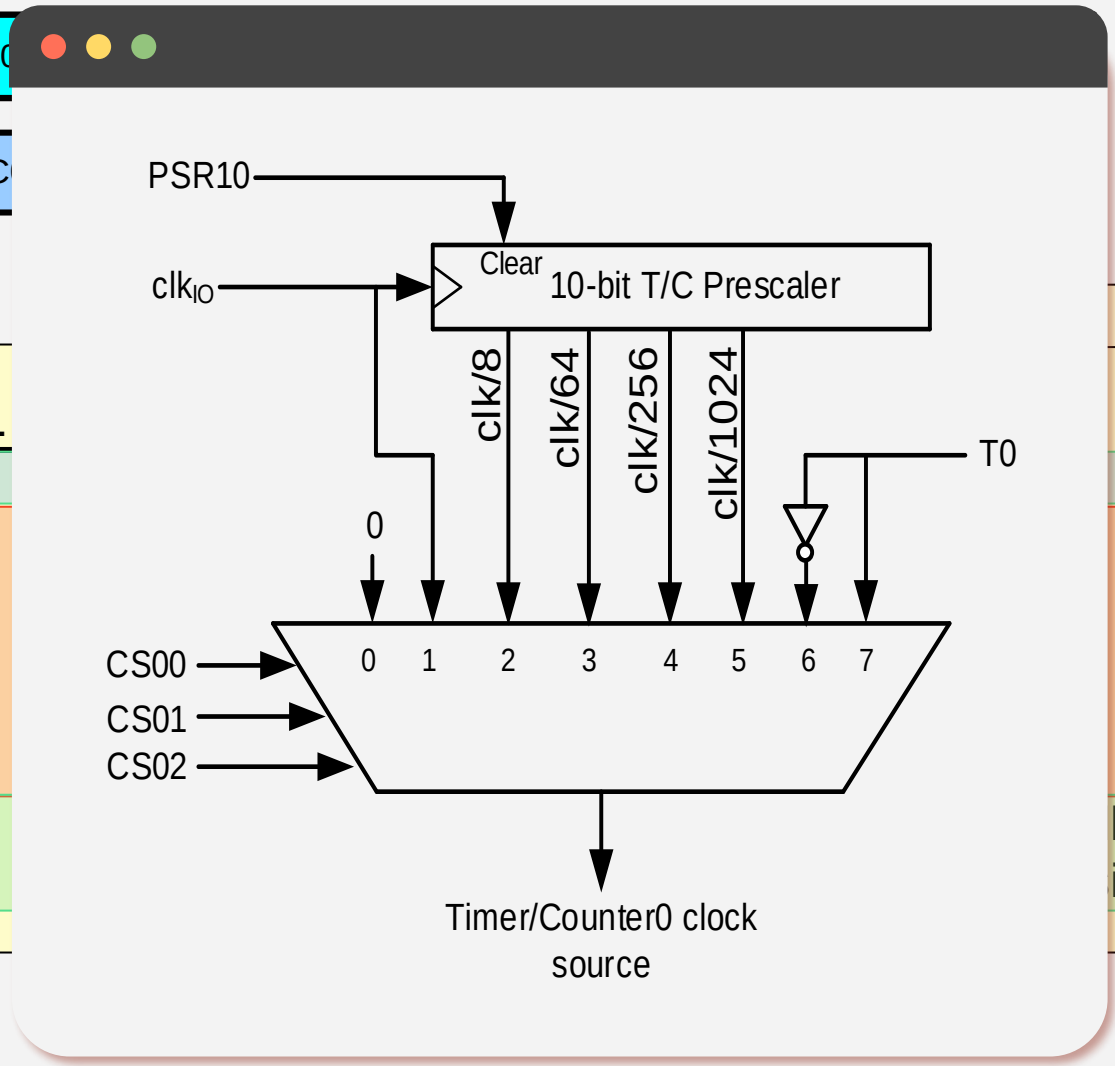
ling edge
ing edge

Timer 0 (8-bit timer)

COM0A1	COM0
FOC0A	FOC

TCCR0A

TCCR0B



CS02	CS01
0	0
0	0
0	1
0	1
1	0
1	0
1	1
1	1

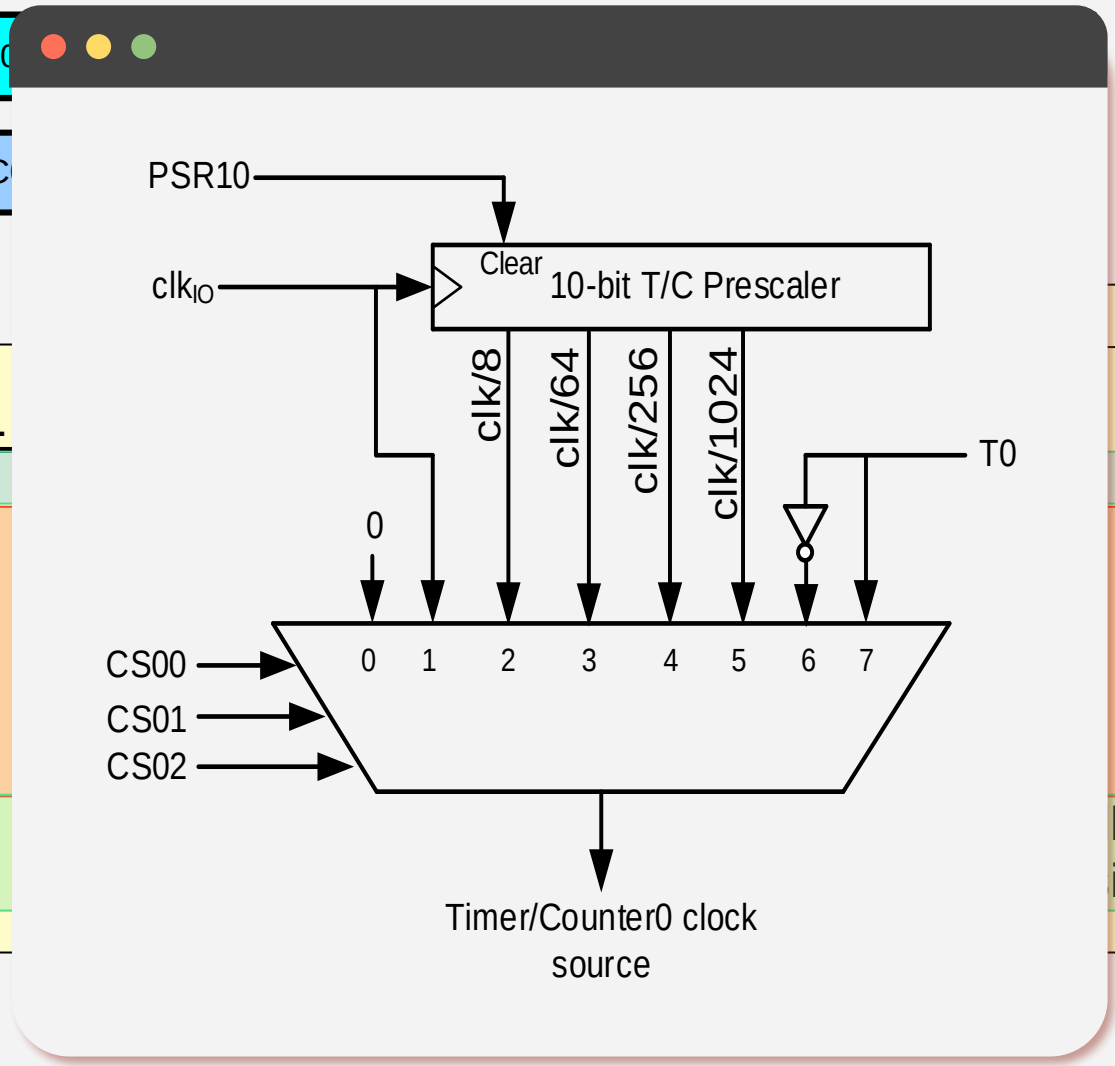
ling edge
ing edge

Timer 0 (8-bit timer)

COM0A1	COM0
FOC0A	FOC

TCCR0A

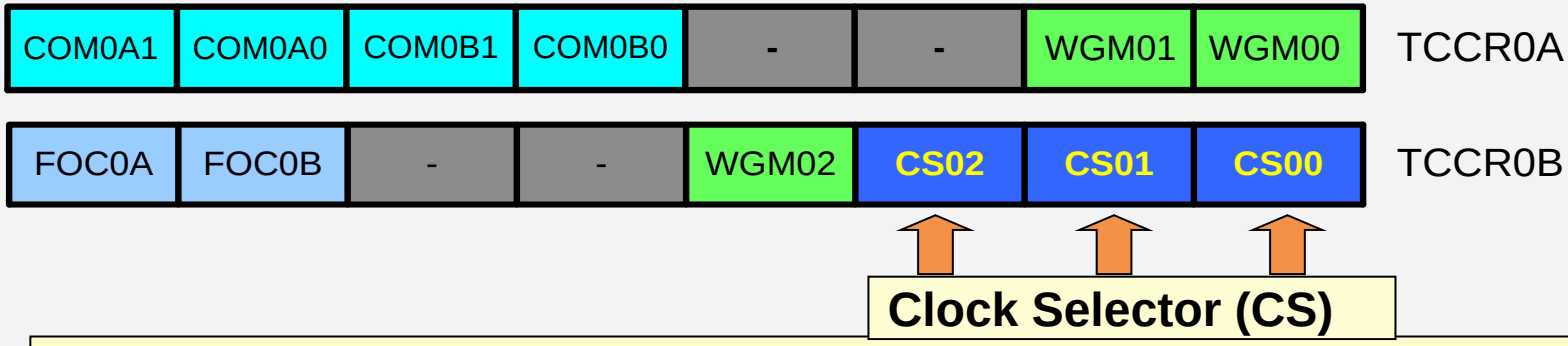
TCCR0B



CS02	CS01
0	0
0	0
0	1
0	1
1	0
1	0
1	1
1	1

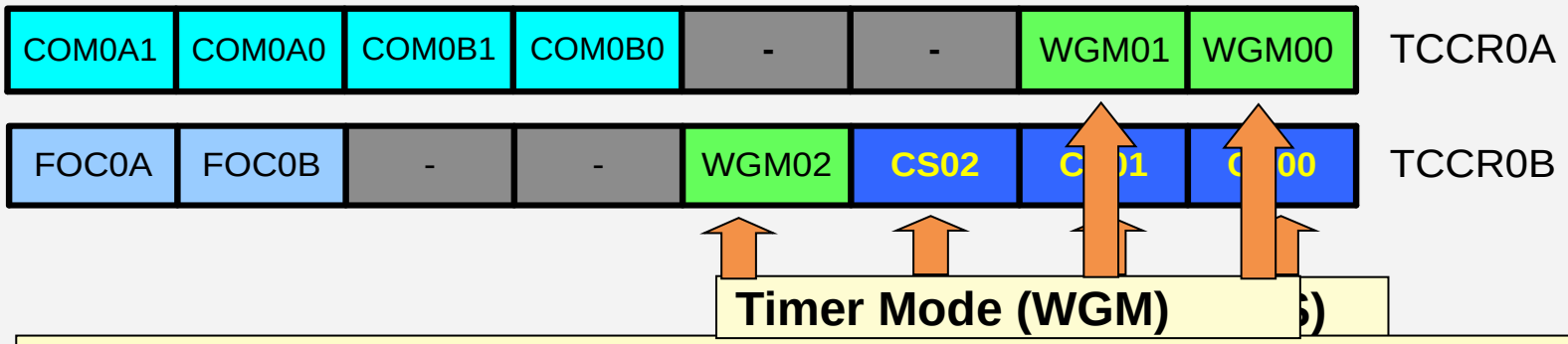
ling edge
ing edge

Timer 0 (8-bit timer)



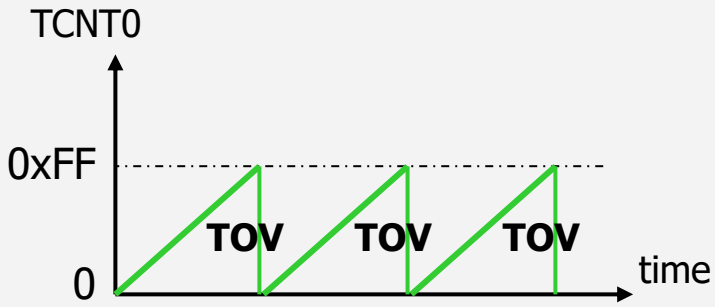
CS02	CS01	CS00	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

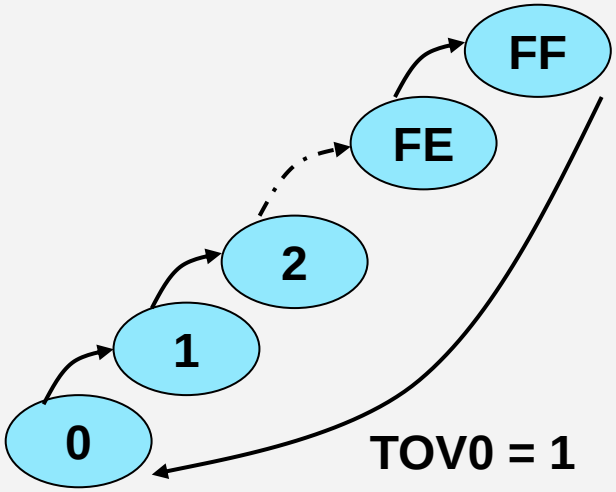
Timer 0 (8-bit timer)



WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC (Clear Timer on Compare Match)
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM

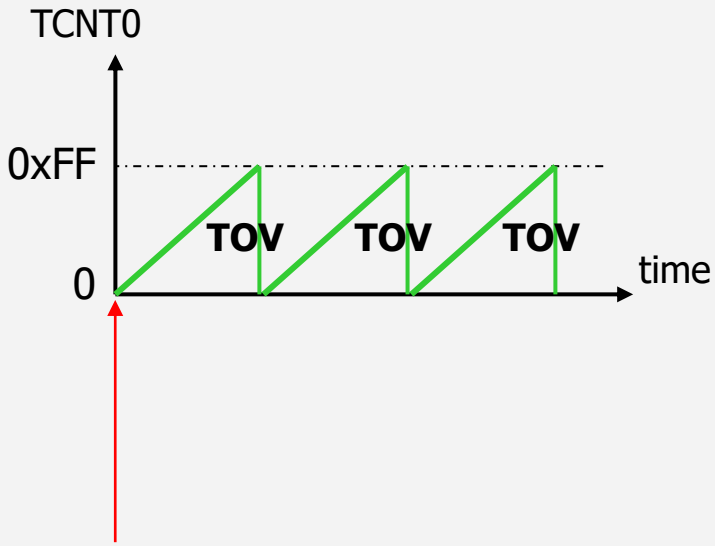
Normal mode

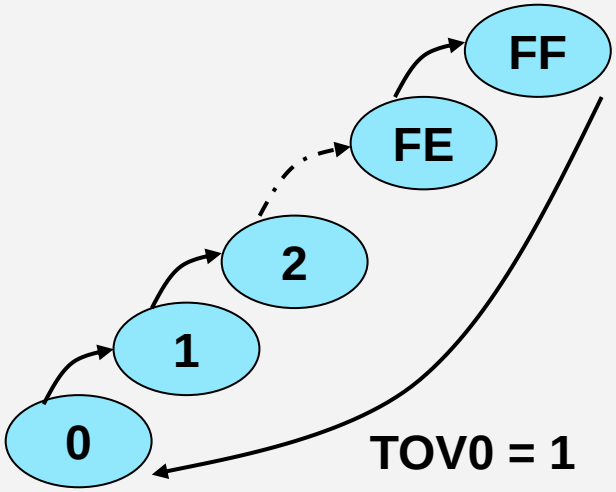




TOV0: 0

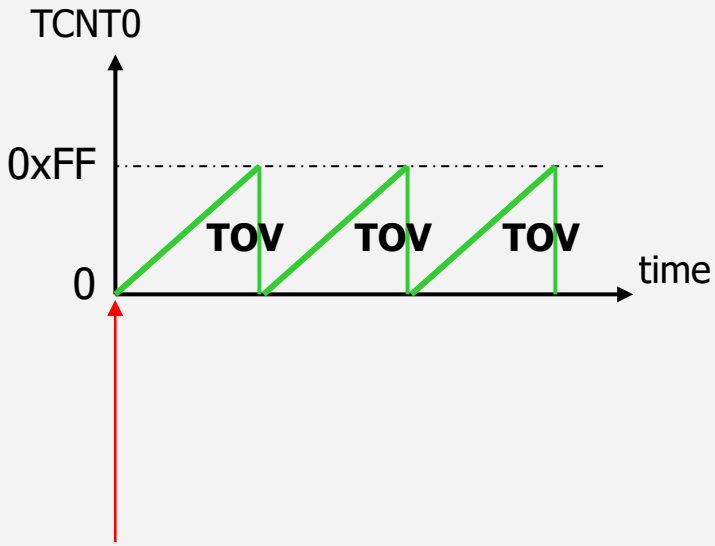
Normal mode

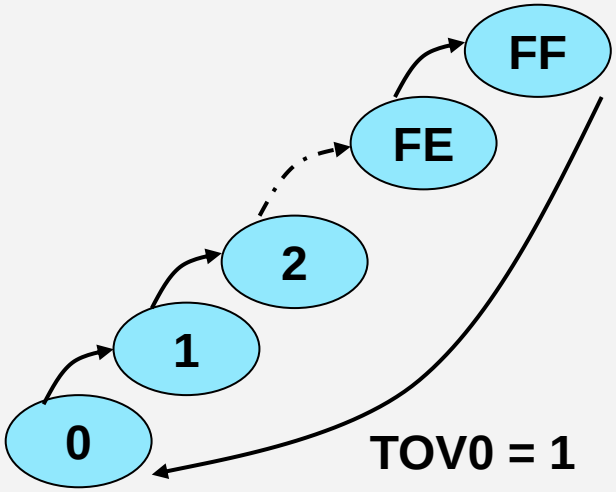




TOV0: **0**

Normal mode

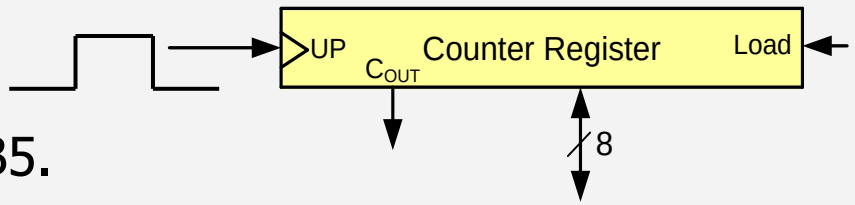




TOV0: **1**

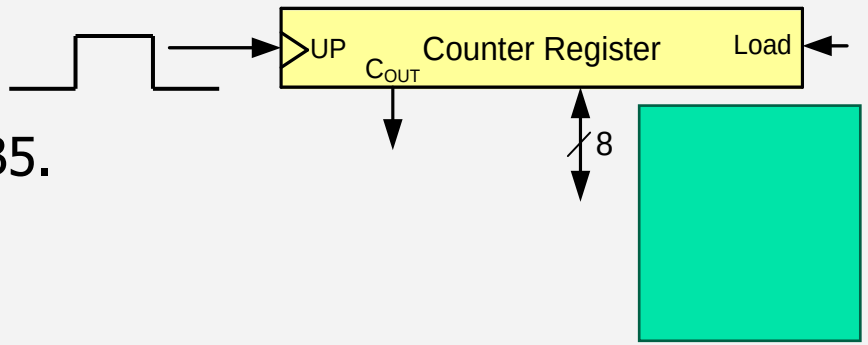
Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



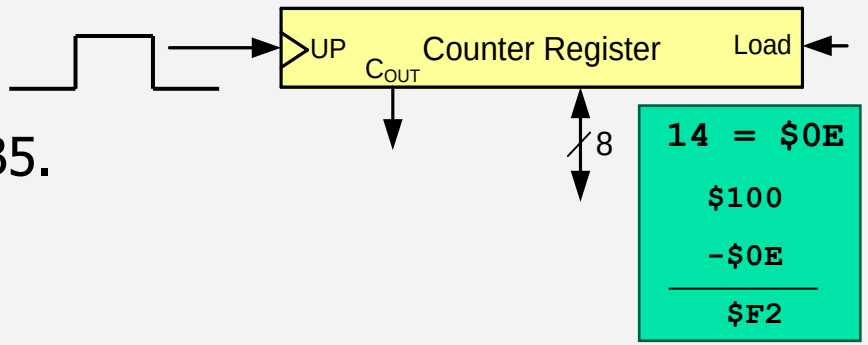
Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



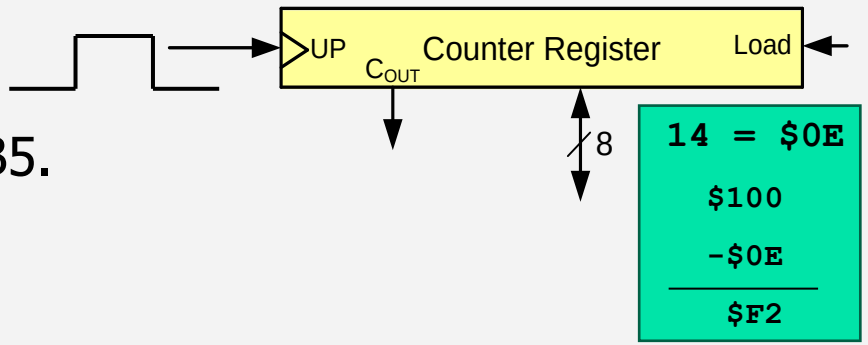
Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



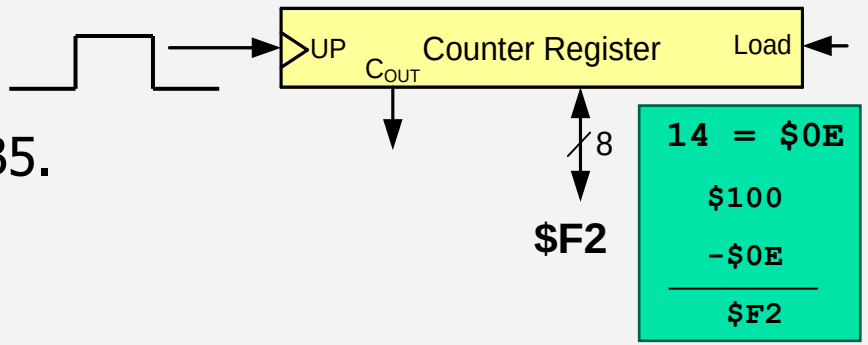
Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



Normal mode

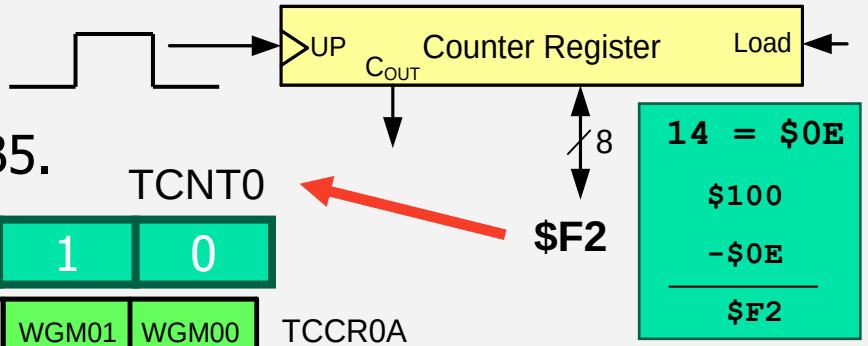
Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



0 0 1

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



TCNT0

1	1	1	1	0	0	1	0

COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

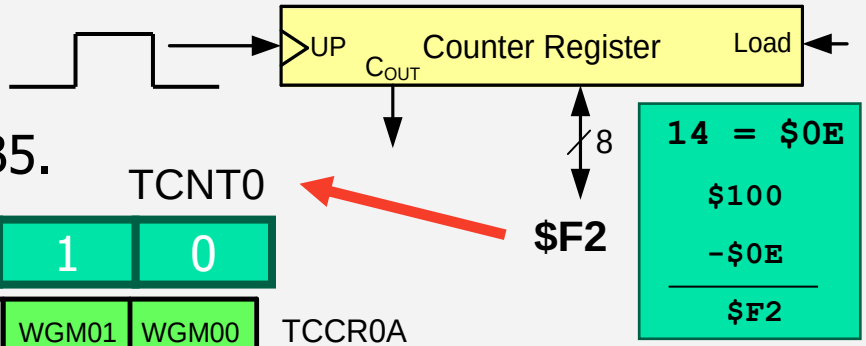
TCCR0A

FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00
					0	0	1

TCCR0B

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



TCNT0

1	1	1	1	0	0	1	0

COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

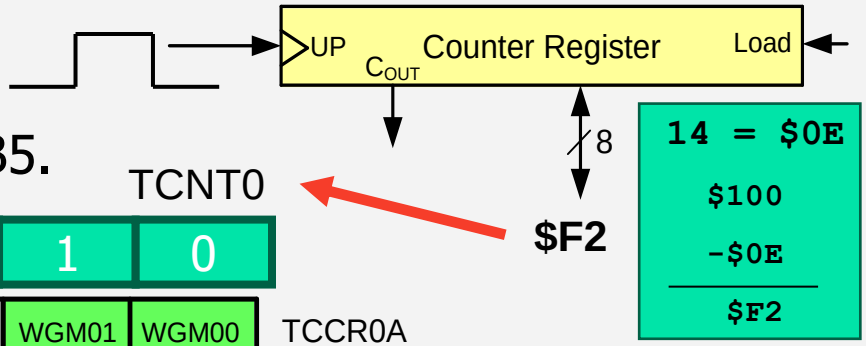
TCCR0A

FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00
					0	0	1

TCCR0B

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



TCNT0

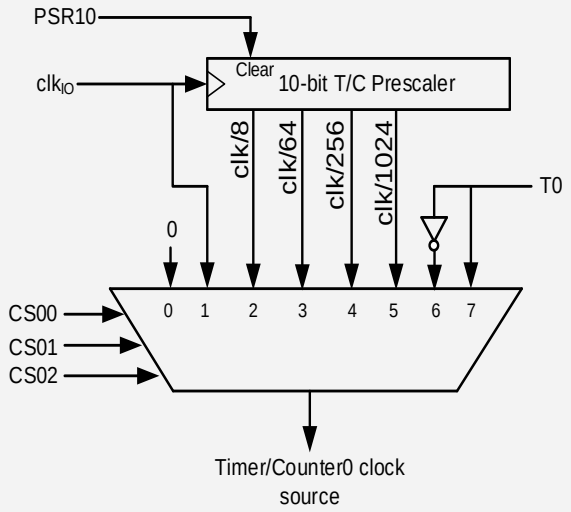
1	1	1	1	0	0	1	0

COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

TCCR0A

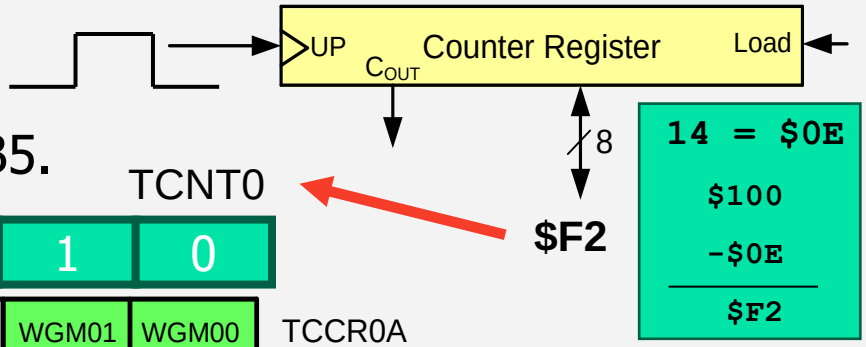
FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00
					0	0	1

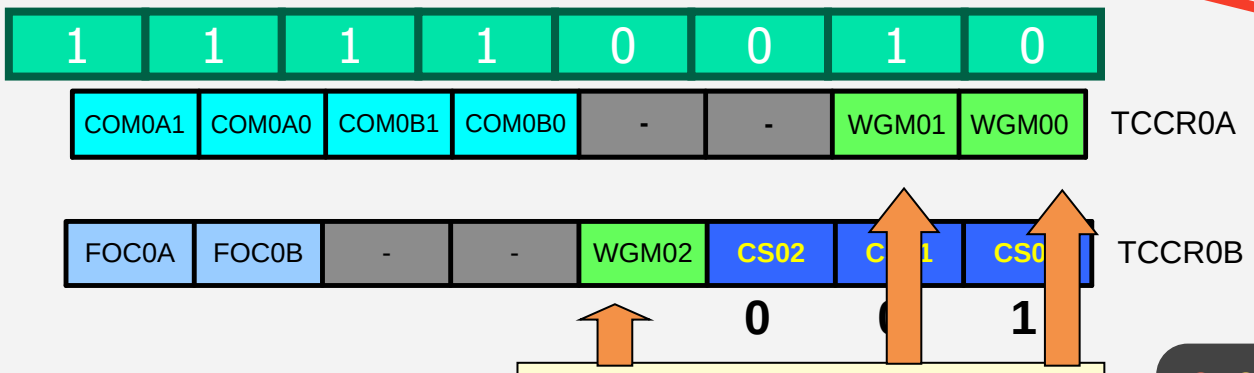
TCCR0B



Normal mode

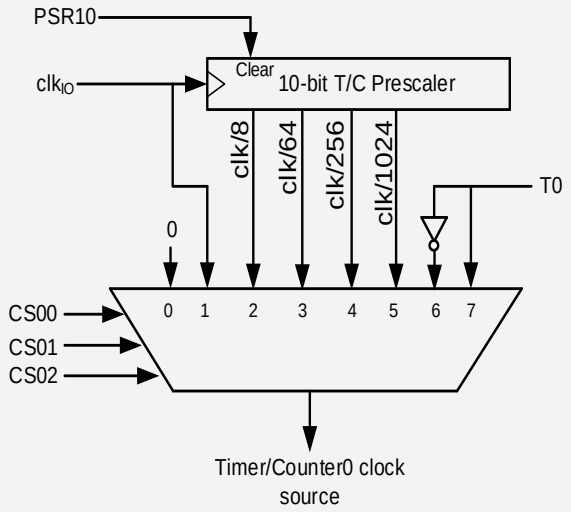
Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.





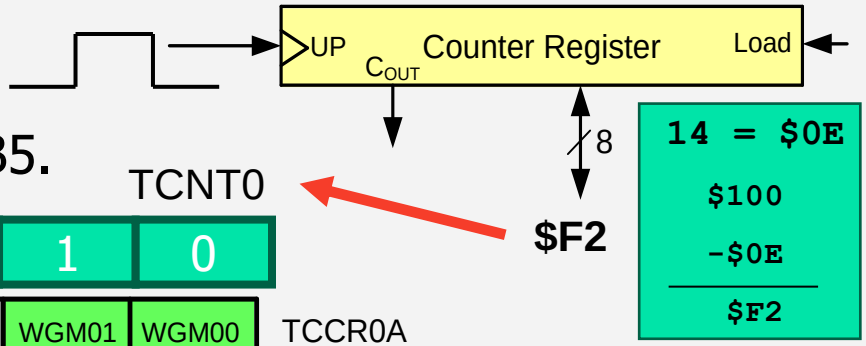
Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



TCNT0

1	1	1	1	0	0	1	0

COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

TCCR0A

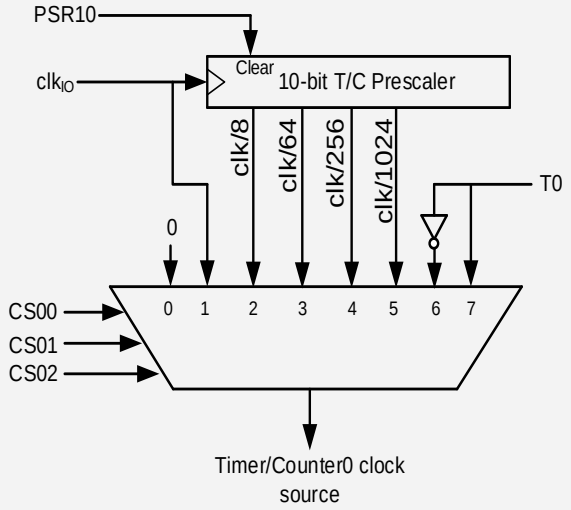
FOC0A	FOC0B	-	-	WGM02	CS02	C 1	CS0

TCCR0B

0 0 0 1

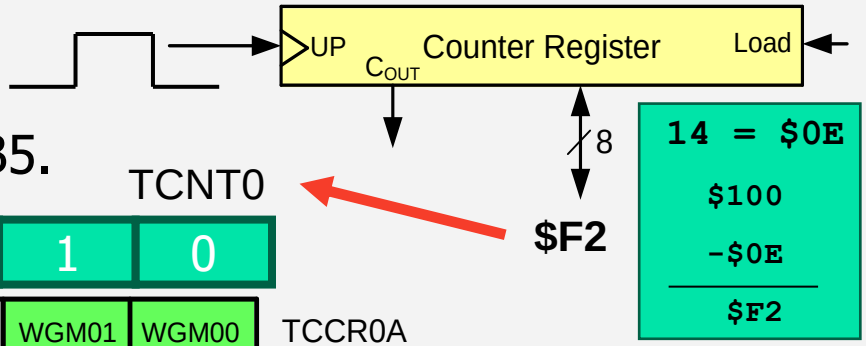
Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



TCNT0

1	1	1	1	0	0	1	0

COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

TCCR0A

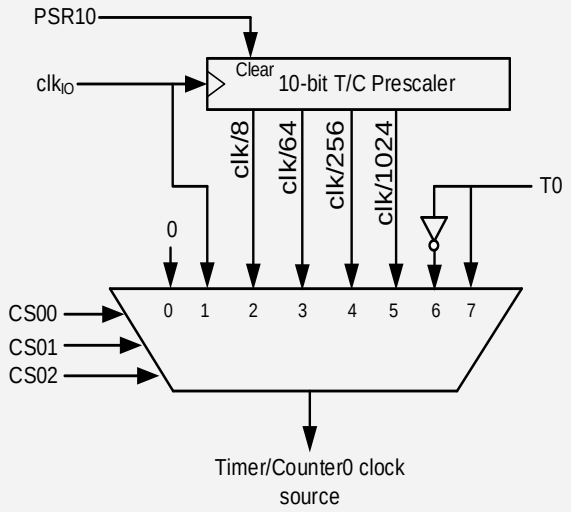
FOC0A	FOC0B	-	-	WGM02	CS02	C 1	CS0

TCCR0B

0 0 0 1

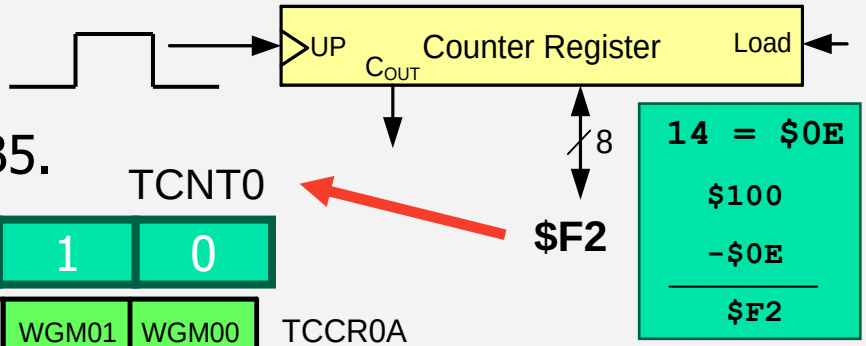
Timer Mode (WGM)

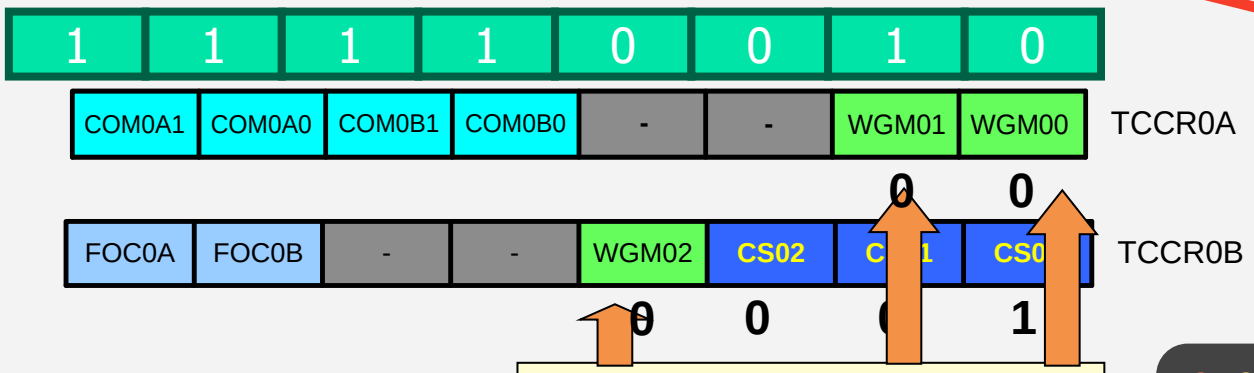
WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Normal mode

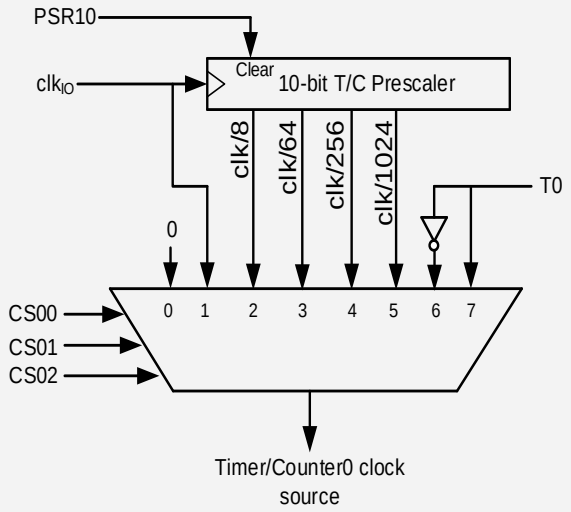
Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.





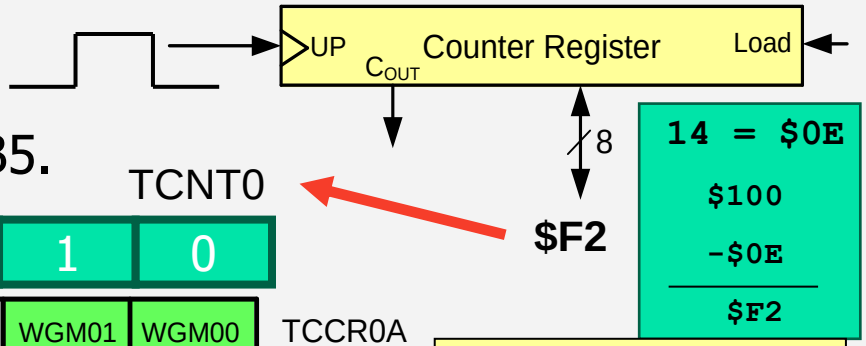
Timer Mode (WGM)

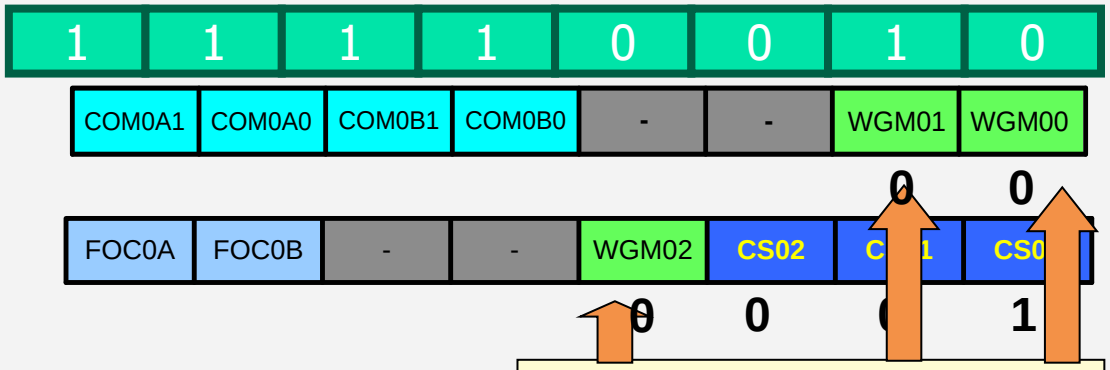
WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.

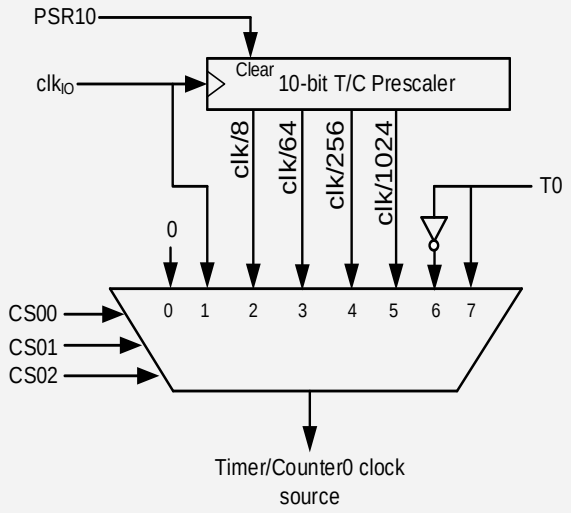





```
TCCR0A = 0x00;
TCCR0B = 0x03;
```

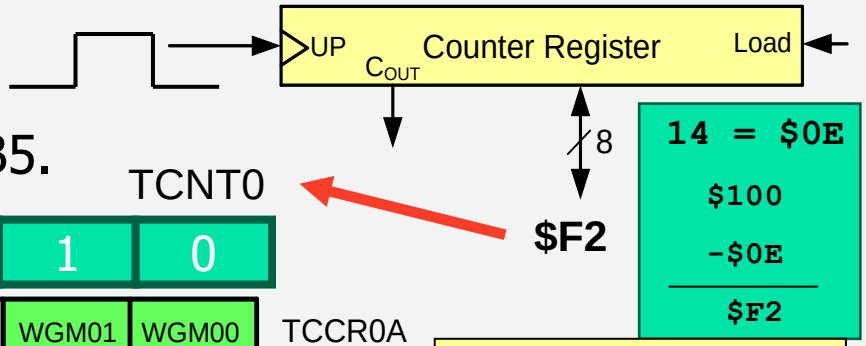
Timer Mode (WGM)

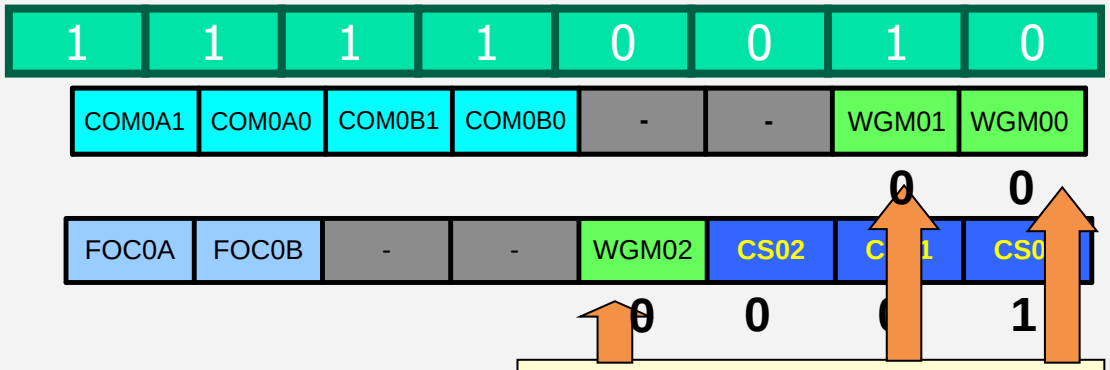
WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.

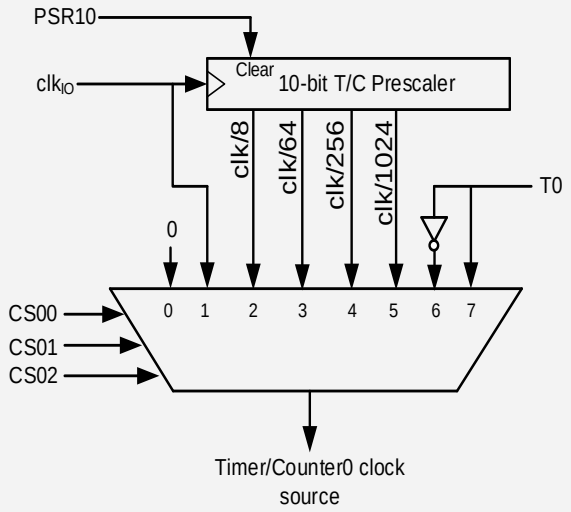





```
TCCR0A = 0x00;
TCCR0B = 0x03;
```

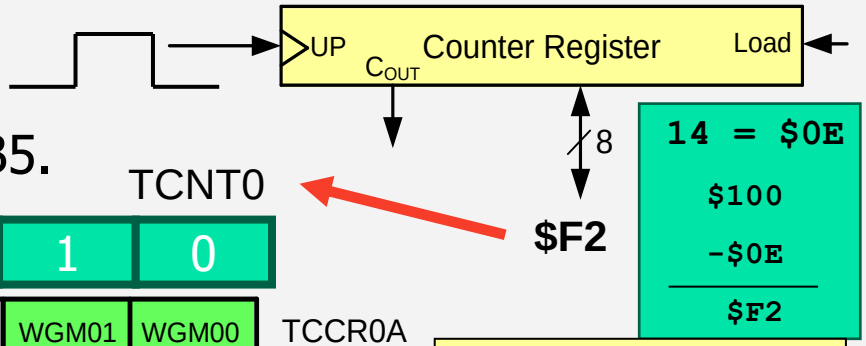
Timer Mode (WGM)

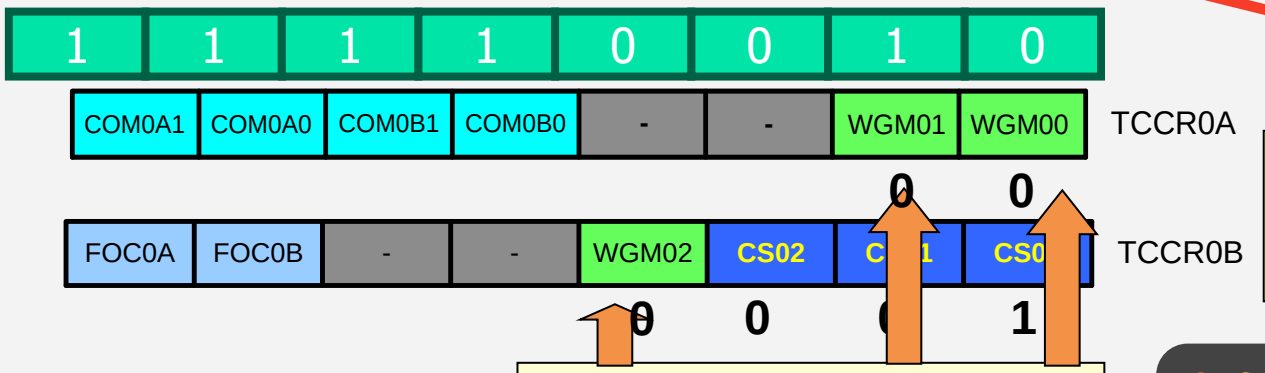
WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.

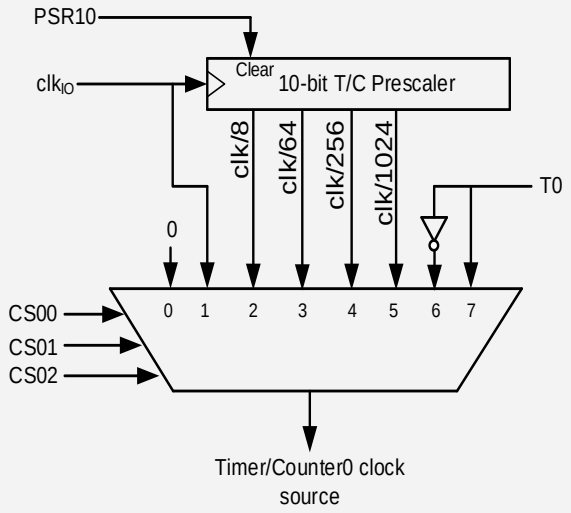





```
TCCR0A = 0x00;
TCCR0B = 0x03;
```

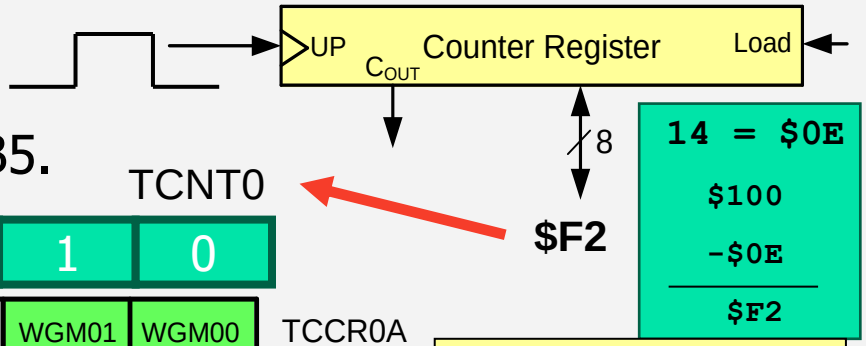
Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



TCNT0

1	1	1	1	0	0	1	0

TCCR0A

COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

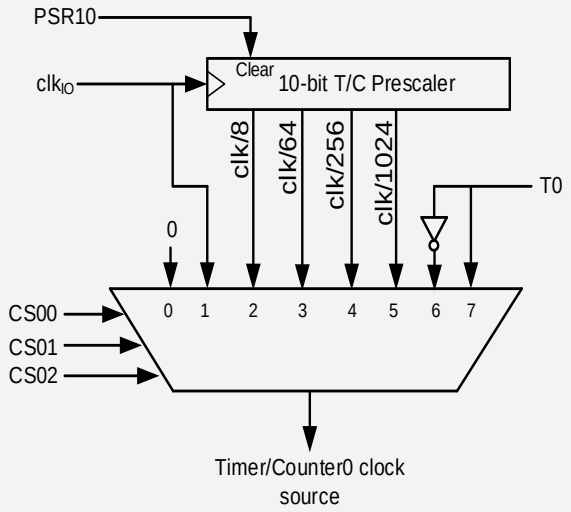
TCCR0B

FOC0A	FOC0B	-	-	WGM02	CS02	C 1	CS0
				0	0	0	1

```
TCCR0A = 0x00;
TCCR0B = 0x03;
```

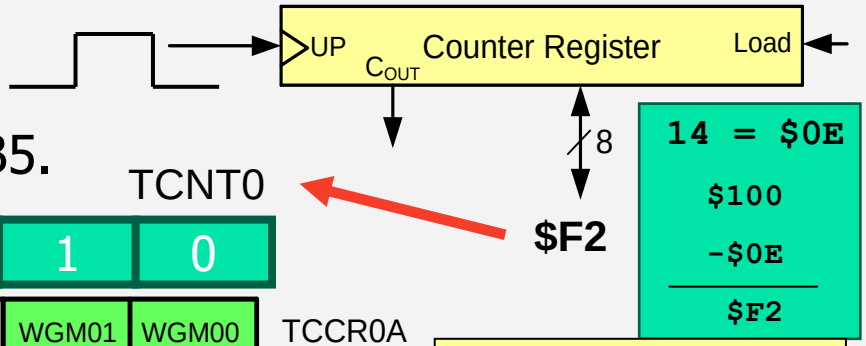
Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



TCNT0

1	1	1	1	0	0	1	0

COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

TCCR0A

FOC0A	FOC0B	-	-	WGM02	CS02	C 1	CS0

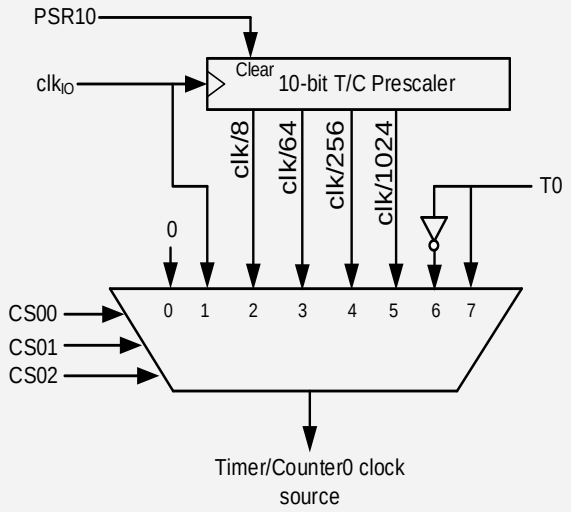
TCCR0B

0 0 0 1

```
TCCR0A = 0x00;
TCCR0B = 0x03;
```

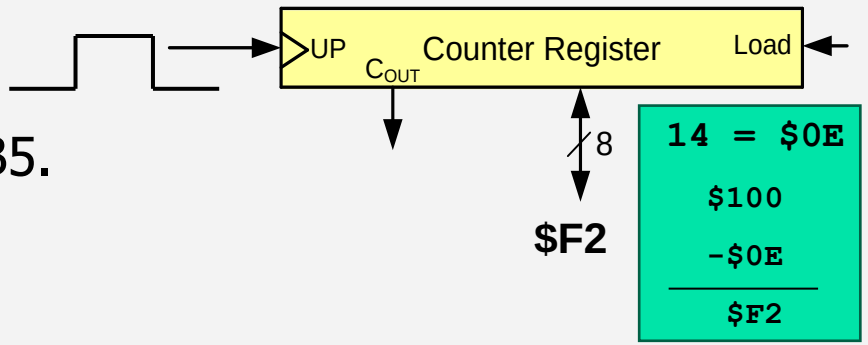
Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM



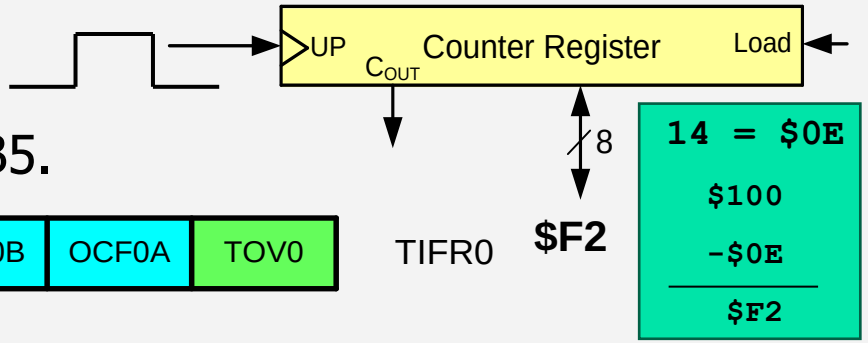
Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.

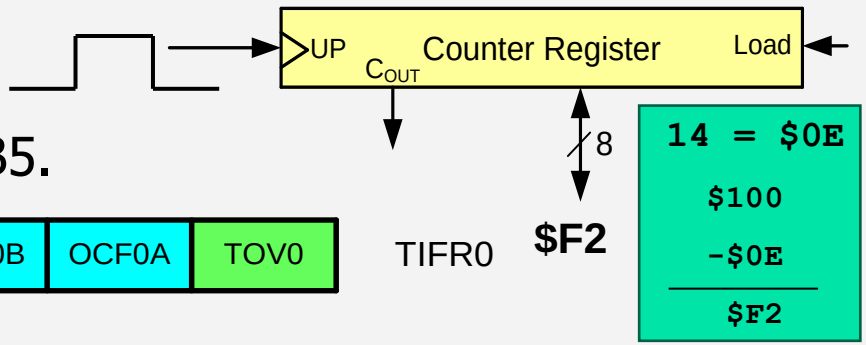


-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.

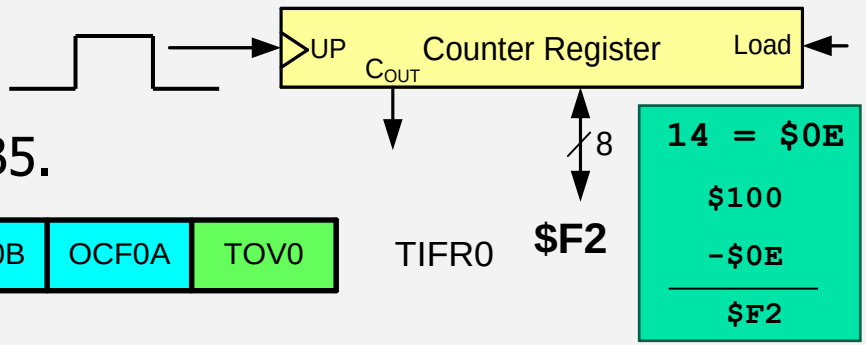


-	-	-	-	-	OCF0B	OCF0A	TOV0

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS   TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.

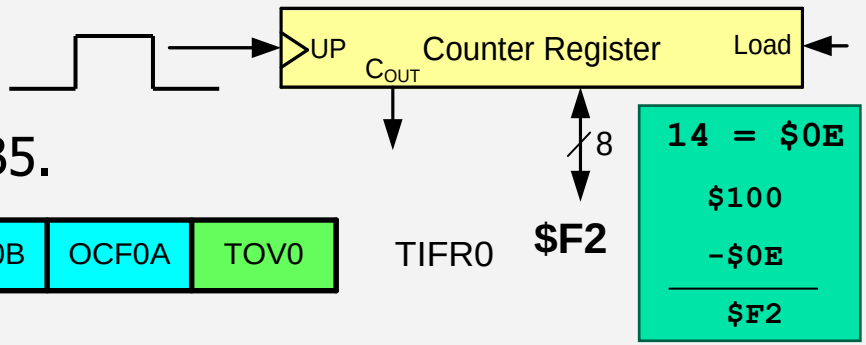


-	-	-	-	-	OCF0B	OCF0A	TOV0

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.

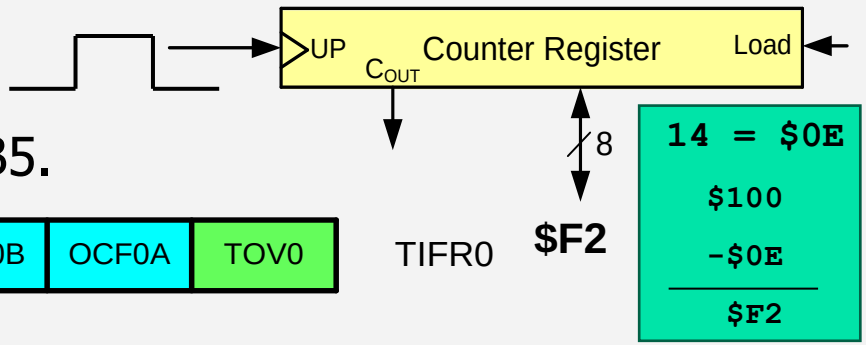


-	-	-	-	-	OCF0B	OCF0A	TOV0

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



-	-	-	-	-	OCF0B	OCF0A	TOV0

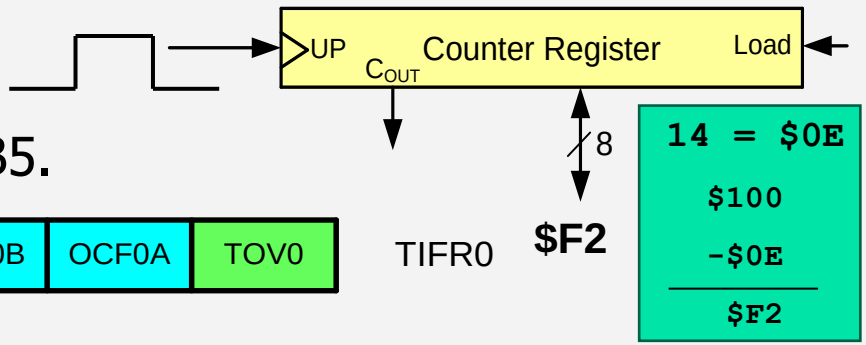
Assembly

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

```
DDRB = 1<<5;
PORTB &= ~(1<<5);  //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&(1<<TOV0))==0)
  {}
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



-	-	-	-	-	OCF0B	OCF0A	TOV0

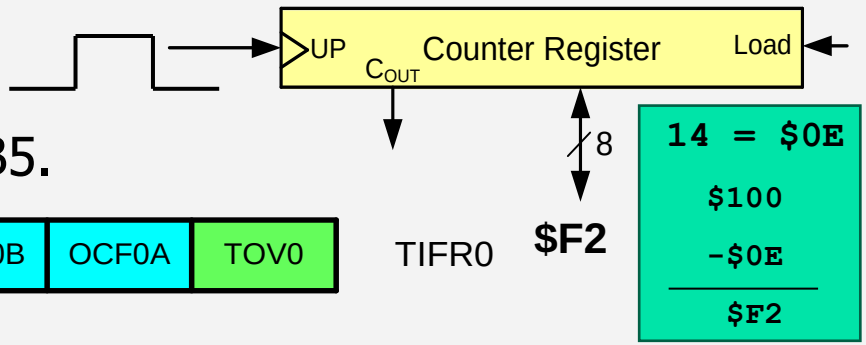
Assembly

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

```
DDRB = 1<<5;
PORTB &= ~(1<<5);   //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&(1<<TOV0))==0)
  {}
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



Assembly

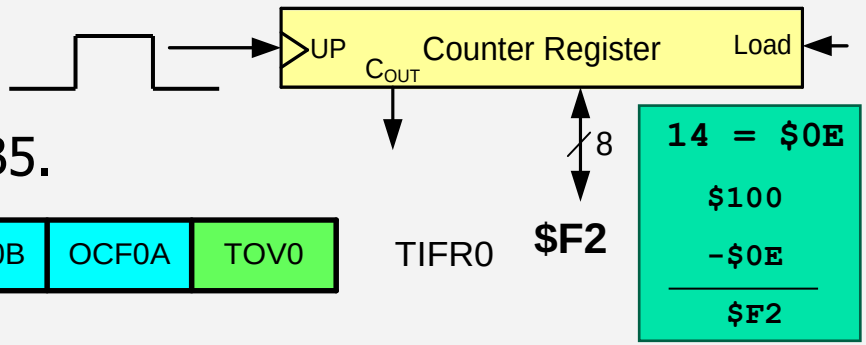
```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS  TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1<<5;
PORTB &= ~(1<<5);  //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&(1<<TOV0))==0)
  {}
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



-	-	-	-	-	OCF0B	OCF0A	TOV0

Assembly

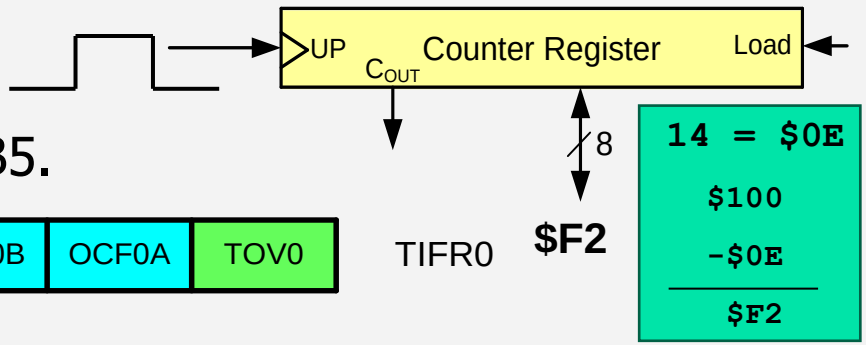
```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1<<5;
PORTB &= ~(1<<5);   //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&(1<<TOV0))==0)
  {}
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



-	-	-	-	-	OCF0B	OCF0A	TOV0

Assembly

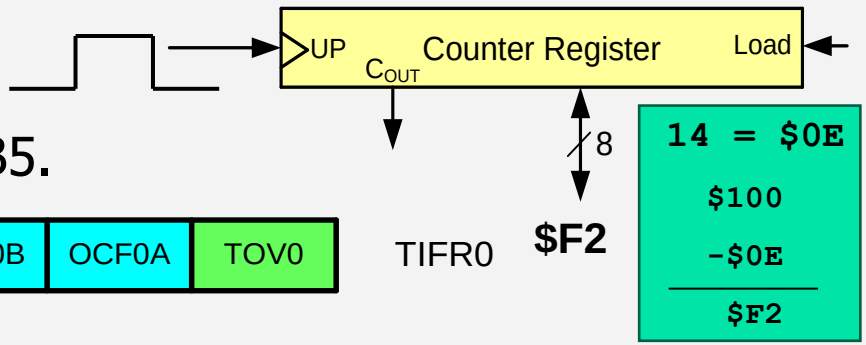
```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1<<5;
PORTB &= ~(1<<5);   //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&(1<<TOV0))==0)
  {}
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



-	-	-	-	-	OCF0B	OCF0A	TOV0

Assembly

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

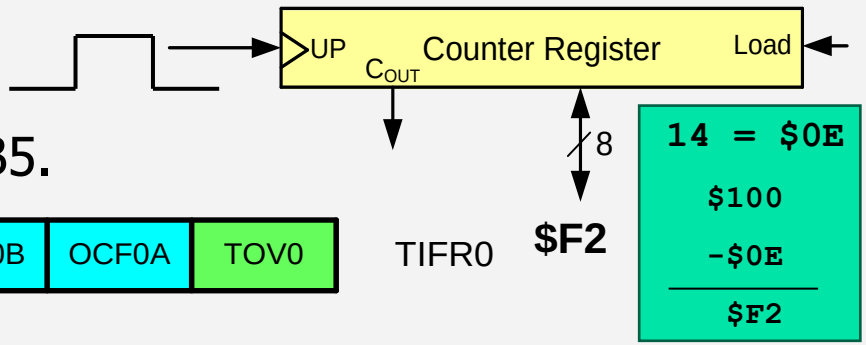
C

```
DDRB = 1<<5;
PORTB &= ~(1<<5);  //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&(1<<TOV0))==0)
  {}
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Note that we have to write 1 to the TOV0 bit to clear the flag.

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



-	-	-	-	-	OCF0B	OCF0A	TOV0

Assembly

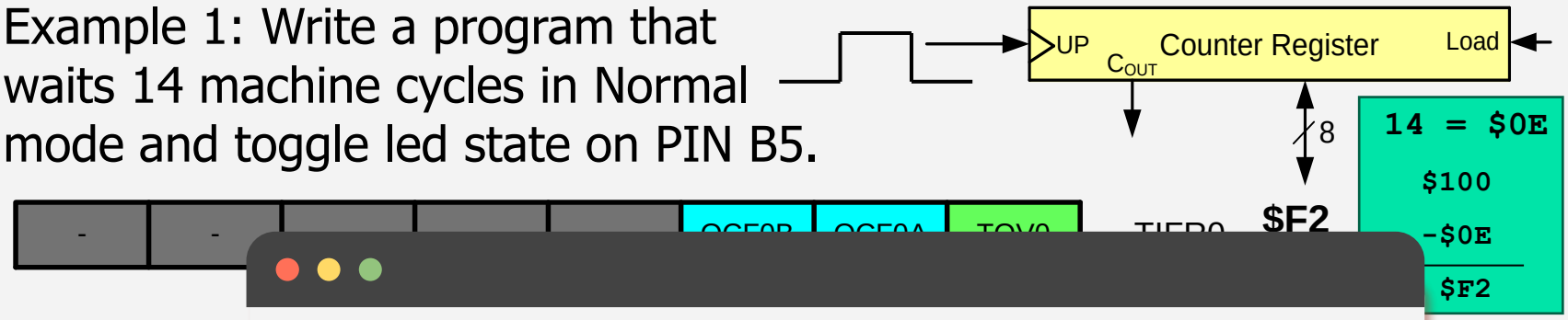
```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1<<5;
PORTB &= ~(1<<5);   //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&(1<<TOV0))==0)
  {}
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



How to calculate the delay generated by the timer?

Answer:

1) Calculate how much a machine clock lasts.
   T= 1/f
2) Calculate how many machine clocks it waits.
3) Delay = T * number of machine cycles

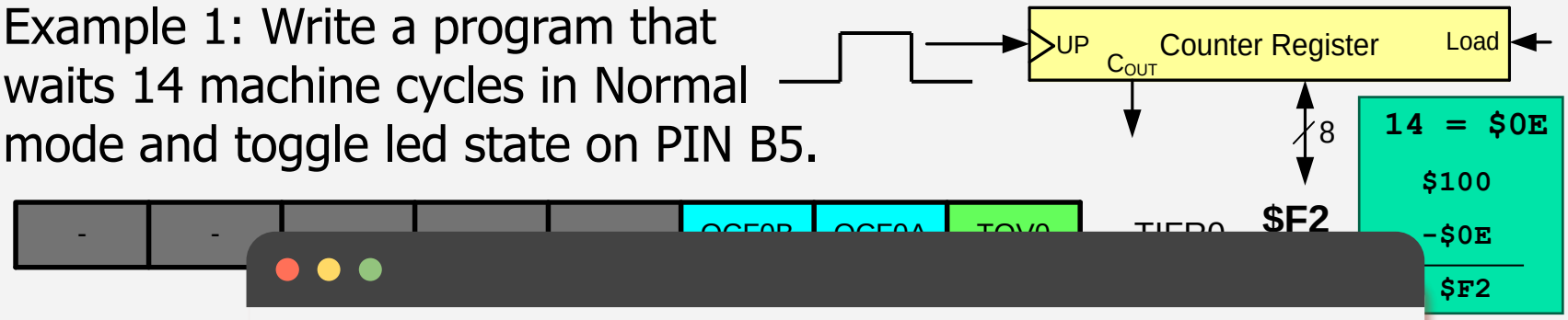
Assembl

```
          LDI
          SBI
          LDI
          OUT
BEGIN:    LDI
          OUT
          LDI
          OUT
          LDI
          OUT
AGAIN:    SBIS
          RJMP      AGAIN
          LDI       R20,0x0
          OUT       TCCR0B,R20          ;stop Timer0
          LDI       R20,(1<<TOV0)       ;R20 = 0x01
          OUT       TIFR0,R20 ;clear TOV0 flag

          EOR       R17,R16             ;toggle D5 of R17
          OUT       PORTB,R17 ;toggle PB5
          RJMP      BEGIN
```

```
PB5=0
V0))==0)
  TCCR0B = 0;
  TIFR0 = (1<<TOV0);
  PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



How to calculate the delay generated by the timer?

Answer:

1) Calculate how much a machine clock lasts.
   T= 1/f
2) Calculate how many machine clocks it waits.
3) Delay = T * number of machine cycles

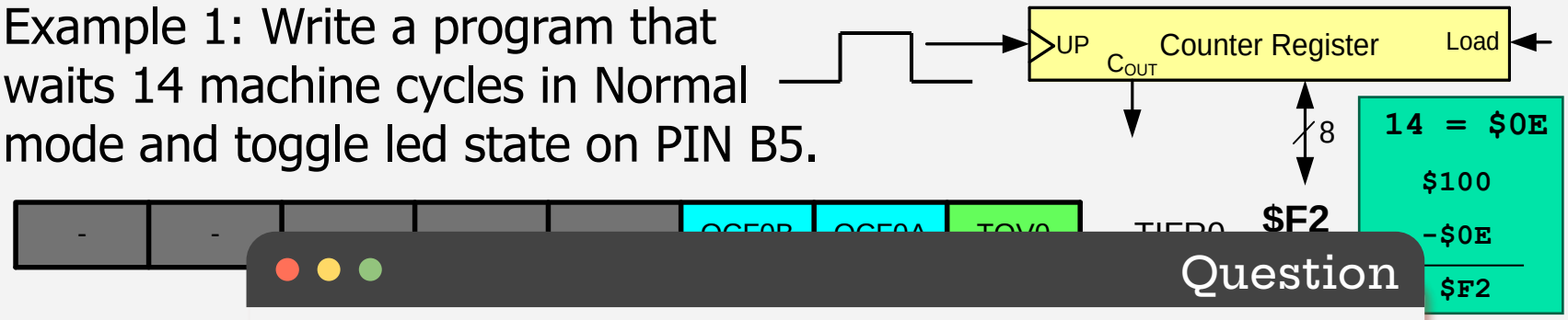
Assembl

```
        LDI
        SBI
        LDI
        OUT
BEGIN:  LDI
        OUT
        LDI
        OUT
        LDI
        OUT
AGAIN:  SBIS
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

```
PB5=0
V0))==0)
    TCCR0B = 0;
    TIFR0 = (1<<TOV0);
    PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



Question

How to calculate the delay generated by the timer?

Answer:

1) Calculate how much a machine clock lasts.
   T= 1/f
2) Calculate how many machine clocks it waits.
3) Delay = T * number of machine cycles

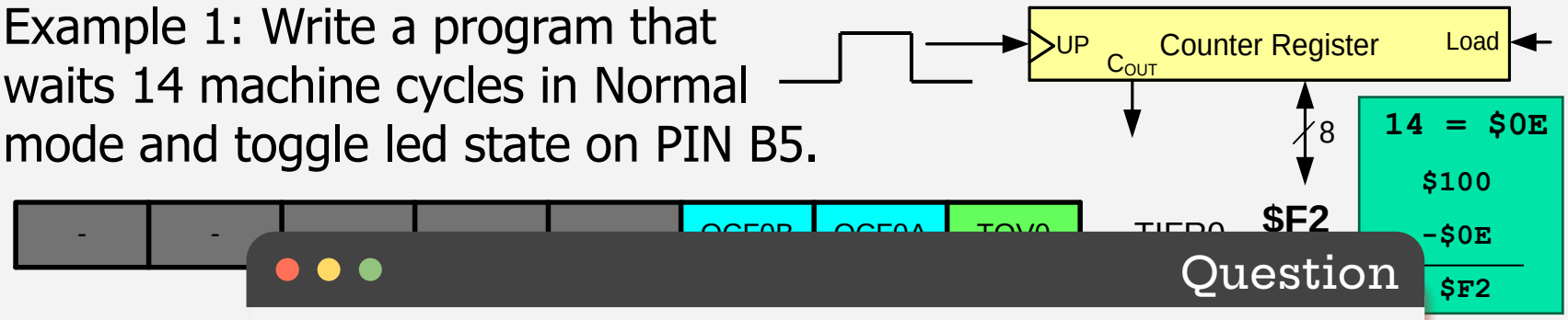
Assembl

```
        LDI
        SBI
        LDI
        OUT
BEGIN:  LDI
        OUT
        LDI
        OUT
        LDI
        OUT
AGAIN:  SBIS
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

```
PB5=0
V0))==0)
    TCCR0B = 0;
    TIFR0 = (1<<TOV0);
    PORTB ^= (1<<5);
}
```

Normal mode

Example 1: Write a program that waits 14 machine cycles in Normal mode and toggle led state on PIN B5.



Question

How to calculate the delay generated by the timer?

Answer:

1) Calculate how much a machine clock lasts.
   T= 1/f
2) Calculate how many machine clocks it waits.
3) Delay = T * number of machine cycles

```
Assembl
        LDI
        SBI
        LDI
        OUT
BEGIN:  LDI
        OUT
        LDI
        OUT
        LDI
        OUT
AGAIN:  SBIS
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16             ;toggle D5 of R17
        OUT     PORTB,R17 ;toggle PB5
        RJMP    BEGIN
```

```
PB5=0
V0))==0)
    TCCR0B = 0;
    TIFR0 = (1<<TOV0);
    PORTB ^= (1<<5);
}
```

Solution

In example 1 calculate the delay. XTAL = 10 MHz.

Solution 1 (inaccurate)

1) Calculating T:

T = 1/f = 1/10M = 0.1µs

2) Calculating num of machine cycles:

```
 $100
-$F2
 ----
  $0E = 14
```

3) Calculating delay

14 * 0.1µs = 1.4 µs

Assembly

```
          LDI     R16,(1<<5) ;R16=0x20
          SBI     DDRB,5     ;PB5 as an output
          LDI     R17,0
          OUT     PORTB,R17
BEGIN:    LDI     R20,0xF2
          OUT     TCNT0,R20 ;load timer0
          LDI     R20,0x0
          OUT     TCCR0A,R20
          LDI     R20,0x01
          OUT     TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:    SBIS   TIFR0,TOV0 ;if TOV0 is set skip next
          RJMP    AGAIN
          LDI     R20,0x0
          OUT     TCCR0B,R20          ;stop Timer0
          LDI     R20,(1<<TOV0)       ;R20 = 0x01
          OUT     TIFR0,R20 ;clear TOV0 flag

          EOR     R17,R16             ;toggle D5 of R17
          OUT     PORTB,R17 ;toggle PB5
          RJMP    BEGIN
```

Accurate calculating

Other than timer, executing the instructions consumes time; so if we want to calculate the accurate delay a program causes we should add the delay caused by instructions to the delay caused by the timer

```
        LDI     R16,0x20
        SBI     DDRB,5
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0xF2        1
        OUT     TCNT0,R20       1
        LDI     R20,0x00        1
        OUT     TCCR0A,R20      1
        LDI     R20,0x01        1
        OUT     TCCR0B,R20      1
AGAIN:  SBIS    TIFR0,TOV0      1 / 2
        RJMP    AGAIN           2
        LDI     R20,0x0         1
        OUT     TCCR0B,R20      1
        LDI     R20,0x01        1
        OUT     TIFR0,R20       1
        EOR     R17,R16         1
        OUT     PORTB,R17       1
        RJMP    BEGIN           2
                                18
```

Delay caused by timer = 14 * 0.1µs = 1.4 µs **Delay caused by instructions = 18 * 0.1µs = 1.8**

Total delay = 3.2 µs ➔ wave period = 2*3.2 µs = 6.4 µs ➔ wave frequency = 156.25 KHz

Finding values to be loaded into the timer

1. Calculate the period of clock source.
   - Period = 1 / Frequency
     - E.g. For XTAL = 16 MHz ➔ T = 1/16MHz
2. Divide the desired time delay by period of clock.
3. Perform 256 - n, where n is the decimal value we got in Step 2.
4. Set TCNT0 = 256 - n

Example 2

Question: Assuming that XTAL = 10 MHz, write a program to generate a square wave with a period of µ10 s on pin PORTB.3.

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. 256 - 50 = 206.

Assembly

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,206
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, int. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16     ;toggle D5 of R17
        OUT     PORTB,R17   ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1<<3;
PORTB &= ~ (1<<3);
while (1)
{
  TCNT0 = 206;
  TCCR0A = 0x00;
  TCCR0B = 0x01;
  while((TIFR0&0x01) == 0)
  {}
  TCCR0B = 0;
  TIFR0 = 1<<TOV0;
  PORTB = PORTB ^ (1<<3);
}
```

Example 3

Question: Modify TCNT0 in Example 2 to get the largest time delay possible with no prescaler. Find the delay in µs. In your calculation, do not include the overhead due to instructions.

- To get the largest delay we make TCNT0 zero. This will count up from 00 to 0xFF and then roll over to zero.

Assembly

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R17,0
        OUT     PORTB,R17
BEGIN:  LDI     R20,0
        OUT     TCNT0,R20 ;load timer0
        LDI     R20,0x0
        OUT     TCCR0A,R20
        LDI     R20,0x01
        OUT     TCCR0B,R20 ;Normal mode, int. clk
AGAIN:  SBIS    TIFR0,TOV0 ;if TOV0 set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,(1<<TOV0)       ;R20 = 0x01
        OUT     TIFR0,R20 ;clear TOV0 flag

        EOR     R17,R16     ;toggle D5 of R17
        OUT     PORTB,R17   ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1 << 3;
PORTB &= ~(1<<3);
while (1)
{
  TCNT0 = 0x00;
  TCCR0A = 0x00;
  TCCR0B = 0x01;

while((TIFR0&(1<<TOV0))==0)
{}
  TCCR0B = 0;
  TIFR0 = 0x01;
  PORTB = PORTB^(1<<3);
}
```

Example 3

Question: Modify TCNT0 in Example 2 to get the largest time delay possible with no prescaler. Find the delay in µs. In your calculation, do not include the overhead due to instructions.

- To get the largest delay we make TCNT0 zero. This will count up from 00 to 0xFF and then roll over to zero.

Assembly

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R
        OUT     P
BEGIN:  LDI     R
        OUT     T
        LDI     R
        OUT     T
        LDI     R
        OUT     T
AGAIN:  SBIS    T
        RJMP    A
        LDI     R
        OUT     T
        LDI     R
        OUT     T

        EOR     R17,R16     ;toggle D5 of R17
        OUT     PORTB,R17   ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1 << 3;
= ~(1<<3);
1)

 = 0x00;
A = 0x00;
B = 0x01;

TIFR0&(1<<TOV0))==0)

B = 0;
 = 0x01;
 = PORTB^(1<<3);
}
```

Example 3

Question: Modify TCNT0 in Example 2 to get the largest time delay possible with no prescaler. Find the delay in µs. In your calculation, do not include the overhead due to instructions.

- To get the largest delay we make TCNT0 zero. This will count up from 00 to 0xFF and then roll over to zero.

Assembly

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R
        OUT     P
BEGIN:  LDI     R
        OUT     T
        LDI     R
        OUT     T
        LDI     R
        OUT     T
AGAIN:  SBIS    T
        RJMP    A
        LDI     R
        OUT     T
        LDI     R
        OUT     T

        EOR     R17,R16     ;toggle D5 of R17
        OUT     PORTB,R17   ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1 << 3;
= ~(1<<3);
1)

 = 0x00;
A = 0x00;
B = 0x01;

TIFR0&(1<<TOV0))==0)

B = 0;
 = 0x01;
 = PORTB^(1<<3);
}
```

Question

Example 3

Question: Modify TCNT0 in Example 2 to get the largest time delay possible with no prescaler. Find the delay in µs. In your calculation, do not include the overhead due to instructions.

- To get the largest delay we make TCNT0 zero. This will count up from 00 to 0xFF and then roll over to zero.

Assembly

```
        LDI     R16,(1<<5) ;R16=0x20
        SBI     DDRB,5     ;PB5 as an output
        LDI     R
        OUT     P
BEGIN:  LDI     R
        OUT     T
        LDI     R
        OUT     T
        LDI     R
        OUT     T
AGAIN:  SBIS    T
        RJMP    A
        LDI     R
        OUT     T
        LDI     R
        OUT     T

        EOR     R17,R16     ;toggle D5 of R17
        OUT     PORTB,R17   ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1 << 3;
= ~(1<<3);
1)

 = 0x00;
A = 0x00;
B = 0x01;

TIFR0&(1<<TOV0))==0)

B = 0;
 = 0x01;
 = PORTB^(1<<3);
}
```

Question

Example 3

Question: Modify TCNT0 in Example 2 to get the largest time delay possible with no prescaler. Find the delay in µs. In your calculation, do not include the overhead due to instructions.

- To get the largest delay we make TCNT0 zero. This will count up from 00 to 0xFF and then roll over to zero.

Assembly

```
        LDI     R16,(1<<5)  ;R16=0x20
        SBI     DDRB,5      ;PB5 as an output
        LDI     R
        OUT     P
BEGIN:  LDI     R
        OUT     T
        LDI     R
        OUT     T
        LDI     R
        OUT     T
AGAIN:  SBIS    T
        RJMP    A
        LDI     R
        OUT     T
        LDI     R
        OUT     T

        EOR     R17,R16     ;toggle D5 of R17
        OUT     PORTB,R17   ;toggle PB5
        RJMP    BEGIN
```

C

```
DDRB = 1 << 3;
= ~(1<<3);
1)

 = 0x00;
A = 0x00;
B = 0x01;

TIFR0&(1<<TOV0))==0)

B = 0;
 = 0x01;
 = PORTB^(1<<3);
}
```

Question

Solution

1) Calculating T:

T = 1/f = 1/10MHz = 0.1µs

2) Calculating delay

256 * 0.1µs = 25.6µs

Generating Large Delays

- Using loop
- Prescaler
- Bigger counters

Generating Large Delays

- Using loop
- Prescaler
- Bigger counters

Generating Large Delays

- Using loop
- Prescaler
- Bigger counters

Generating Large Delays

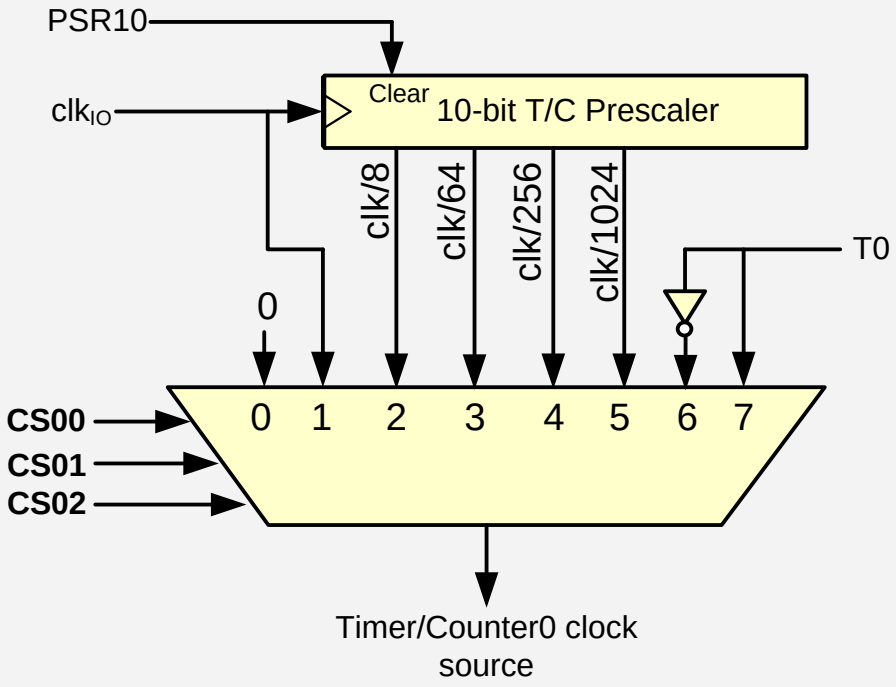
- Using loop
- Prescaler
- Bigger counters

Prescaler and generating a large time delay

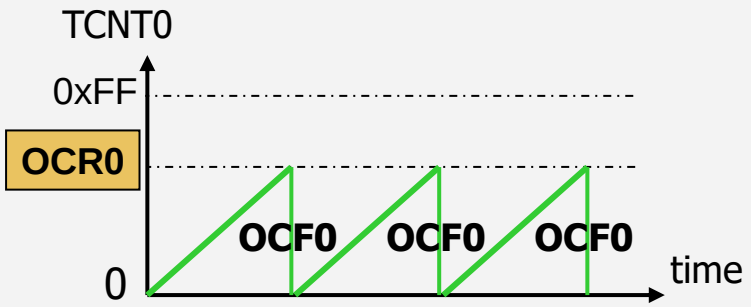
Generating Large Delays

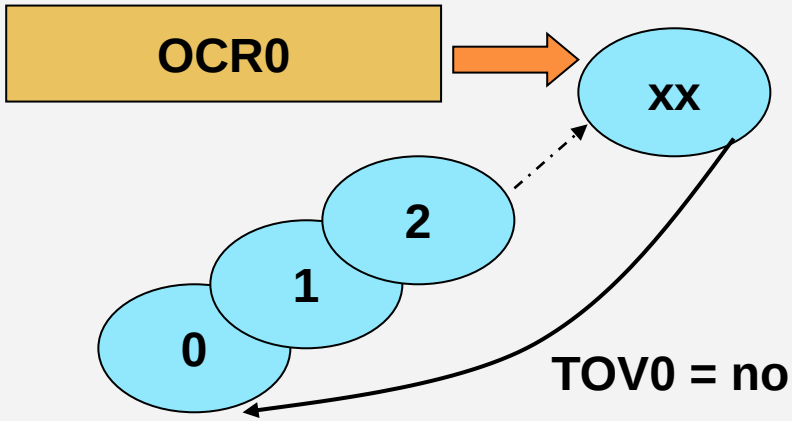
- Using loop
- Prescaler
- Bigger counters

Prescaler and generating a large time delay



CTC (Clear Timer on Compare match) mode





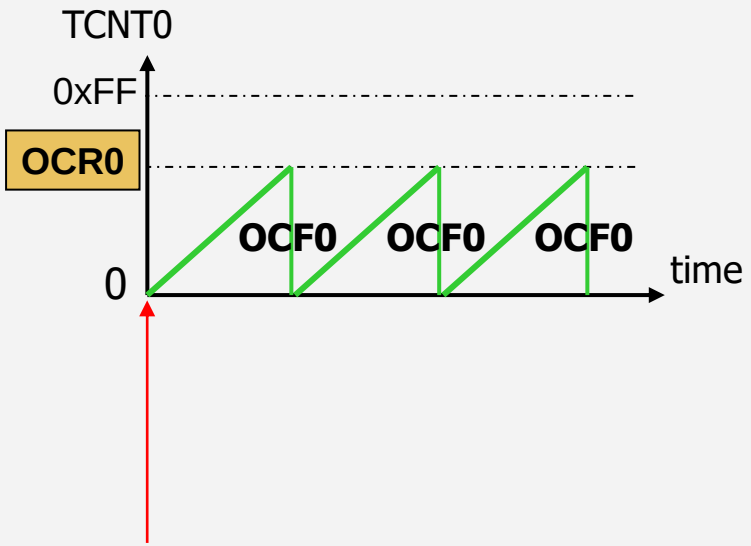
TOV0 = no change

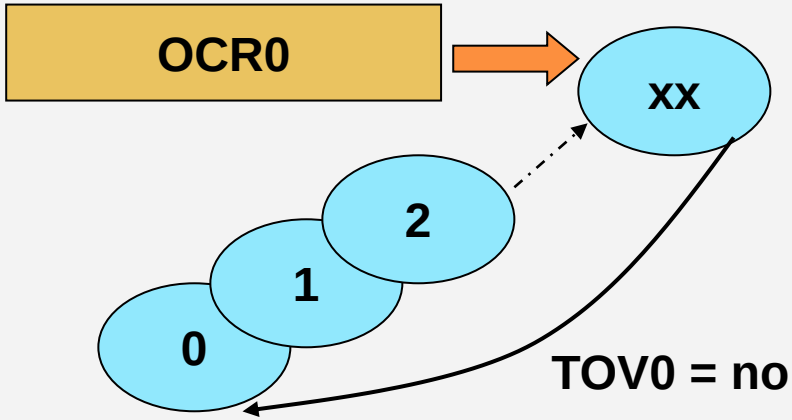
OCF0 = 1

TOV0: 0

OCF0: 0

CTC (Clear Timer on Compare match) mode





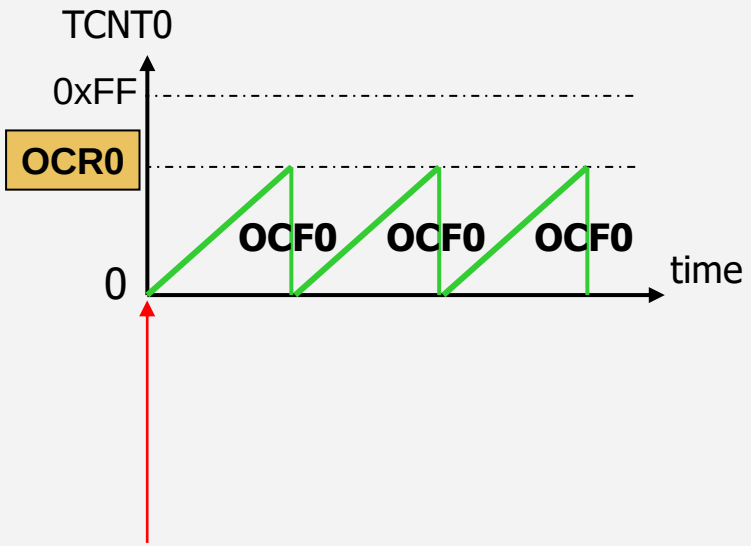
TOV0 = no change

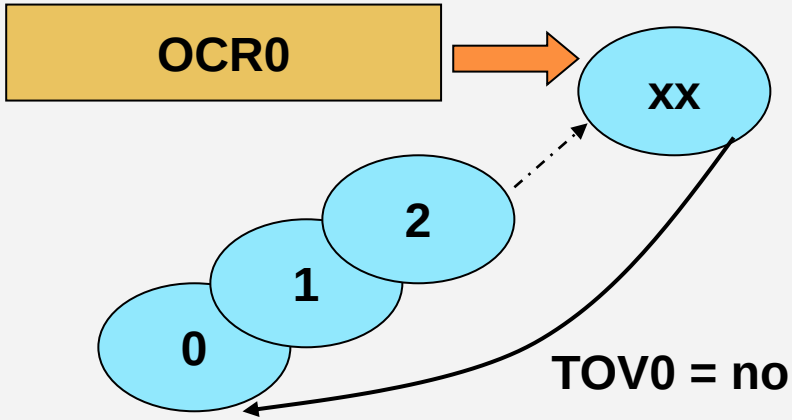
OCF0 = 1

TOV0: 0

OCF0: 0

CTC (Clear Timer on Compare match) mode





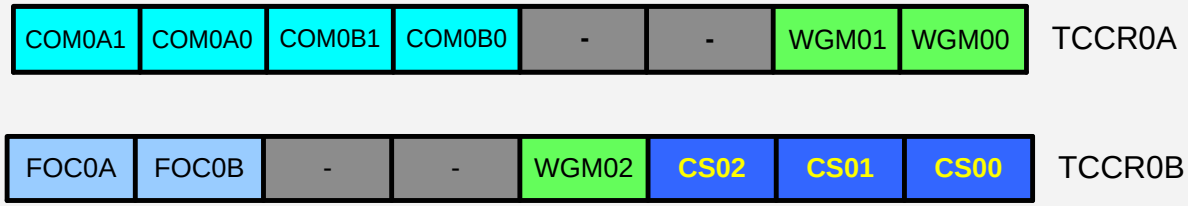
TOV0 = no change

OCF0 = 1

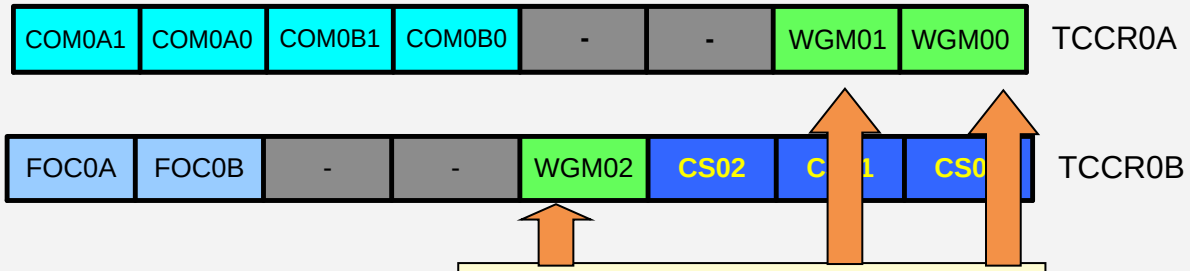
TOV0: 0

OCF0: 1

CTC mode (Cont.)



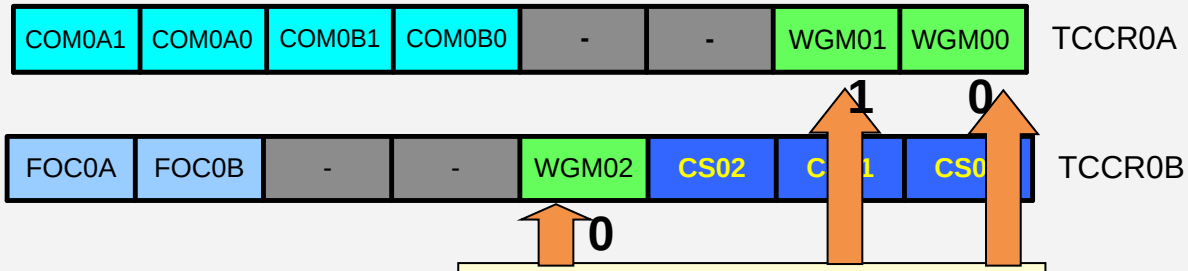
CTC mode (Cont.)



Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM

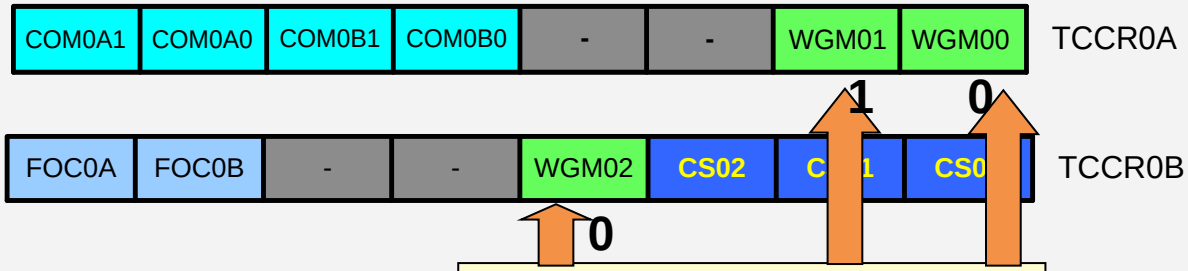
CTC mode (Cont.)



Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM

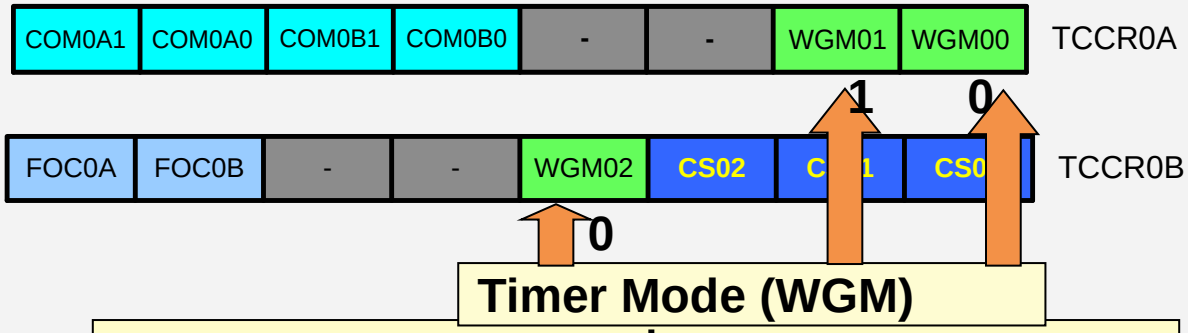
CTC mode (Cont.)



Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM

CTC mode (Cont.)



Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase correct PWM
0	1	0	CTC
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	Phase correct PWM
1	1	0	Reserved
1	1	1	Fast PWM

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 μs.)

- For a square wave with T = 10 μs we must have a time delay of 5 μs. Because XTAL = 10 MHz, the counter counts up every 0.1 μs. This means that we need 5 μs / 0.1 μs = 50 clocks. Therefore, we have OCR0A= 49.

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 μs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 μs we must have a time delay of 5 μs. Because XTAL = 10 MHz, the counter counts up every 0.1 μs. This means that we need 5 μs / 0.1 μs = 50 clocks. Therefore, we have OCR0A= 49.

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 µs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0A= 49.

```
        LDI     R16,(1<<3)          ;R16 = 0x08
        SBI     DDRB,3              ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20 ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)      ;TCCR0A=0x02
        OUT     TCCR0A,R20  ;CTC mode, int. clk
        LDI     R20,1               ; TCCR0B = 1
        OUT     TCCR0B,R20          ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20 ;clear OCF0A flag
        EOR     R17,R16             ;toggle D3 of R17
        OUT     PORTB,R17 ;toggle PB3
        RJMP    BEGIN
```

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 µs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0A= 49.

```
        LDI     R16,(1<<3)          ;R16 = 0x08
        SBI     DDRB,3              ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20 ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)      ;TCCR0A=0x02
        OUT     TCCR0A,R20  ;CTC mode, int. clk
        LDI     R20,1               ; TCCR0B = 1
        OUT     TCCR0B,R20          ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20 ;clear OCF0A flag
        EOR     R17,R16             ;toggle D3 of R17
        OUT     PORTB,R17 ;toggle PB3
        RJMP    BEGIN
```

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 µs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0A= 49.

```
        LDI     R16,(1<<3)          ;R16 = 0x08
        SBI     DDRB,3              ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20 ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)      ;TCCR0A=0x02
        OUT     TCCR0A,R20  ;CTC mode, int. clk
        LDI     R20,1               ; TCCR0B = 1
        OUT     TCCR0B,R20          ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20          ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20 ;clear OCF0A flag
        EOR     R17,R16             ;toggle D3 of R17
        OUT     PORTB,R17 ;toggle PB3
        RJMP    BEGIN
```

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 µs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0A= 49.

Assembly

```
        LDI     R16,(1<<3)              ;R16 = 0x08
        SBI     DDRB,3                  ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20 ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)          ;TCCR0A=0x02
        OUT     TCCR0A,R20   ;CTC mode, int. clk
        LDI     R20,1                   ; TCCR0B = 1
        OUT     TCCR0B,R20              ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20              ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20 ;clear OCF0A flag
        EOR     R17,R16                 ;toggle D3 of R17
        OUT     PORTB,R17 ;toggle PB3
        RJMP    BEGIN
```

```
DDRB |= 1<<3;
while (1)
{
  OCR0A = 49;
  TCCR0A = (1<<WGM01); //CTC
  TCCR0B = 1;  //N = 1
 while((TIFR0&(1<<OCF0A))==0)
  { }
  TCCR0B = 0; //stop timer0
  TIFR0 = 1<<OCF0A;
  PORTB ^= 1<<3; //toggle PB3
}
```

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 µs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0A= 49.

Assembly

```
        LDI     R16,(1<<3)              ;R16 = 0x08
        SBI     DDRB,3                  ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20 ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)          ;TCCR0A=0x02
        OUT     TCCR0A,R20   ;CTC mode, int. clk
        LDI     R20,1                   ; TCCR0B = 1
        OUT     TCCR0B,R20              ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20              ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20 ;clear OCF0A flag
        EOR     R17,R16                 ;toggle D3 of R17
        OUT     PORTB,R17 ;toggle PB3
        RJMP    BEGIN
```

```
DDRB |= 1<<3;
while (1)
{
  OCR0A = 49;
  TCCR0A = (1<<WGM01); //CTC
  TCCR0B = 1;  //N = 1
 while((TIFR0&(1<<OCF0A))==0)
  { }
  TCCR0B = 0; //stop timer0
  TIFR0 = 1<<OCF0A;
  PORTB ^= 1<<3; //toggle PB3
}
```

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 µs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0A= 49.

Assembly

```
        LDI     R16,(1<<3)              ;R16 = 0x08
        SBI     DDRB,3                  ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20 ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)          ;TCCR0A=0x02
        OUT     TCCR0A,R20   ;CTC mode, int. clk
        LDI     R20,1                   ; TCCR0B = 1
        OUT     TCCR0B,R20              ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20              ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20 ;clear OCF0A flag
        EOR     R17,R16                 ;toggle D3 of R17
        OUT     PORTB,R17 ;toggle PB3
        RJMP    BEGIN
```

C

```
DDRB |= 1<<3;
while (1)
{
  OCR0A = 49;
  TCCR0A = (1<<WGM01); //CTC
  TCCR0B = 1;  //N = 1
 while((TIFR0&(1<<OCF0A))==0)
  { }
  TCCR0B = 0; //stop timer0
  TIFR0 = 1<<OCF0A;
  PORTB ^= 1<<3; //toggle PB3
}
```

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 µs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0A= 49.

Assembly

```
        LDI     R16,(1<<3)              ;R16 = 0x08
        SBI     DDRB,3                  ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20 ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)          ;TCCR0A=0x02
        OUT     TCCR0A,R20   ;CTC mode, int. clk
        LDI     R20,1                   ; TCCR0B = 1
        OUT     TCCR0B,R20              ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20              ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20 ;clear OCF0A flag
        EOR     R17,R16                 ;toggle D3 of R17
        OUT     PORTB,R17 ;toggle PB3
        RJMP    BEGIN
```

C

```
DDRB |= 1<<3;
while (1)
{
  OCR0A = 49;
  TCCR0A = (1<<WGM01); //CTC
  TCCR0B = 1;  //N = 1
 while((TIFR0&(1<<OCF0A))==0)
  { }
  TCCR0B = 0; //stop timer0
  TIFR0 = 1<<OCF0A;
  PORTB ^= 1<<3; //toggle PB3
}
```

Example 4: Rewrite example 2 using CTC

(Assuming XTAL = 10MHz, write a program to generate a square wave with a period of 10 µs.)

-	-	-	-	-	OCF0B	OCF0A	TOV0

TIFR0

- For a square wave with T = 10 µs we must have a time delay of 5 µs. Because XTAL = 10 MHz, the counter counts up every 0.1 µs. This means that we need 5 µs / 0.1 µs = 50 clocks. Therefore, we have OCR0A= 49.

Assembly

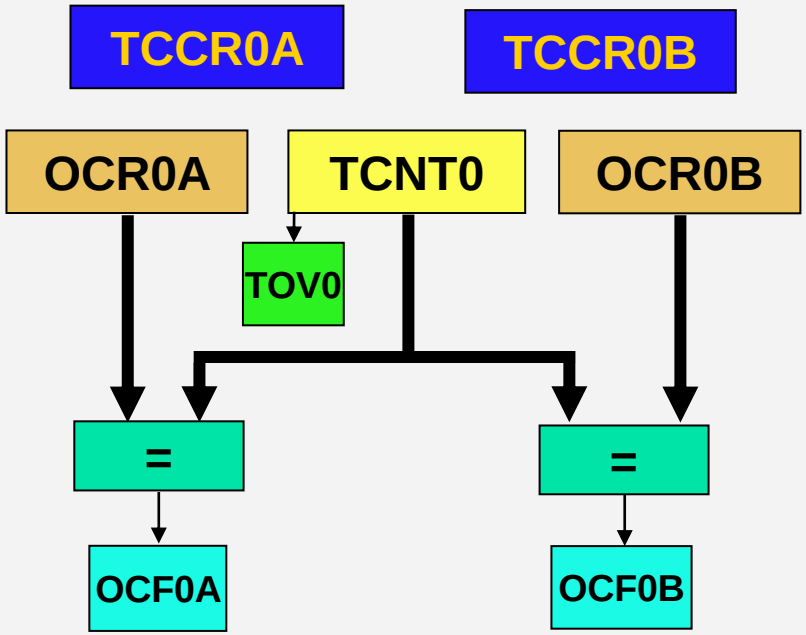
```
        LDI     R16,(1<<3)              ;R16 = 0x08
        SBI     DDRB,3                  ;PB3 as an output
        LDI     R17,0
        OUT     PORTB,R17
        LDI     R20,49
        OUT     OCR0A,R20 ;load OCR0A
BEGIN:  LDI     R20,(1<<WGM01)          ;TCCR0A=0x02
        OUT     TCCR0A,R20   ;CTC mode, int. clk
        LDI     R20,1                   ; TCCR0B = 1
        OUT     TCCR0B,R20              ; N = 1
AGAIN:  SBIS    TIFR0,OCF0A ;if OCF0 is set skip next
        RJMP    AGAIN
        LDI     R20,0x0
        OUT     TCCR0B,R20              ;stop Timer0
        LDI     R20,1<<OCF0A
        OUT     TIFR0,R20 ;clear OCF0A flag
        EOR     R17,R16                 ;toggle D3 of R17
        OUT     PORTB,R17 ;toggle PB3
        RJMP    BEGIN
```

C

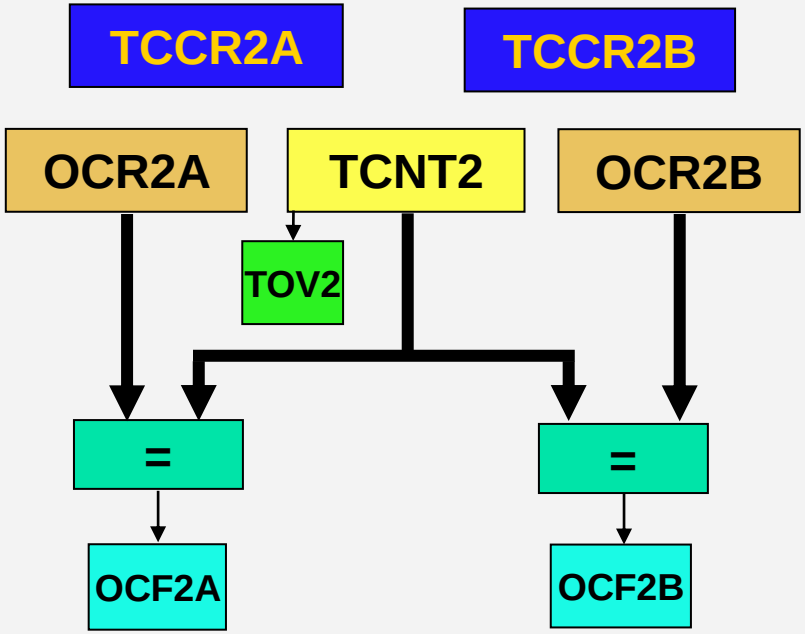
```
DDRB |= 1<<3;
while (1)
{
  OCR0A = 49;
  TCCR0A = (1<<WGM01); //CTC
  TCCR0B = 1;  //N = 1
 while((TIFR0&(1<<OCF0A))==0)
  { }
  TCCR0B = 0; //stop timer0
  TIFR0 = 1<<OCF0A;
  PORTB ^= 1<<3; //toggle PB3
}
```

Timer2 vs. Timer0

- Timer0



- Timer2



The difference between Timer0 and Timer2

- Timer0

CS02	CS01	CS00	Comment
0	0	0	Timer/Counter stopped
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock (falling edge)
1	1	1	External clock (rising edge)

- Timer2

CS22	CS21	CS20	Comment
0	0	0	Timer/Counter stopped
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 32
1	0	0	clk / 64
1	0	1	clk / 128
1	1	0	clk / 256
1	1	1	clk / 1024

Timer 1

Timer 1

TCNT1H TCNT1L

-	-	ICF1	-	-	OCF1B	OCF1A	TOV1

TIFR1

Timer 1

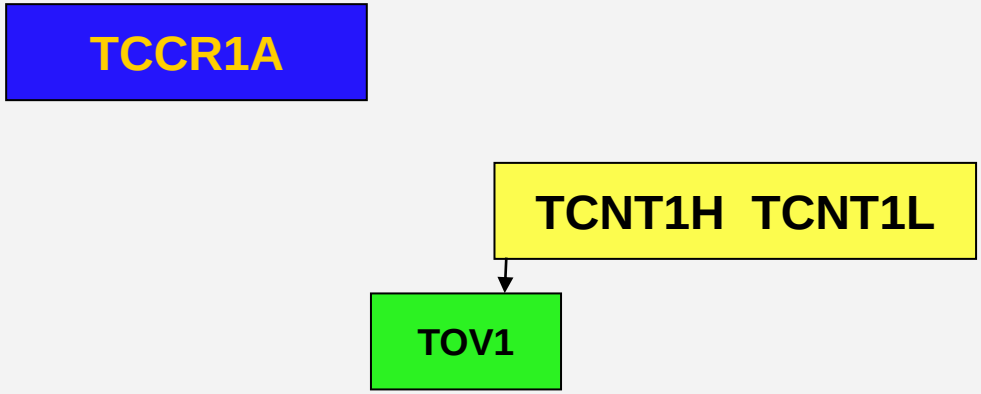
TCCR1A

TCNT1H TCNT1L

TOV1

TIFR1

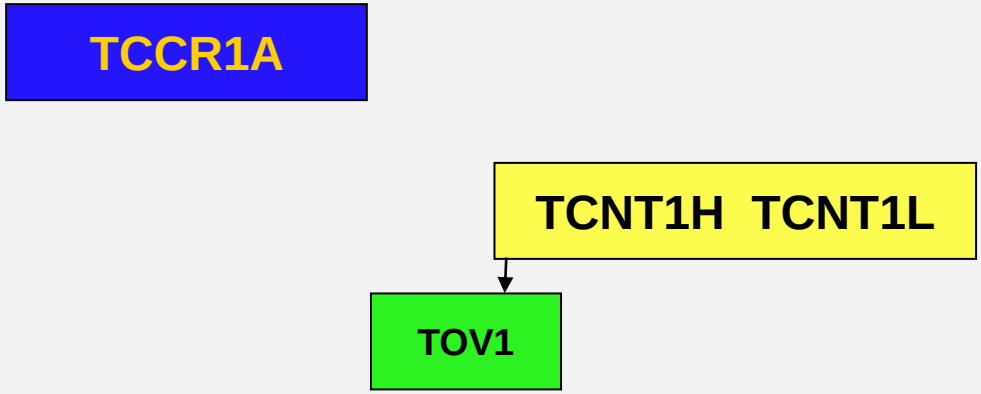
Timer 1



-	-	ICF1	-	-	OCF1B	OCF1A	TOV1

TIFR1

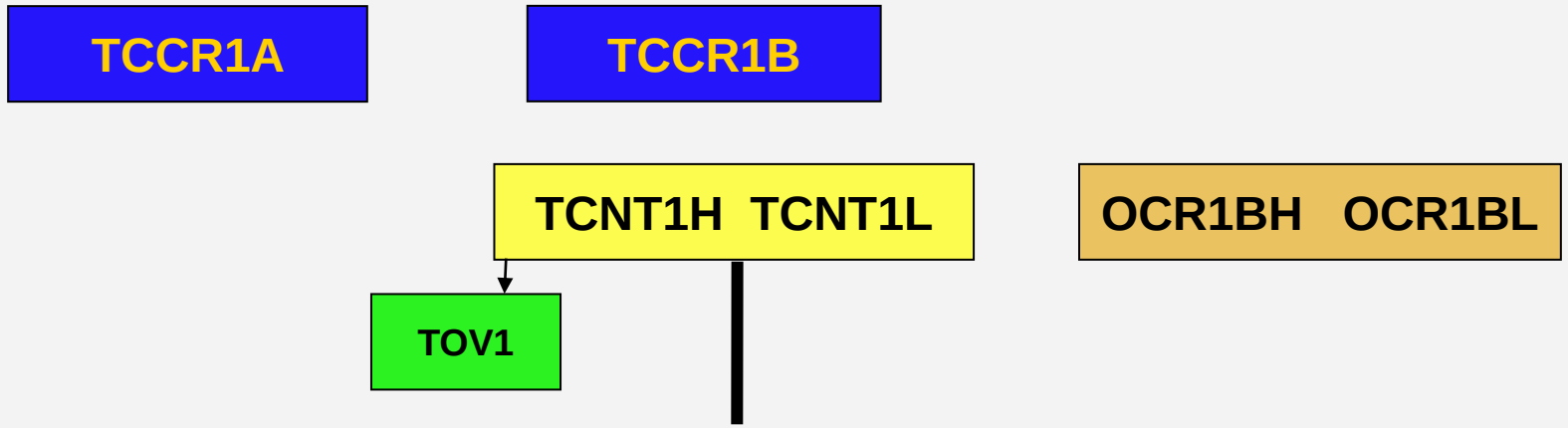
Timer 1



-	-	ICF1	-	-	OCF1B	OCF1A	TOV1

TIFR1

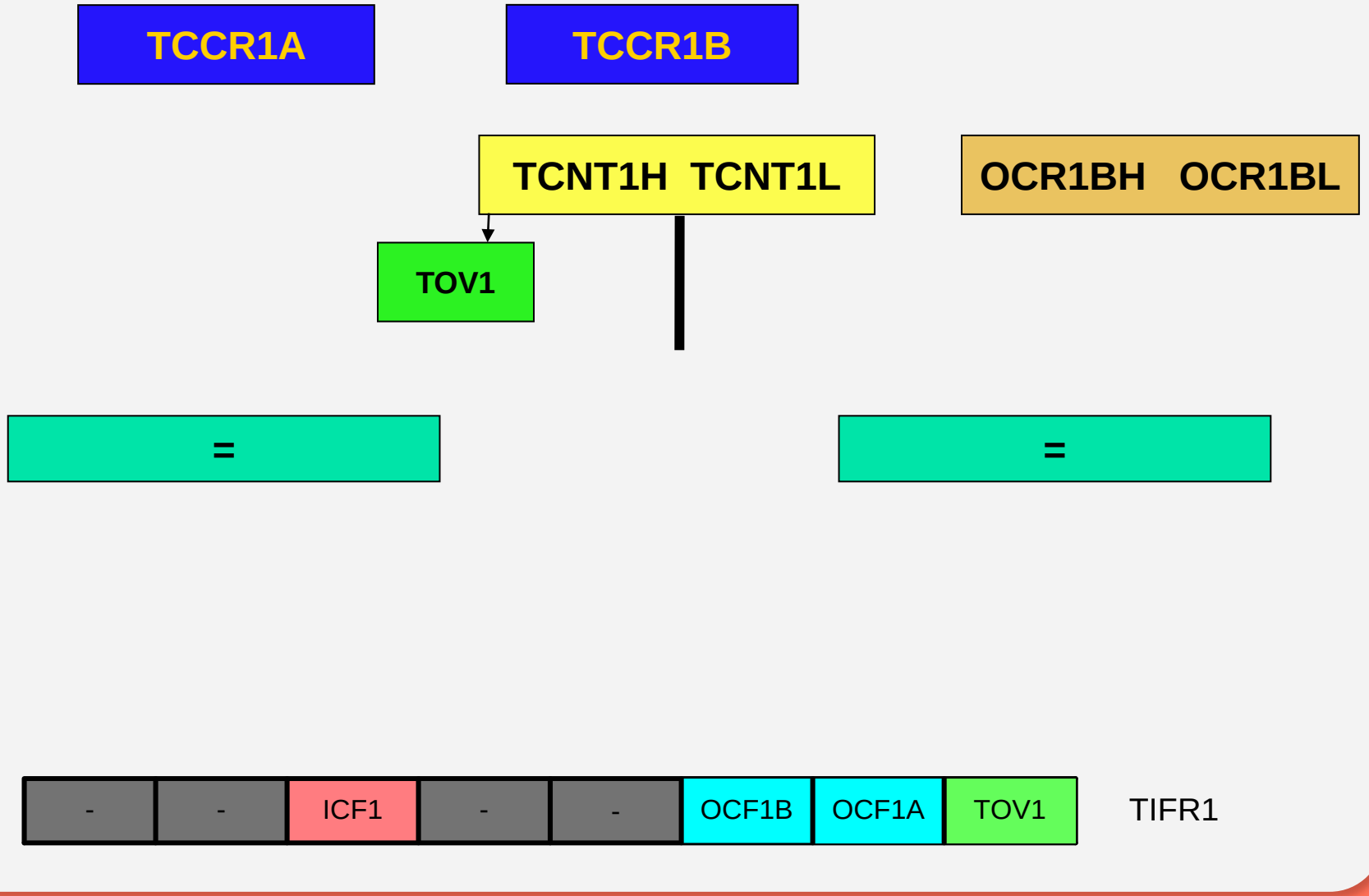
Timer 1



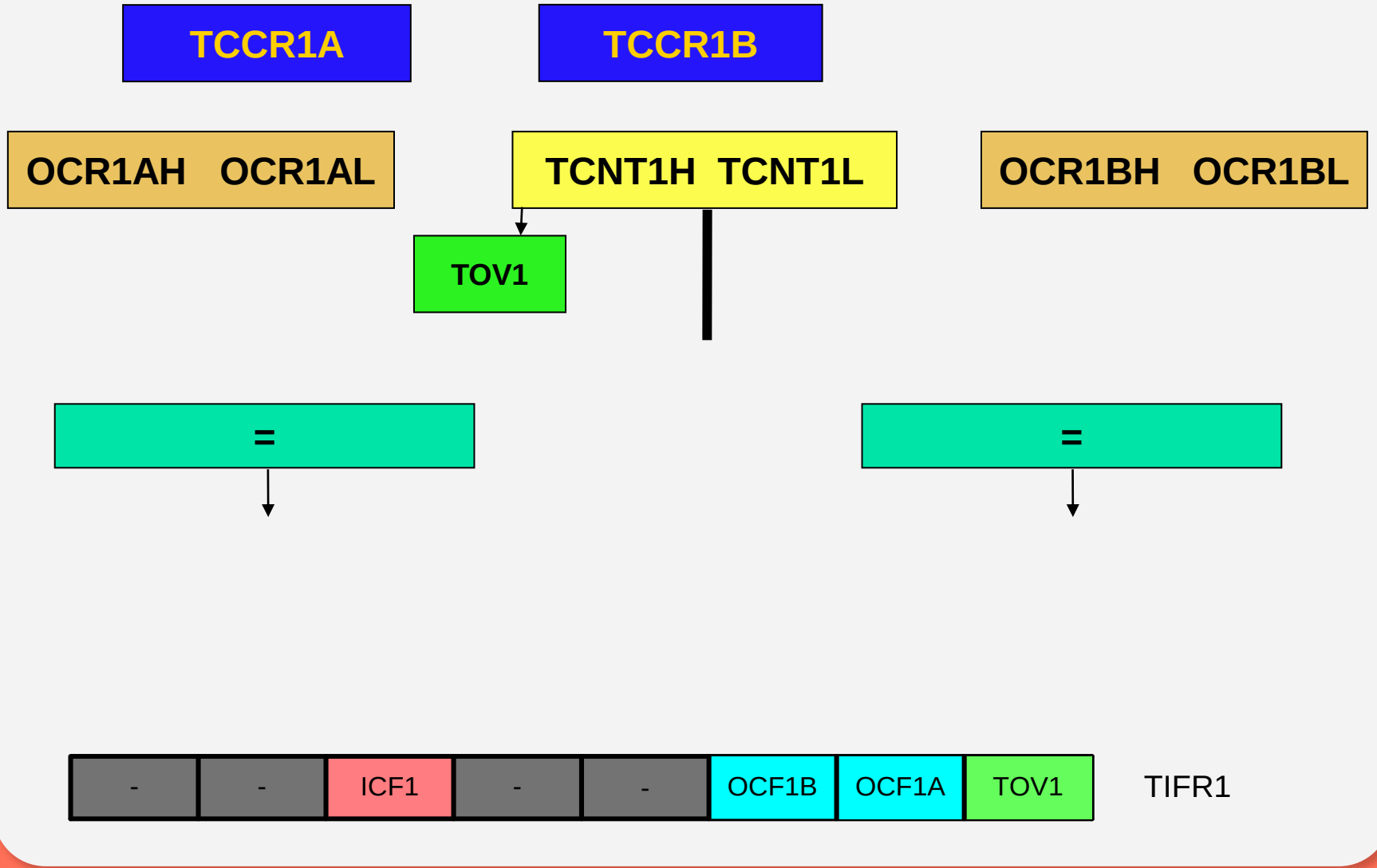
-	-	ICF1	-	-	OCF1B	OCF1A	TOV1

TIFR1

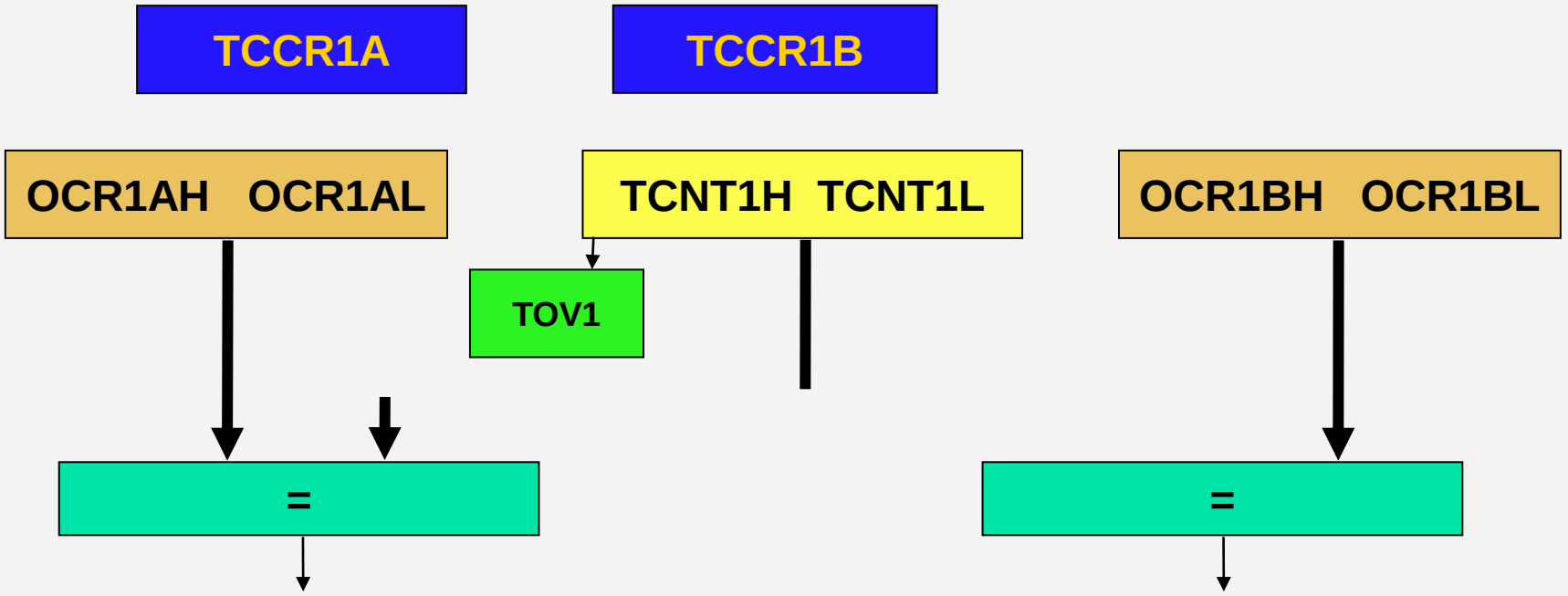
Timer 1


TCCR1A
TCCR1B
TCNT1H TCNT1L
OCR1BH OCR1BL
TOV1
=
=
- - ICF1 - - OCF1B OCF1A TOV1
TIFR1

Timer 1

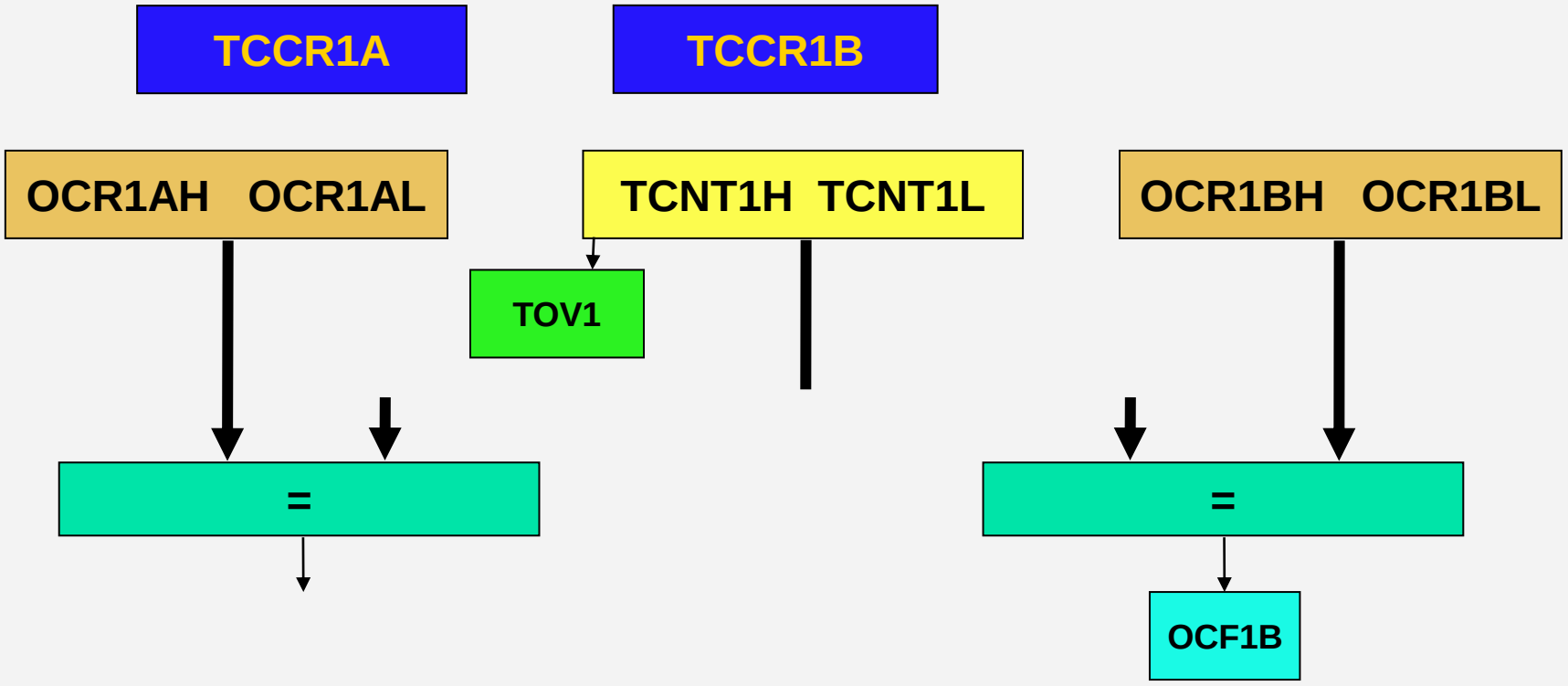

TCCR1A
TCCR1B
OCR1AH OCR1AL
TCNT1H TCNT1L
OCR1BH OCR1BL
TOV1
=
=
- - ICF1 - - OCF1B OCF1A TOV1
TIFR1

Timer 1



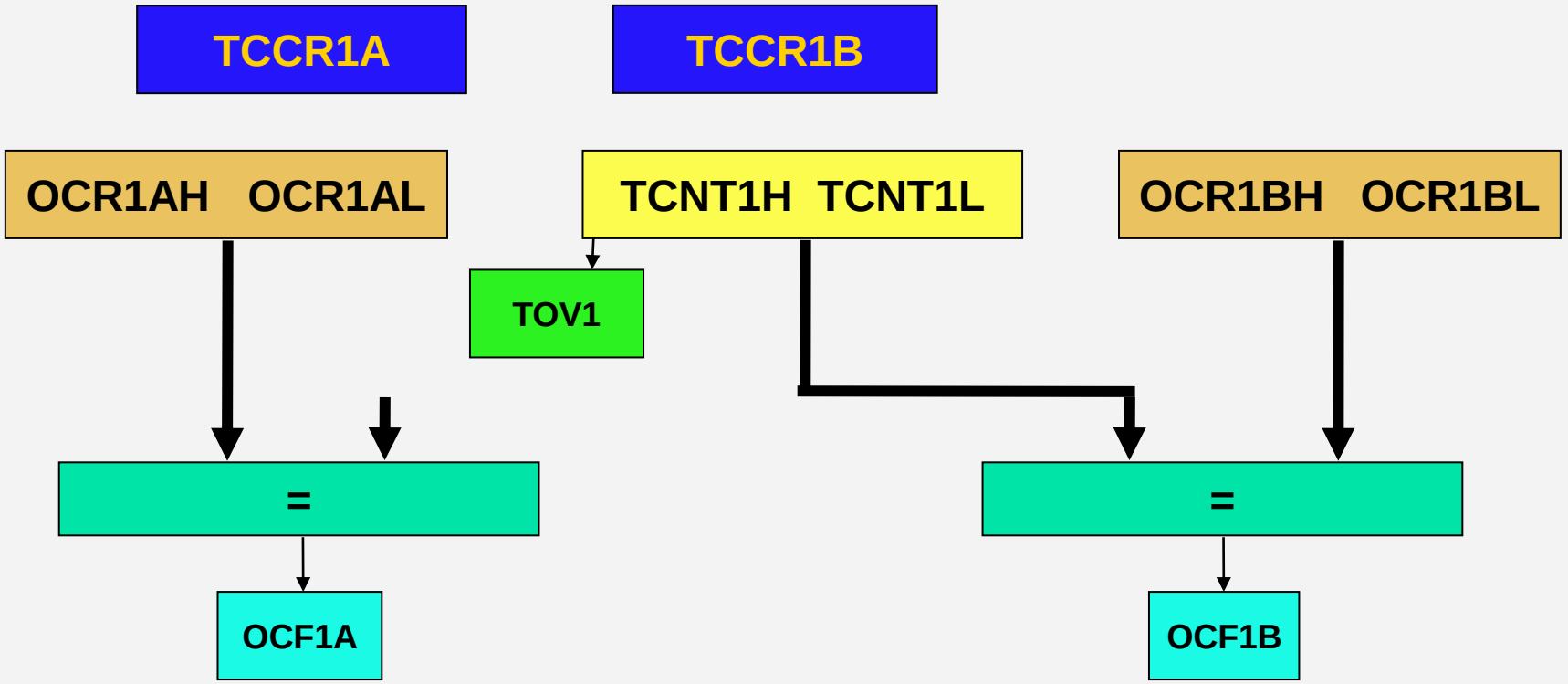
TIFR1

Timer 1



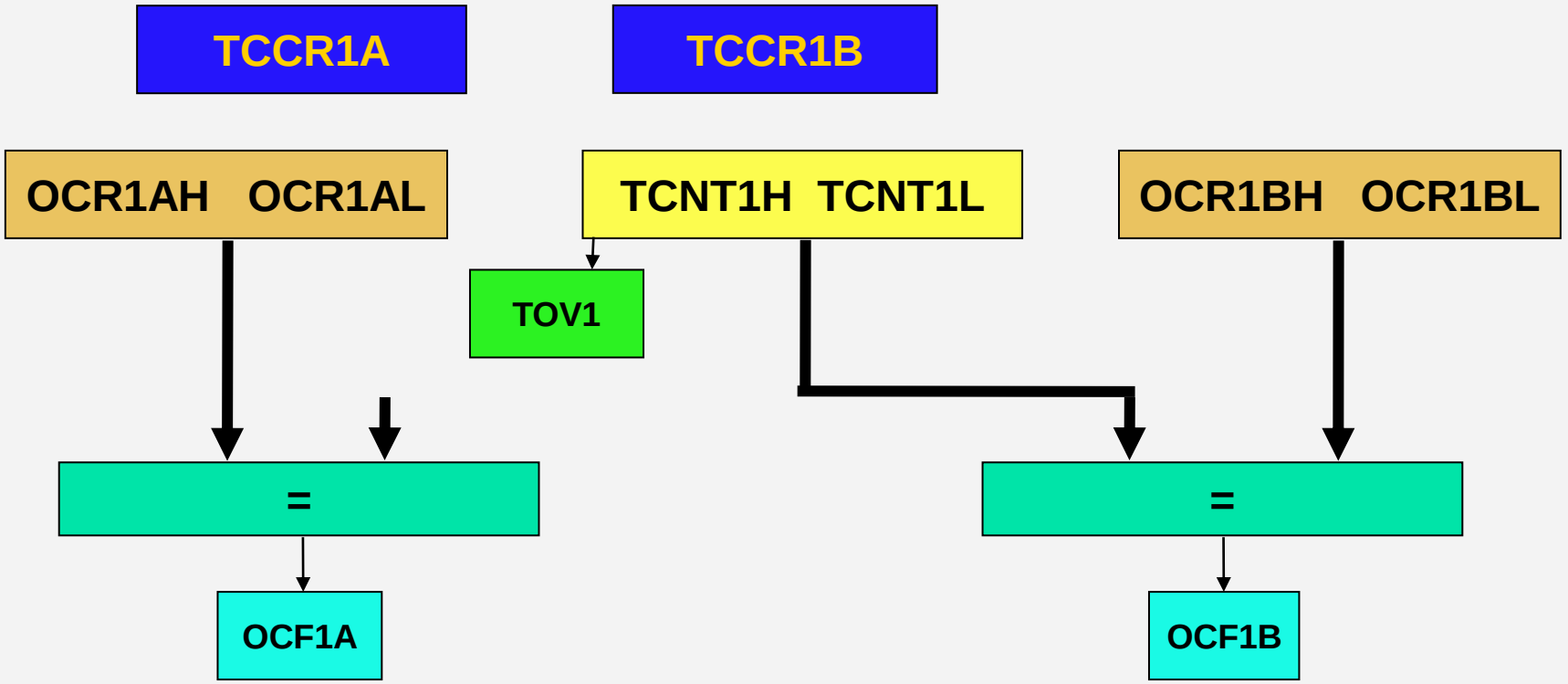
TIFR1

Timer 1



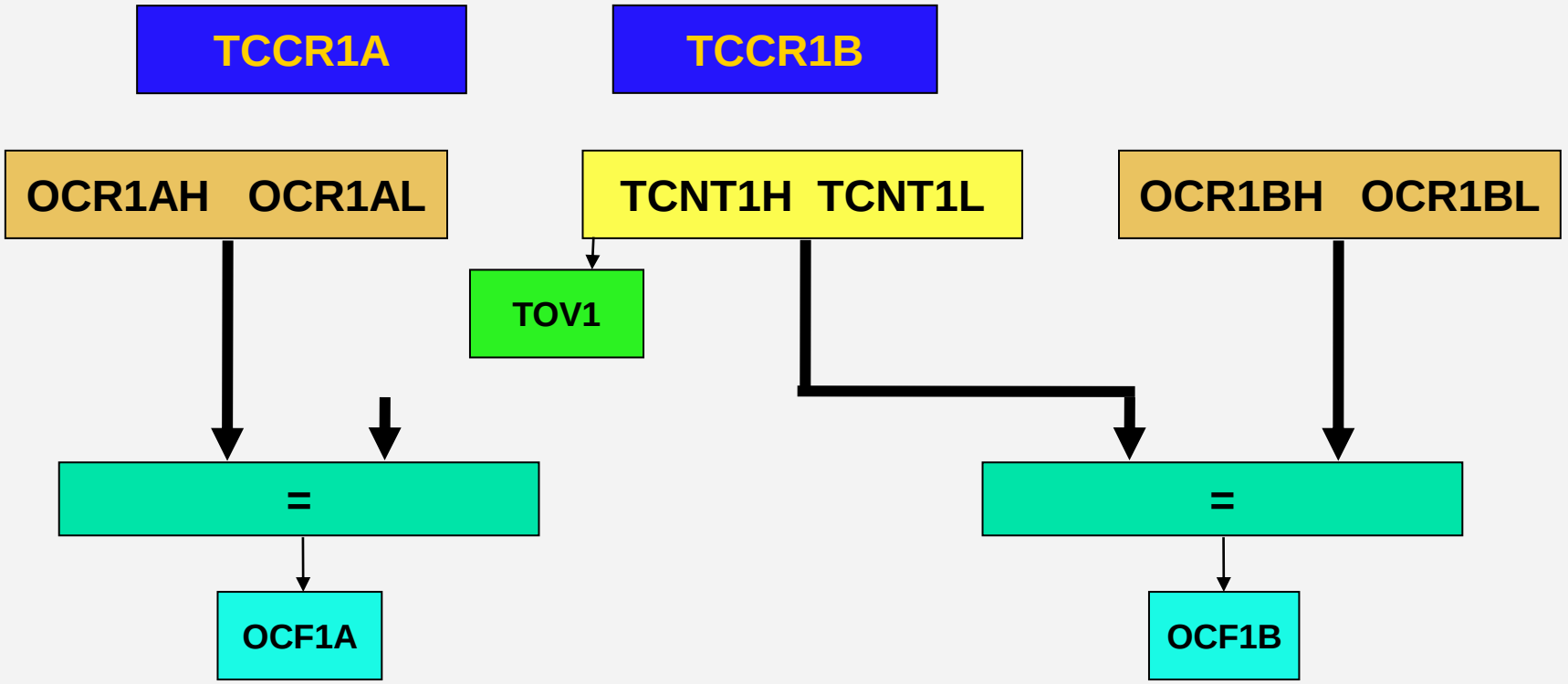
TIFR1

Timer 1



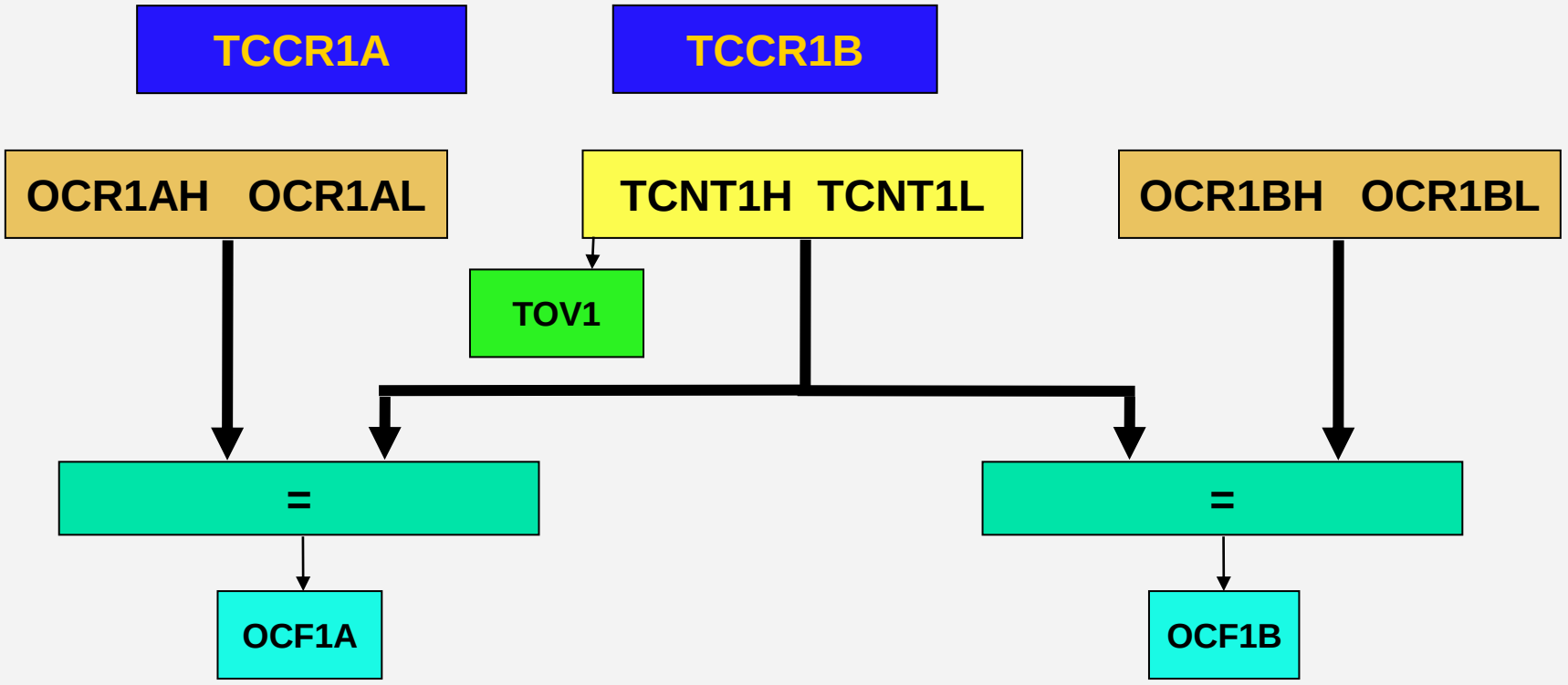
TIFR1

Timer 1



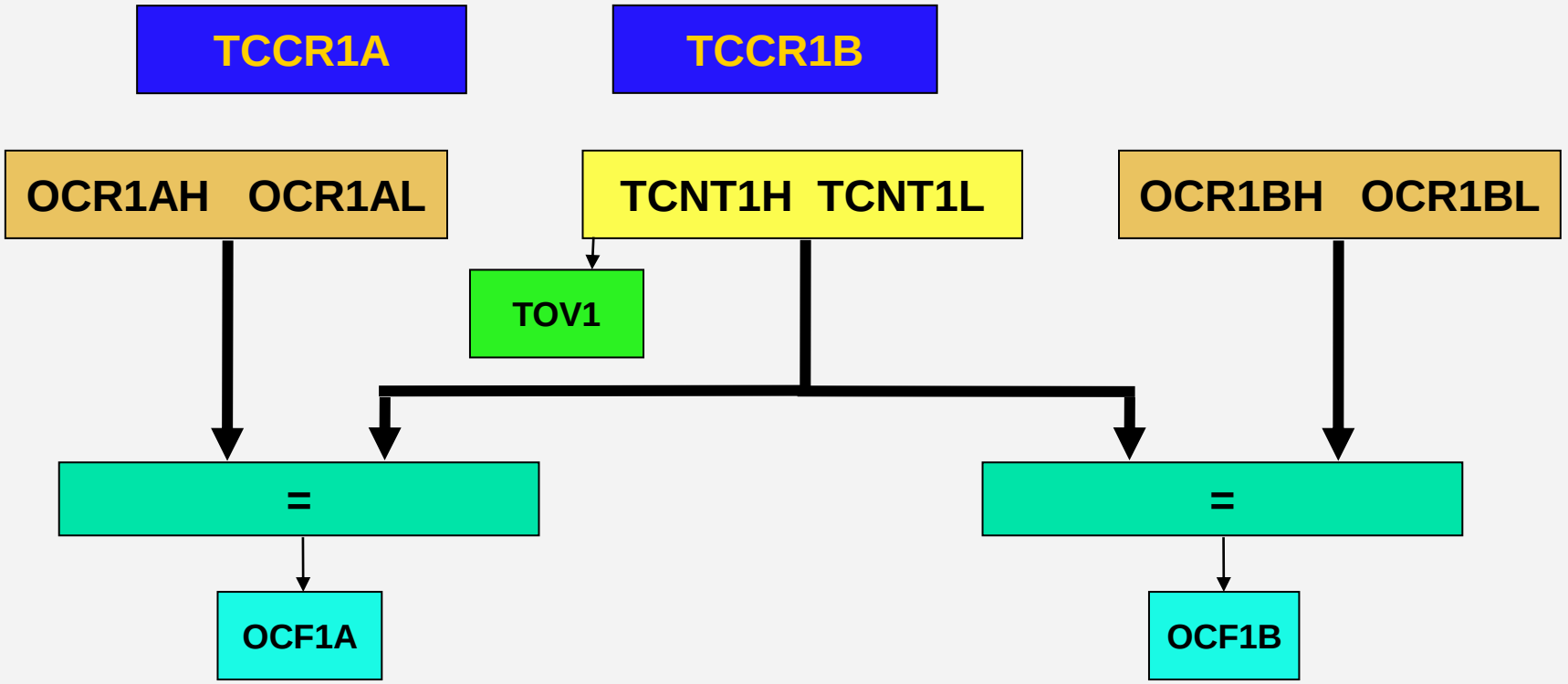
TIFR1

Timer 1



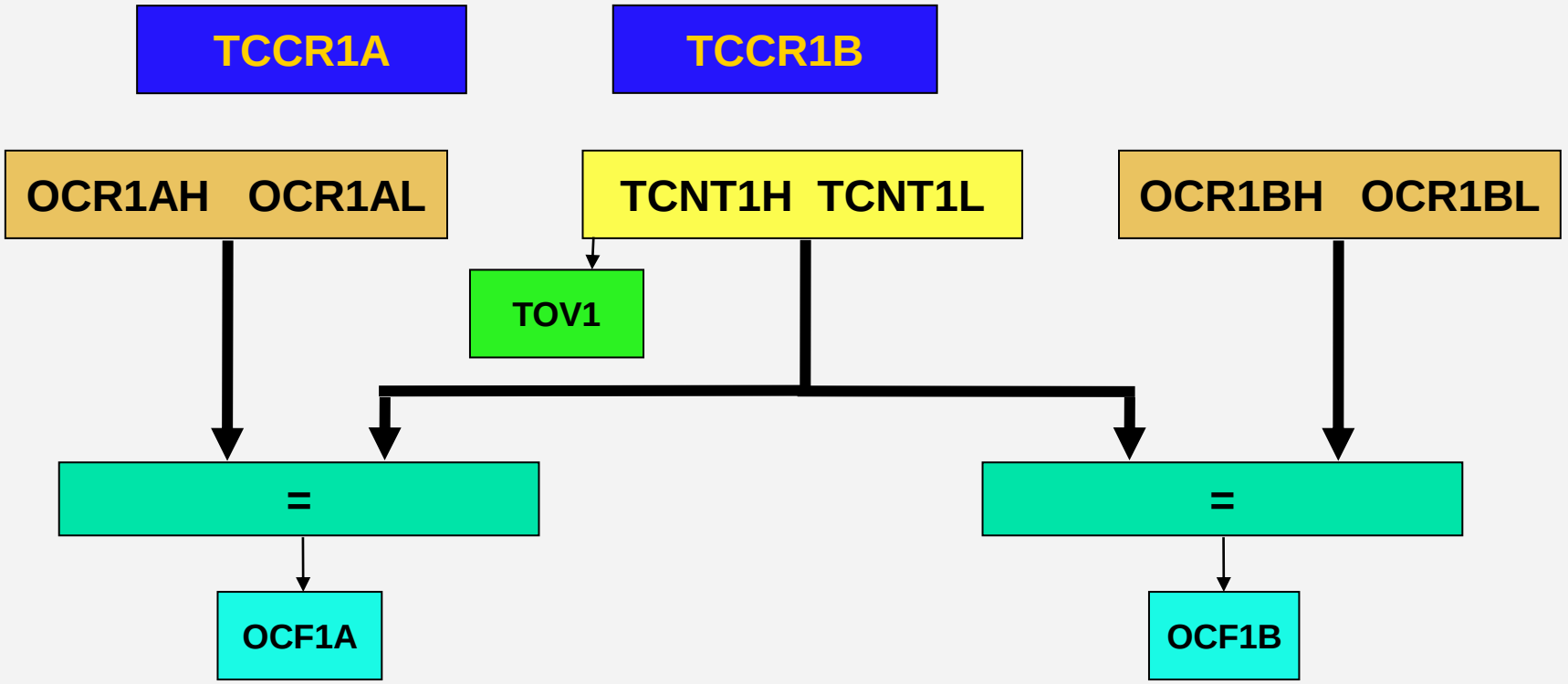
TIFR1

Timer 1



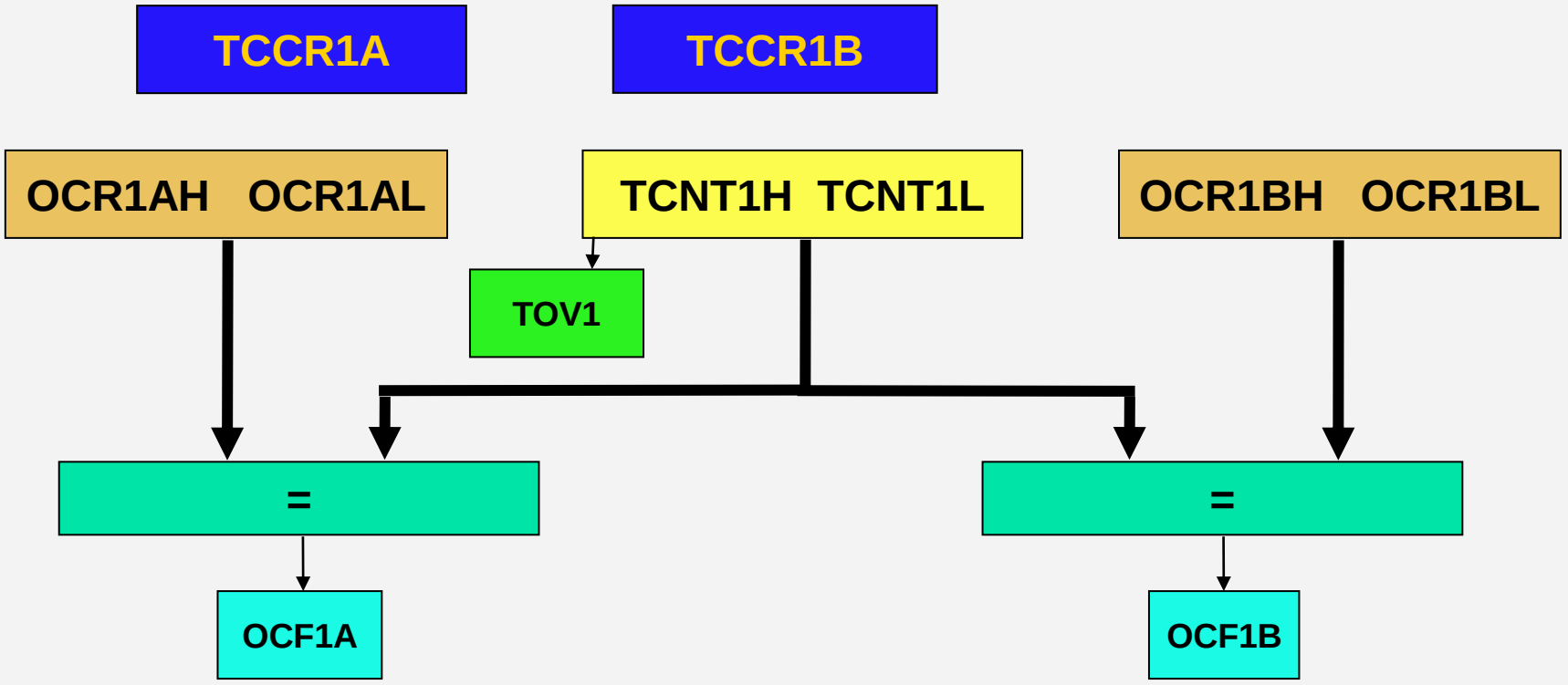
TIFR1

Timer 1



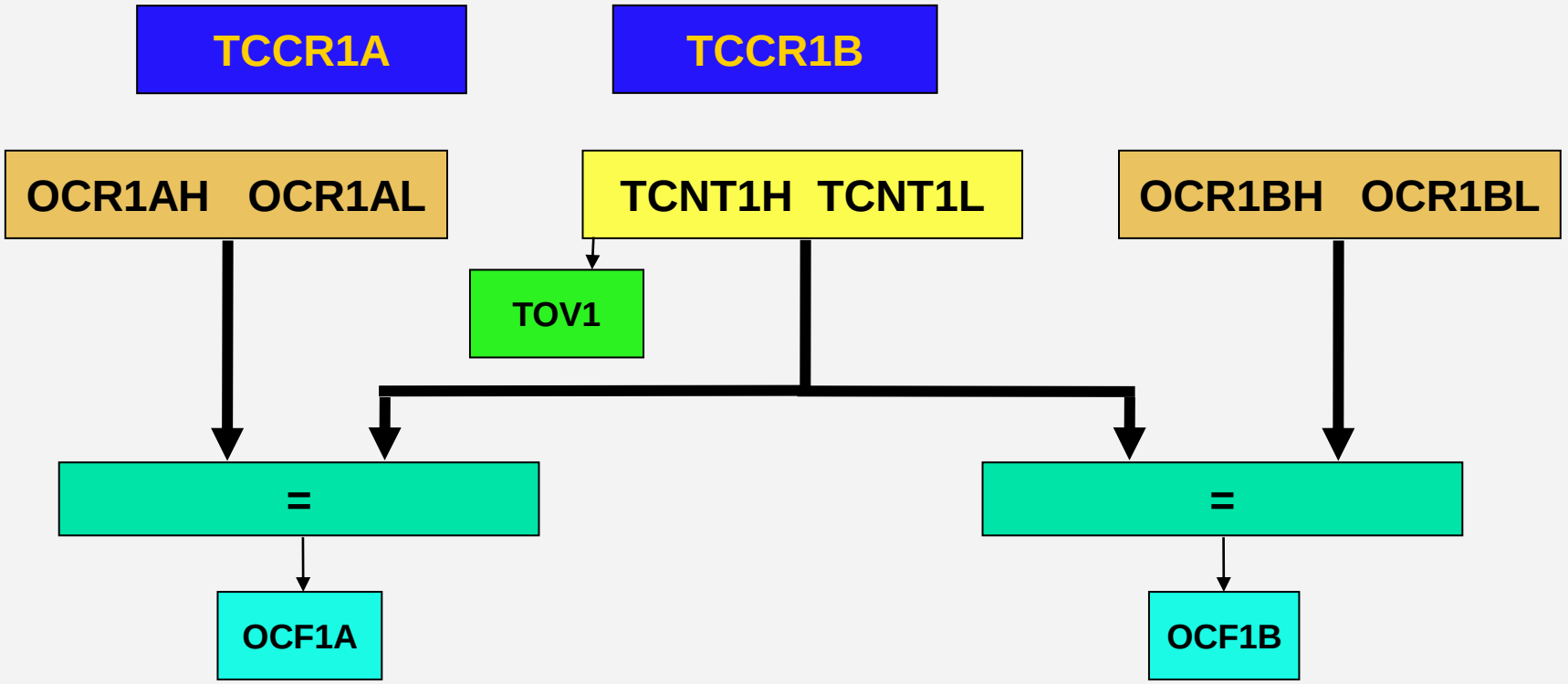
TIFR1

Timer 1



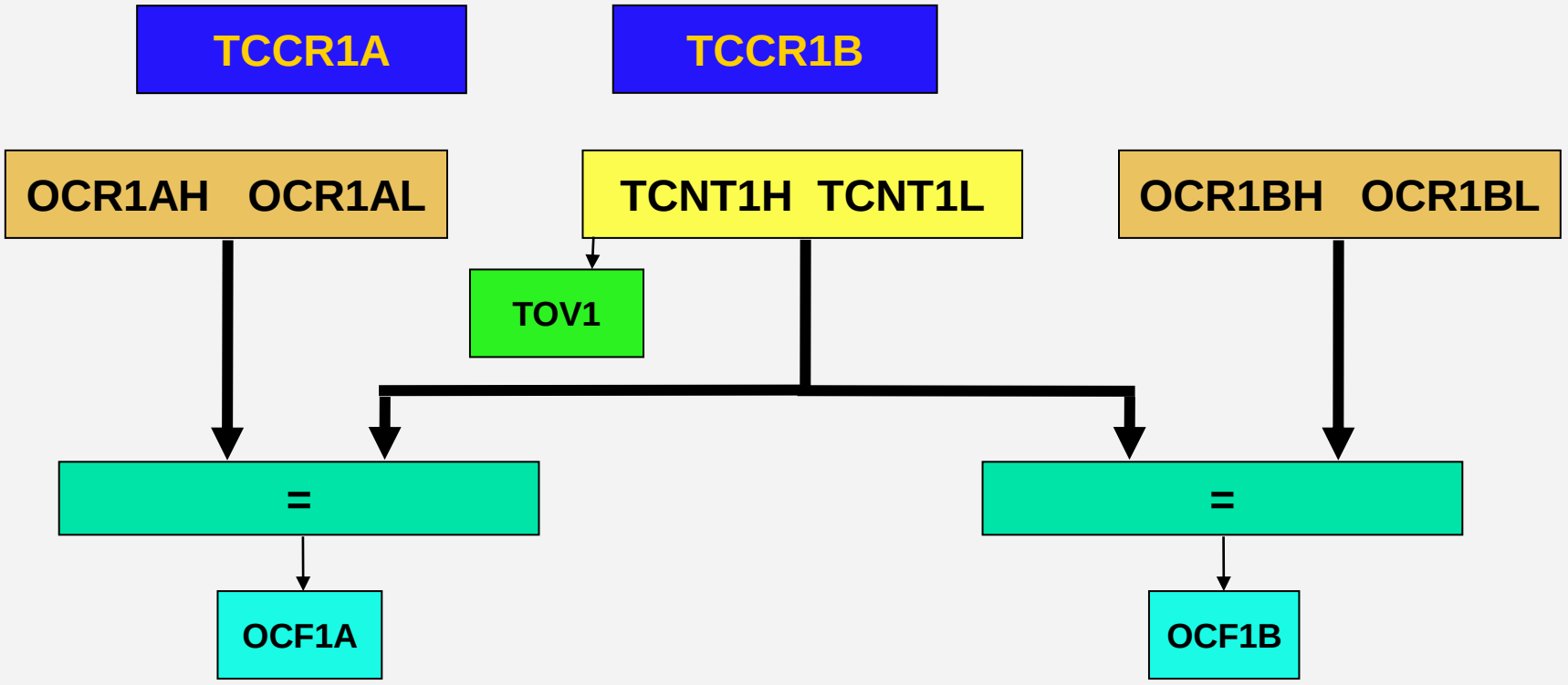
TIFR1

Timer 1



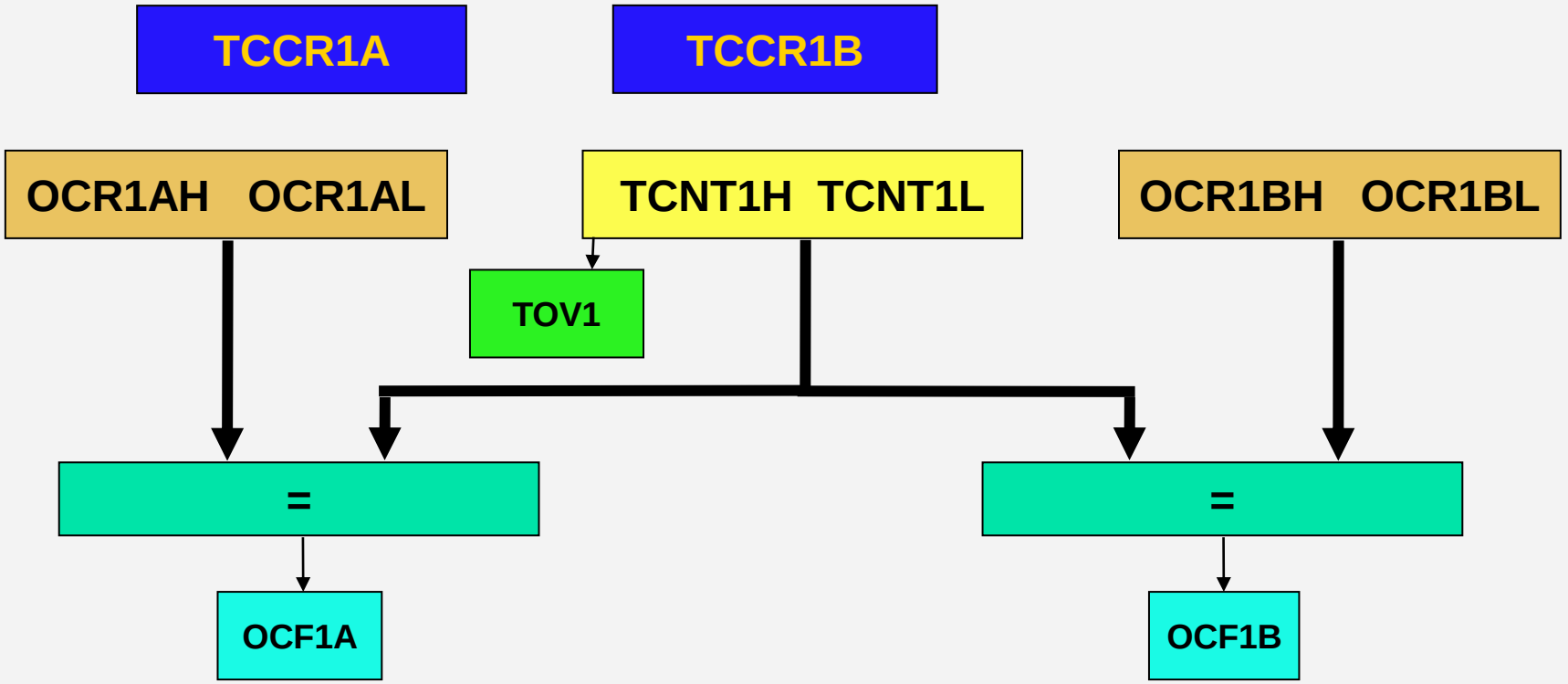
TIFR1

Timer 1



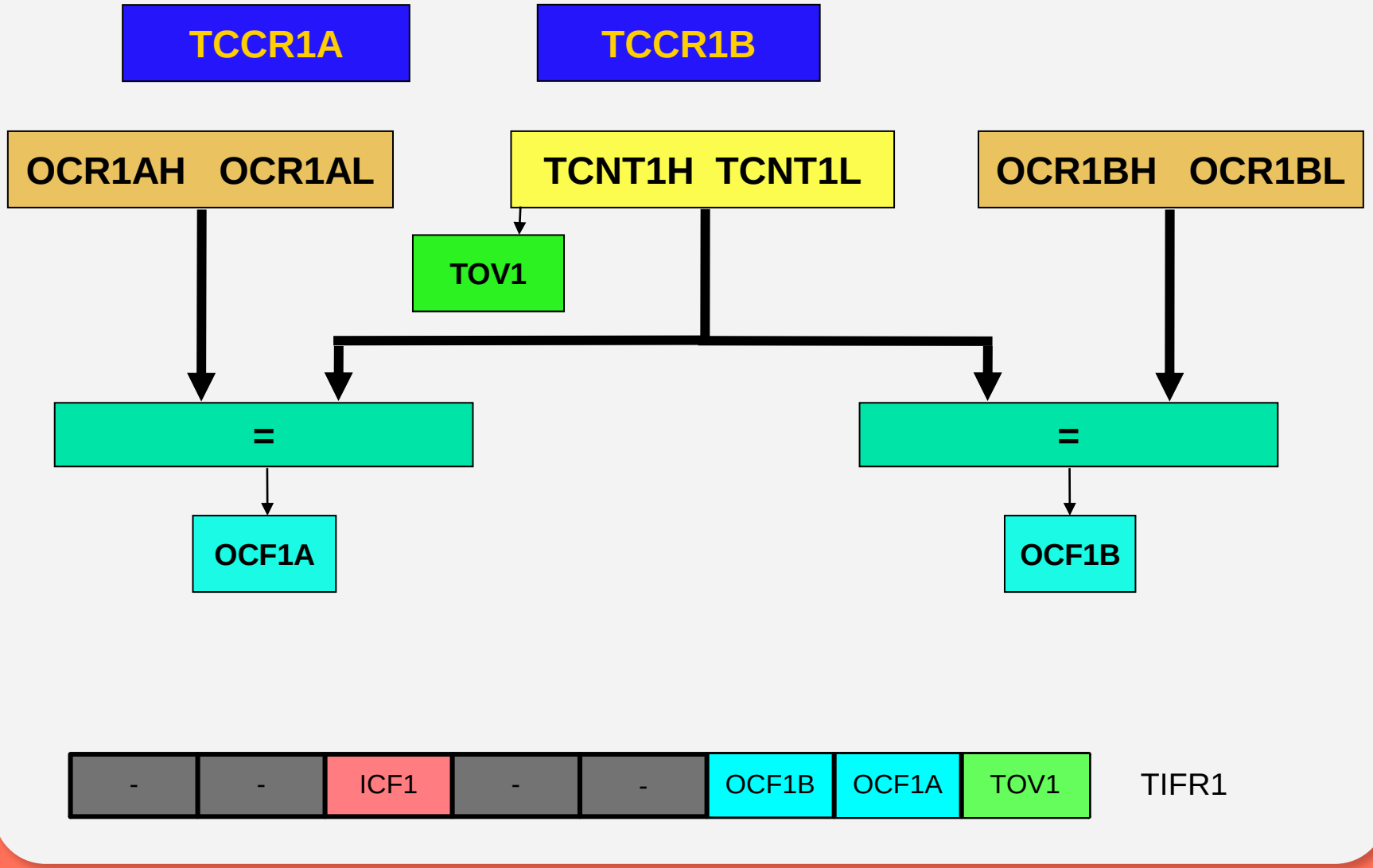
TIFR1

Timer 1

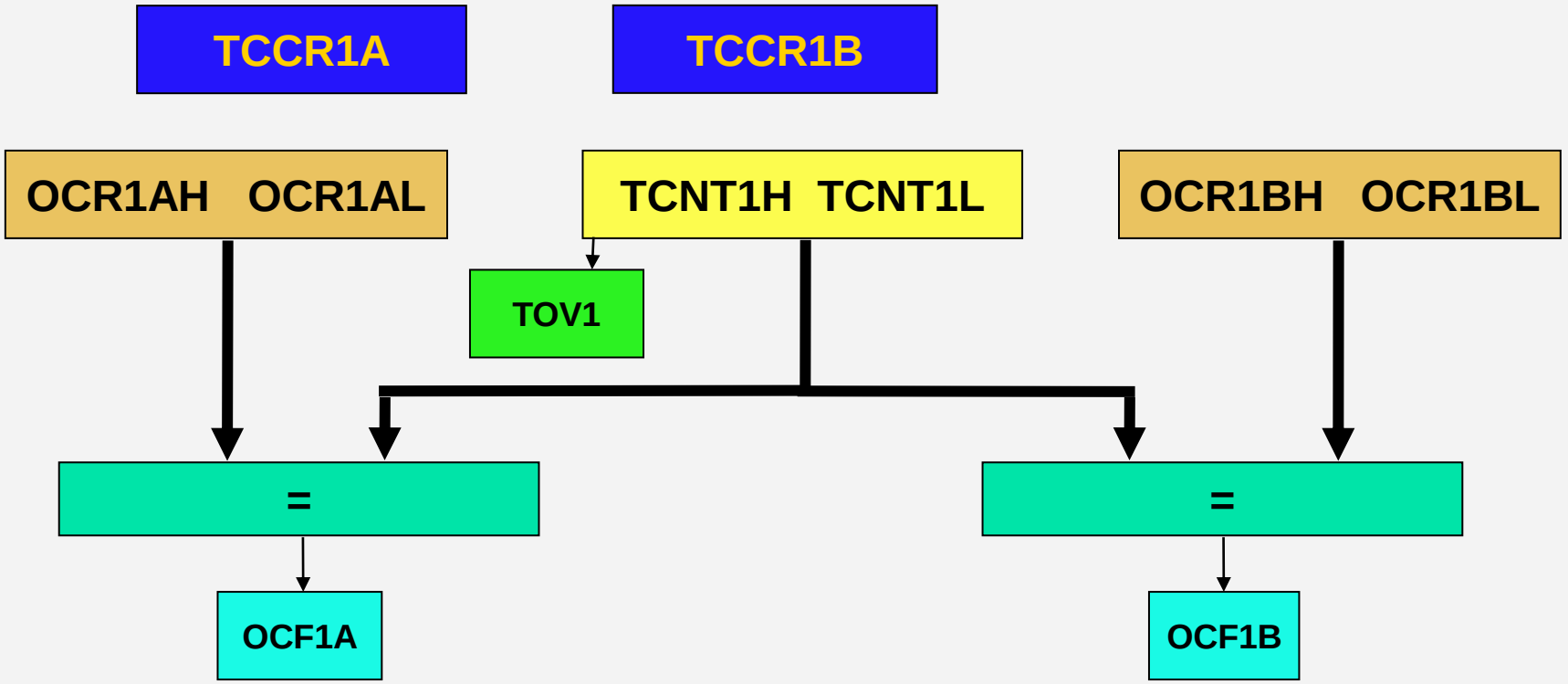


TIFR1

Timer 1

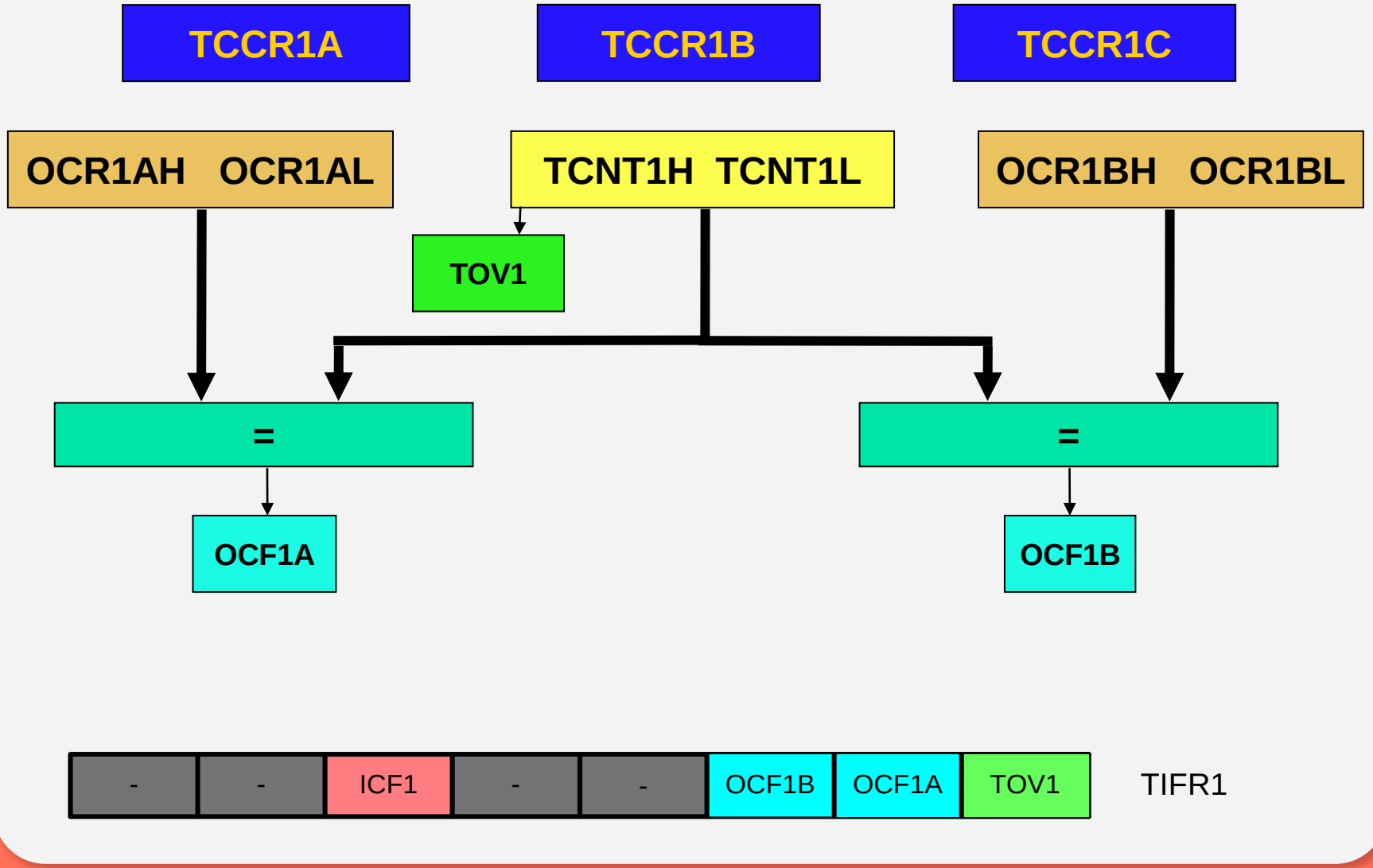

TCCR1A
TCCR1B
OCR1AH OCR1AL
TCNT1H TCNT1L
OCR1BH OCR1BL
TOV1
=
=
OCF1A
OCF1B
- - ICF1 - - OCF1B OCF1A TOV1
TIFR1

Timer 1

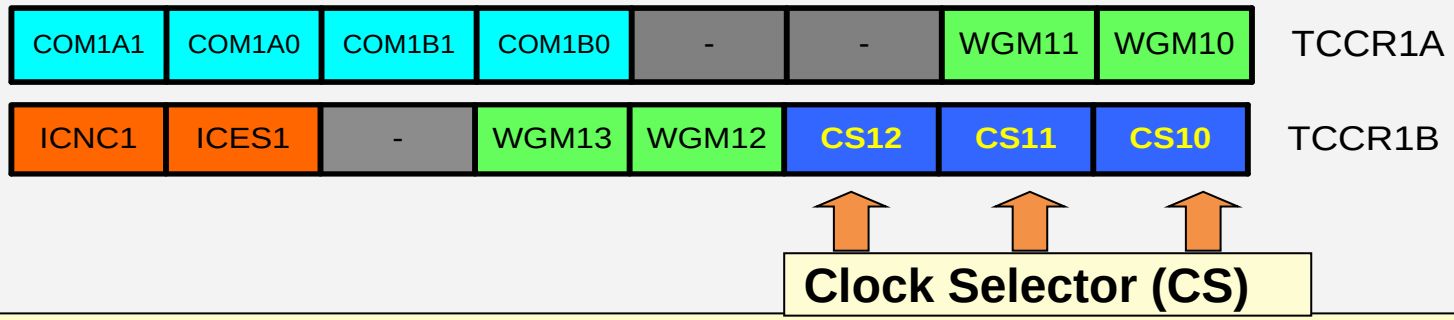


TIFR1

Timer 1

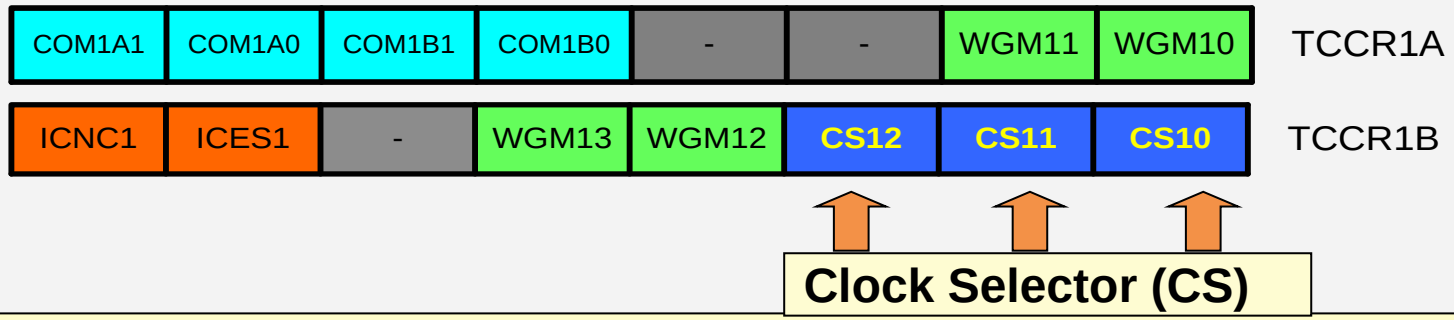

TCCR1A
TCCR1B
TCCR1C
OCR1AH OCR1AL
TCNT1H TCNT1L
OCR1BH OCR1BL
TOV1
=
=
OCF1A
OCF1B
- - ICF1 - - OCF1B OCF1A TOV1
TIFR1

Timer 1



CS12	CS11	CS10	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Timer 1



CS12	CS11	CS10	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Timer 1

COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10

TCCR1A

ICNC1	ICES

CR1B

CS12	CS11	CS1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

g edge

edge

Timer 1

CS12	CS11	CS
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

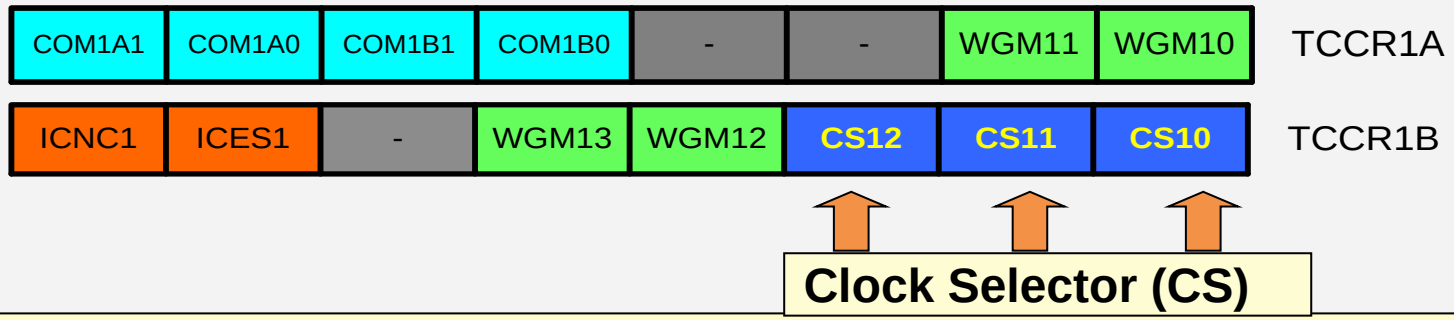
edge
edge

Timer 1

CS12	CS11	CS1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

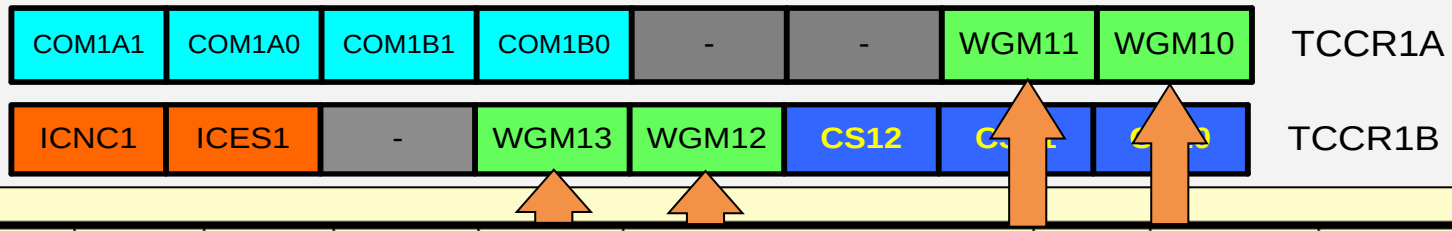
g edge
edge

Timer 1



CS12	CS11	CS10	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Timer 1

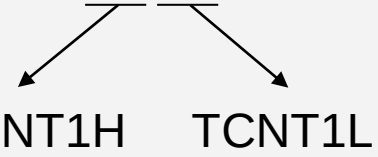


Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Example

Assuming XTAL = 10 MHz write a program that toggles PB5 once per millisecond, using Normal mode.

XTAL = 10 MHz ➔ 1/10 MHz = 0.1 µs
Num. of machine cycles = 1 ms / 0.1 µs = 10,000
TCNT1 = 65,536 – 10,000 = 55,536 = $D8F0



Example

Assum
millise

XTAL
Num.
TCNT

Solution

```
        LDI    R16,HIGH(RAMEND)     ;init stack pointer
        OUT    SPH,R16
        LDI    R16,LOW(RAMEND)
        OUT    SPL,R16
        SBI    DDRB,5               ;PB5 as an output
BEGIN:SBI  PORTB,5                  ;PB5 = 1
        RCALL  DELAY_1ms
        CBI    PORTB,5              ;PB5 = 0
        RCALL  DELAY_1ms
        RJMP   BEGIN

DELAY_1ms:
        LDI    R20,0xD8
        STS    TCNT1H,R20           ;TEMP = 0xD8
        LDI    R20,0xF0
        STS    TCNT1L,R20           ;TCNT1L = 0xF0, TCNT1H = TEMP
        LDI    R20,0x0
        STS    TCCR1A,R20           ;WGM11:10=00
        LDI    R20,0x1
        STS    TCCR1B,R20           ;WGM13:12=00,CS=CLK
AGAIN:SBIS TIFR1,TOV1               ;if TOV1 is set skip next instruction
        RJMP   AGAIN
        LDI    R20,1<<TOV1
        OUT    TIFR1,R20            ;clear TOV1 flag
        LDI    R19,0
        STS    TCCR1B,R19           ;stop timer
        STS    TCCR1A,R19           ;
        RET
```

Example

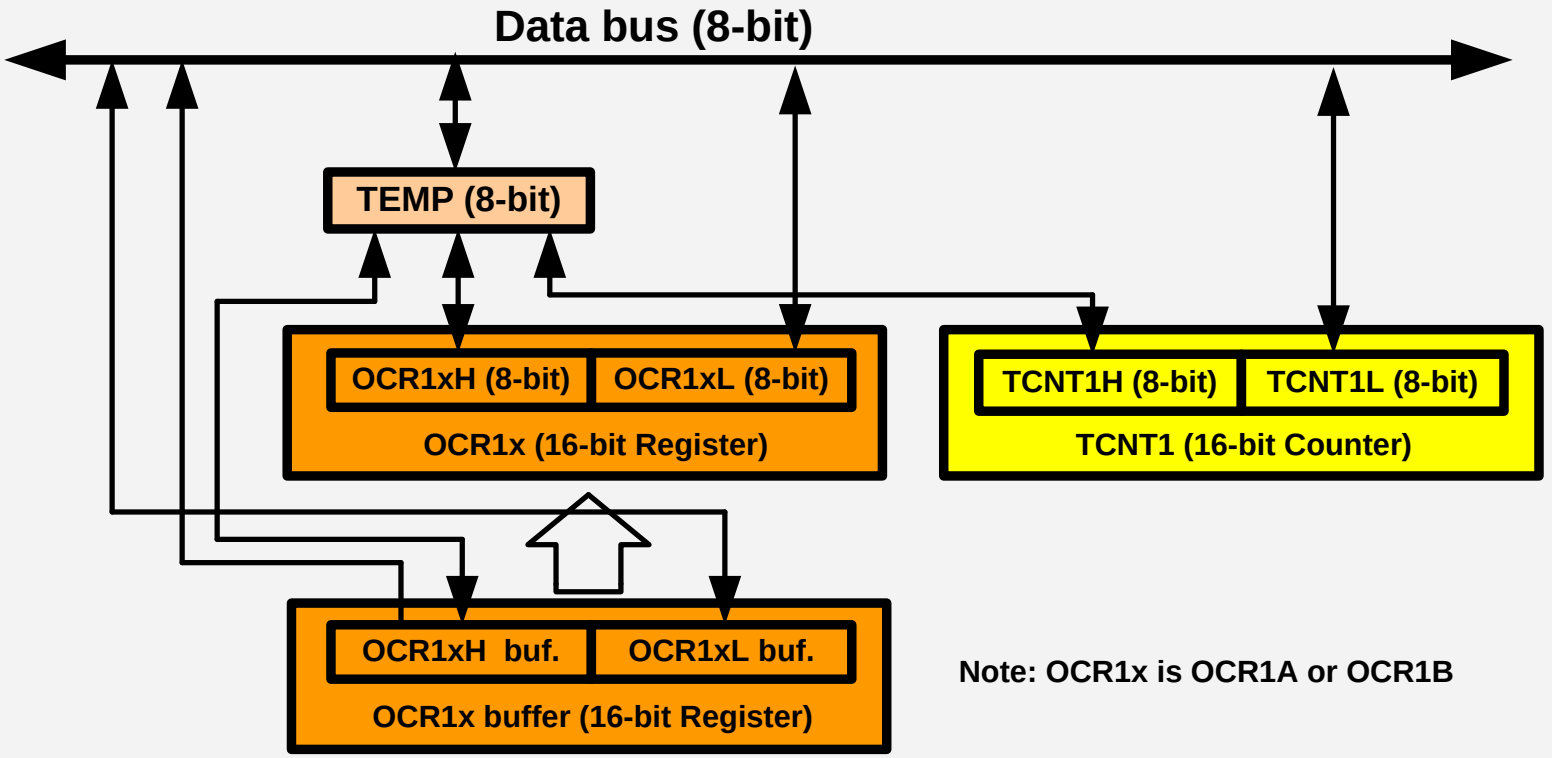
Assum
millise

XTAL
Num.
TCNT

Solution

```
        LDI    R16,HIGH(RAMEND)     ;init stack pointer
        OUT    SPH,R16
        LDI    R16,LOW(RAMEND)
        OUT    SPL,R16
        SBI    DDRB,5               ;PB5 as an output
BEGIN:SBI  PORTB,5                  ;PB5 = 1
        RCALL  DELAY_1ms
        CBI    PORTB,5              ;PB5 = 0
        RCALL  DELAY_1ms
        RJMP   BEGIN

    DELAY_1ms:
        LDI    R20,HIGH(-10000)
        STS    TCNT1H,R20
        LDI    R20, ,LOW(-10000)
        STS    TCNT1L,R20           ;Timer1 overflows after 10000 machine cycles
        LDI    R20,0x0
        STS    TCCR1A,R20           ;WGM11:10=00
        LDI    R20,0x1
        STS    TCCR1B,R20           ;WGM13:12=00,CS=CLK
    AGAIN:SBIS TIFR1,TOV1           ;if TOV1 is set skip next instruction
        RJMP   AGAIN
        LDI    R20,1<<TOV1
        OUT    TIFR1,R20            ;clear TOV1 flag
        LDI    R19,0
        STS    TCCR1B,R19           ;stop timer
        STS    TCCR1A,R19           ;
        RET
```

TEMP register

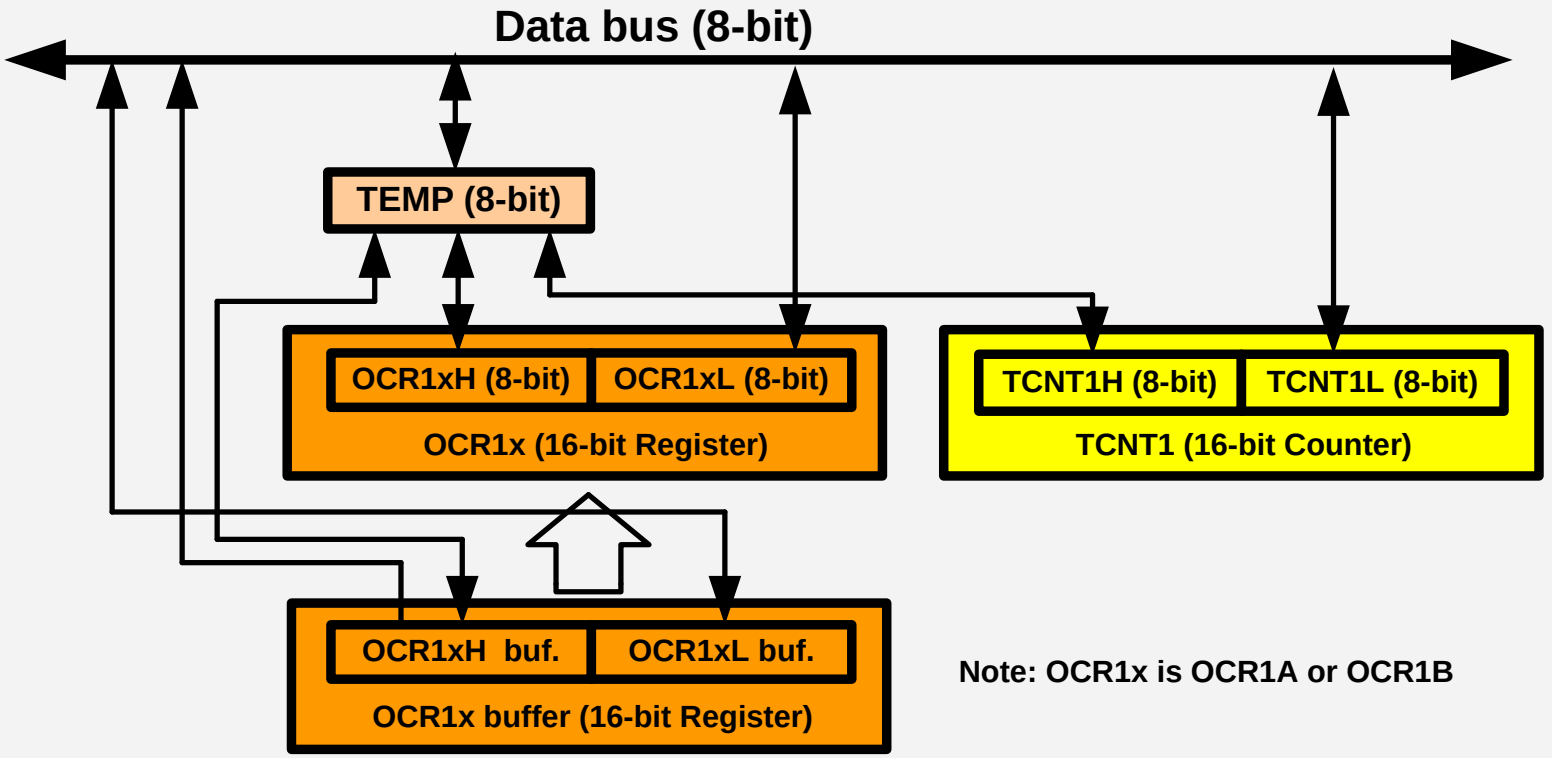


Note: OCR1x is OCR1A or OCR1B

```
LDI   R20,0xF3
STS   TCNT1H,R20
LDI   R20,0x53
STS   TCNT1L,R20
```

```
TCNT1H = 0xF3;
TCNT1L = 0x53;
```

TEMP register

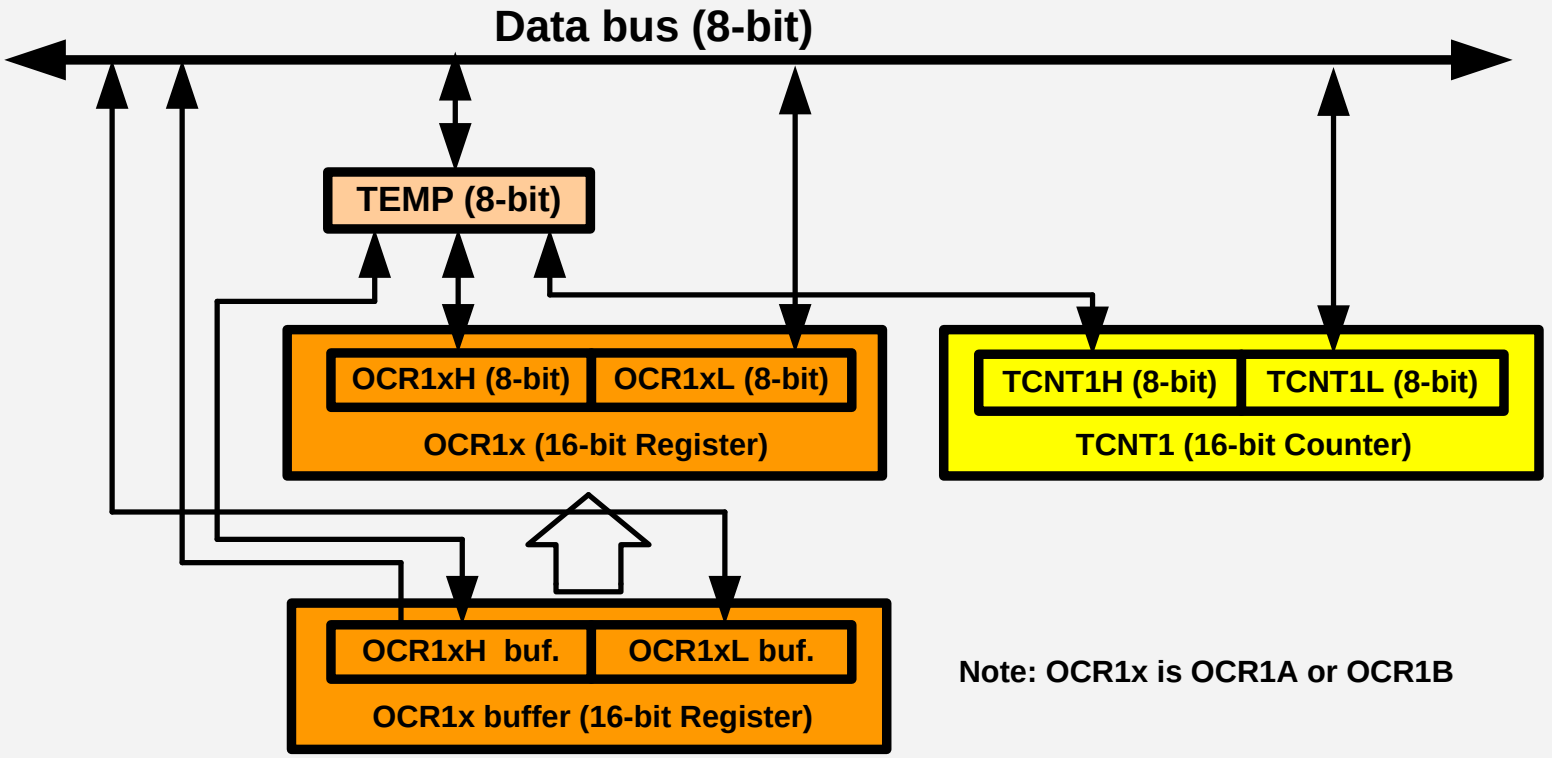


Note: OCR1x is OCR1A or OCR1B

```
LDS   R20,TCNT1L
STS   TCNT1H,R20
LDI   R20,0x53
STS   TCNT1L,R20
```

```
TCNT1H = 0xF3;
TCNT1L = 0x53;
```

TEMP register



Note: OCR1x is OCR1A or OCR1B

```
LDS   R20,TCNT1L
STS   TCNT1H,R20
LDI   R20,0x53
STS   TCNT1L,R20
```

```
TCNT1H=0xF3;
TCNT1L=0x53;
```

Example

Assuming XTAL = 10 MHz, write a program that toggles PB5 once per millisecond, using CTC mode.

Example

...oggles PB5 once per

```
          LDI       R16,HIGH(RAMEND)
          OUT       SPH,R16
          LDI       R16,LOW(RAMEND)
          OUT       SPL,R16

          SBI       DDRB,5     ;PB5 as an output
BEGIN:SBI PORTB,5    ;PB5 = 1
          RCALL     DELAY_1ms
          CBI       PORTB,5    ;PB5 = 0
          RCALL     DELAY_1ms
          RJMP      BEGIN
DELAY_1ms:
          LDI       R20,0x00
          STS       TCNT1H,R20          ;TEMP = 0
          STS       TCNT1L,R20          ;TCNT1L = 0,
TCNT1H = TEMP
          LDI       R20,0x27
          STS       OCR1AH,R20          ;TEMP = 0x27
          LDI       R20,0x0F
          STS       OCR1AL,R20          ;OCR1AL = 0x0F,
OCR1AH = TEMP
          LDI       R20,0x00
          STS       TCCR1A,R20          ;WGM11:10=00
          LDI       R20,0x09
          STS       TCCR1B,R20
          ;WGM13:12=01,CS=CLK
AGAIN:
          SBIS      TIFR1,OCF1A         ;if OCF1A is set
skip next instruction
          RJMP      AGAIN
          LDI       R19,0
          STS       TCCR1B,R19          ;stop timer
          STS       TCCR1A,R19          ;
          LDI       R20,1<<OCF1A
          OUT       TIFR1,R20 ;clear OCF1A flag
          RET
```

Example

...oggles PB5 once per

```
          LDI       R16,HIGH(RAMEND)
          OUT       SPH,R16
          LDI       R16,LOW(RAMEND)
          OUT       SPL,R16

          SBI       DDRB,5     ;PB5 as an output
BEGIN:SBI PORTB,5    ;PB5 = 1
          RCALL     DELAY_1ms
          CBI       PORTB,5    ;PB5 = 0
          RCALL     DELAY_1ms
          RJMP      BEGIN
DELAY_1ms:
          LDI       R20,0x00
          STS       TCNT1H,R20          ;TEMP = 0
          STS       TCNT1L,R20          ;TCNT1L = 0,
TCNT1H = TEMP
          LDI       R20,0x27
          STS       OCR1AH,R20          ;TEMP = 0x27
          LDI       R20,0x0F
          STS       OCR1AL,R20          ;OCR1AL = 0x0F,
OCR1AH = TEMP
          LDI       R20,0x00
          STS       TCCR1A,R20          ;WGM11:10=00
          LDI       R20,0x09
          STS       TCCR1B,R20
          ;WGM13:12=01,CS=CLK
AGAIN:
          SBIS      TIFR1,OCF1A         ;if OCF1A is set
skip next instruction
          RJMP      AGAIN
          LDI       R19,0
          STS       TCCR1B,R19          ;stop timer
          STS       TCCR1A,R19          ;
          LDI       R20,1<<OCF1A
          OUT       TIFR1,R20 ;clear OCF1A flag
          RET
```

Example

oggles PB5 once per

```
          LDI       R16,HIGH(RAMEND)
          OUT       SPH,R16
          LDI       R16,LOW(RAMEND)
          OUT       SPL,R16

          SBI       DDRB,5     ;PB5 as an output
BEGIN:SBI PORTB,5    ;PB5 = 1
          RCALL     DELAY_1ms
          CBI       PORTB,5    ;PB5 = 0
          RCALL     DELAY_1ms
          RJMP      BEGIN
DELAY_1ms:
          LDI       R20,0x00
          STS       TCNT1H,R20          ;TEMP = 0
          STS       TCNT1L,R20          ;TCNT1L = 0,
TCNT1H = TEMP
          LDI       R20,0x27
          STS       OCR1AH,R20          ;TEMP = 0x27
          LDI       R20,0x0F
          STS       OCR1AL,R20          ;OCR1AL = 0x0F,
OCR1AH = TEMP
          LDI       R20,0x00
          STS       TCCR1A,R20          ;WGM11:10=00
          LDI       R20,0x09
          STS       TCCR1B,R20
          ;WGM13:12=01,CS=CLK
AGAIN:
          SBIS      TIFR1,OCF1A         ;if OCF1A is set
skip next instruction
          RJMP      AGAIN
          LDI       R19,0
          STS       TCCR1B,R19          ;stop timer
          STS       TCCR1A,R19          ;
          LDI       R20,1<<OCF1A
          OUT       TIFR1,R20 ;clear OCF1A flag
          RET
```

Example

...oggles PB5 once per

Assembly

```
          LDI       R16,HIGH(RAMEND)
          OUT       SPH,R16
          LDI       R16,LOW(RAMEND)
          OUT       SPL,R16

          SBI       DDRB,5     ;PB5 as an output
BEGIN:SBI PORTB,5    ;PB5 = 1
          RCALL     DELAY_1ms
          CBI       PORTB,5    ;PB5 = 0
          RCALL     DELAY_1ms
          RJMP      BEGIN
DELAY_1ms:
          LDI       R20,0x00
          STS       TCNT1H,R20          ;TEMP = 0
          STS       TCNT1L,R20          ;TCNT1L = 0,
TCNT1H = TEMP
          LDI       R20,0x27
          STS       OCR1AH,R20          ;TEMP = 0x27
          LDI       R20,0x0F
          STS       OCR1AL,R20          ;OCR1AL = 0x0F,
OCR1AH = TEMP
          LDI       R20,0x00
          STS       TCCR1A,R20          ;WGM11:10=00
          LDI       R20,0x09
          STS       TCCR1B,R20
          ;WGM13:12=01,CS=CLK
AGAIN:
          SBIS      TIFR1,OCF1A         ;if OCF1A is set
skip next instruction
          RJMP      AGAIN
          LDI       R19,0
          STS       TCCR1B,R19          ;stop timer
          STS       TCCR1A,R19          ;
          LDI       R20,1<<OCF1A
          OUT       TIFR1,R20 ;clear OCF1A flag
          RET
```

```
#include <avr/io.h>
void  delay1ms();
int main(){
 DDRB |= 1<<5;
 while (1) {
 delay1ms();
 PORTB ^= (1<<5); //toggle PB5
 }
}
void  delay1ms()
{
 TCNT1 = 0;
 OCR1A = 10000-1;
 TCCR1A = 0;     //WGM=0100 (CTC)
 TCCR1B = 0x09;  //N = 1
 while((TIFR1&(1<<OCF1A))==0)
 { } //wait until OCF1A is set
 TCCR1B = 0; //stop timer1
 TIFR1 = 1<<OCF1A;//clear flag
}
```

Example

...oggles PB5 once per

Assembly

```
          LDI       R16,HIGH(RAMEND)
          OUT       SPH,R16
          LDI       R16,LOW(RAMEND)
          OUT       SPL,R16

          SBI       DDRB,5     ;PB5 as an output
BEGIN:SBI PORTB,5    ;PB5 = 1
          RCALL     DELAY_1ms
          CBI       PORTB,5    ;PB5 = 0
          RCALL     DELAY_1ms
          RJMP      BEGIN
DELAY_1ms:
          LDI       R20,0x00
          STS       TCNT1H,R20          ;TEMP = 0
          STS       TCNT1L,R20          ;TCNT1L = 0,
TCNT1H = TEMP
          LDI       R20,0x27
          STS       OCR1AH,R20          ;TEMP = 0x27
          LDI       R20,0x0F
          STS       OCR1AL,R20          ;OCR1AL = 0x0F,
OCR1AH = TEMP
          LDI       R20,0x00
          STS       TCCR1A,R20          ;WGM11:10=00
          LDI       R20,0x09
          STS       TCCR1B,R20
          ;WGM13:12=01,CS=CLK
AGAIN:
          SBIS      TIFR1,OCF1A         ;if OCF1A is set
skip next instruction
          RJMP      AGAIN
          LDI       R19,0
          STS       TCCR1B,R19          ;stop timer
          STS       TCCR1A,R19          ;
          LDI       R20,1<<OCF1A
          OUT       TIFR1,R20 ;clear OCF1A flag
          RET
```

```
#include <avr/io.h>
void  delay1ms();
int main(){
 DDRB |= 1<<5;
 while (1) {
 delay1ms();
 PORTB ^= (1<<5); //toggle PB5
 }
}
void  delay1ms()
{
 TCNT1 = 0;
 OCR1A = 10000-1;
 TCCR1A = 0;     //WGM=0100 (CTC)
 TCCR1B = 0x09;  //N = 1
 while((TIFR1&(1<<OCF1A))==0)
 { } //wait until OCF1A is set
 TCCR1B = 0; //stop timer1
 TIFR1 = 1<<OCF1A;//clear flag
}
```

Example

...oggles PB5 once per

Assembly

```
          LDI       R16,HIGH(RAMEND)
          OUT       SPH,R16
          LDI       R16,LOW(RAMEND)
          OUT       SPL,R16

          SBI       DDRB,5     ;PB5 as an output
BEGIN:SBI PORTB,5    ;PB5 = 1
          RCALL     DELAY_1ms
          CBI       PORTB,5    ;PB5 = 0
          RCALL     DELAY_1ms
          RJMP      BEGIN
DELAY_1ms:
          LDI       R20,0x00
          STS       TCNT1H,R20          ;TEMP = 0
          STS       TCNT1L,R20          ;TCNT1L = 0,
TCNT1H = TEMP
          LDI       R20,0x27
          STS       OCR1AH,R20          ;TEMP = 0x27
          LDI       R20,0x0F
          STS       OCR1AL,R20          ;OCR1AL = 0x0F,
OCR1AH = TEMP
          LDI       R20,0x00
          STS       TCCR1A,R20          ;WGM11:10=00
          LDI       R20,0x09
          STS       TCCR1B,R20
          ;WGM13:12=01,CS=CLK
AGAIN:
          SBIS      TIFR1,OCF1A         ;if OCF1A is set
skip next instruction
          RJMP      AGAIN
          LDI       R19,0
          STS       TCCR1B,R19          ;stop timer
          STS       TCCR1A,R19          ;
          LDI       R20,1<<OCF1A
          OUT       TIFR1,R20 ;clear OCF1A flag
          RET
```

C

```
#include <avr/io.h>
void  delay1ms();
int main(){
 DDRB |= 1<<5;
 while (1) {
 delay1ms();
 PORTB ^= (1<<5); //toggle PB5
 }
}
void  delay1ms()
{
 TCNT1 = 0;
 OCR1A = 10000-1;
 TCCR1A = 0;     //WGM=0100 (CTC)
 TCCR1B = 0x09;  //N = 1
 while((TIFR1&(1<<OCF1A))==0)
 { } //wait until OCF1A is set
 TCCR1B = 0; //stop timer1
 TIFR1 = 1<<OCF1A;//clear flag
}
```

Example

...oggles PB5 once per

Assembly

```
          LDI       R16,HIGH(RAMEND)
          OUT       SPH,R16
          LDI       R16,LOW(RAMEND)
          OUT       SPL,R16

          SBI       DDRB,5     ;PB5 as an output
BEGIN:SBI PORTB,5    ;PB5 = 1
          RCALL     DELAY_1ms
          CBI       PORTB,5    ;PB5 = 0
          RCALL     DELAY_1ms
          RJMP      BEGIN
DELAY_1ms:
          LDI       R20,0x00
          STS       TCNT1H,R20          ;TEMP = 0
          STS       TCNT1L,R20          ;TCNT1L = 0,
TCNT1H = TEMP
          LDI       R20,0x27
          STS       OCR1AH,R20          ;TEMP = 0x27
          LDI       R20,0x0F
          STS       OCR1AL,R20          ;OCR1AL = 0x0F,
OCR1AH = TEMP
          LDI       R20,0x00
          STS       TCCR1A,R20          ;WGM11:10=00
          LDI       R20,0x09
          STS       TCCR1B,R20
          ;WGM13:12=01,CS=CLK
AGAIN:
          SBIS      TIFR1,OCF1A         ;if OCF1A is set
skip next instruction
          RJMP      AGAIN
          LDI       R19,0
          STS       TCCR1B,R19          ;stop timer
          STS       TCCR1A,R19          ;
          LDI       R20,1<<OCF1A
          OUT       TIFR1,R20 ;clear OCF1A flag
          RET
```

C

```
#include <avr/io.h>
void  delay1ms();
int main(){
 DDRB |= 1<<5;
 while (1) {
 delay1ms();
 PORTB ^= (1<<5); //toggle PB5
 }
}
void  delay1ms()
{
 TCNT1 = 0;
 OCR1A = 10000-1;
 TCCR1A = 0;     //WGM=0100 (CTC)
 TCCR1B = 0x09;  //N = 1
 while((TIFR1&(1<<OCF1A))==0)
 { } //wait until OCF1A is set
 TCCR1B = 0; //stop timer1
 TIFR1 = 1<<OCF1A;//clear flag
}
```

Example

...oggles PB5 once per

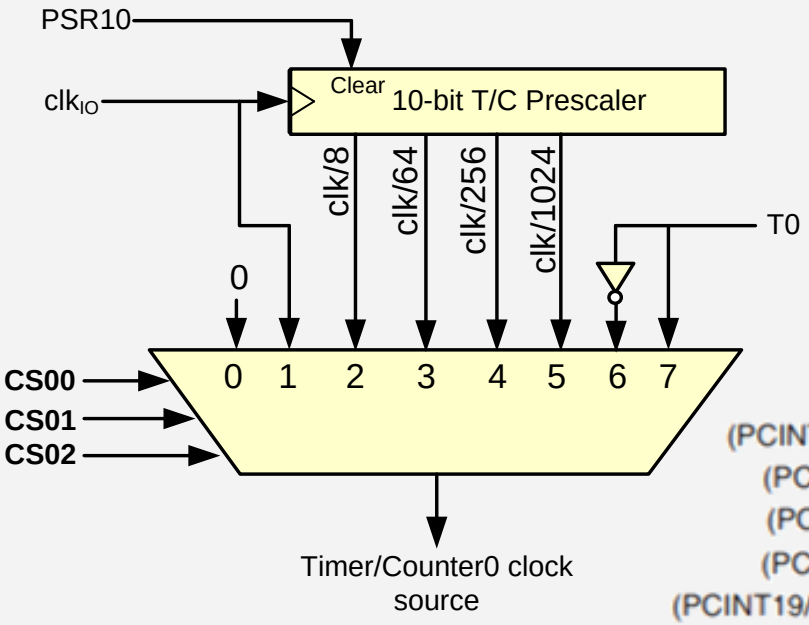
Assembly

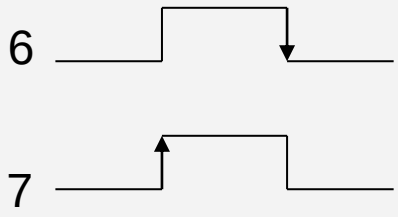
```
          LDI       R16,HIGH(RAMEND)
          OUT       SPH,R16
          LDI       R16,LOW(RAMEND)
          OUT       SPL,R16

          SBI       DDRB,5     ;PB5 as an output
BEGIN:SBI PORTB,5    ;PB5 = 1
          RCALL     DELAY_1ms
          CBI       PORTB,5    ;PB5 = 0
          RCALL     DELAY_1ms
          RJMP      BEGIN
DELAY_1ms:
          LDI       R20,0x00
          STS       TCNT1H,R20          ;TEMP = 0
          STS       TCNT1L,R20          ;TCNT1L = 0,
TCNT1H = TEMP
          LDI       R20,0x27
          STS       OCR1AH,R20          ;TEMP = 0x27
          LDI       R20,0x0F
          STS       OCR1AL,R20          ;OCR1AL = 0x0F,
OCR1AH = TEMP
          LDI       R20,0x00
          STS       TCCR1A,R20          ;WGM11:10=00
          LDI       R20,0x09
          STS       TCCR1B,R20
          ;WGM13:12=01,CS=CLK
AGAIN:
          SBIS      TIFR1,OCF1A         ;if OCF1A is set
skip next instruction
          RJMP      AGAIN
          LDI       R19,0
          STS       TCCR1B,R19          ;stop timer
          STS       TCCR1A,R19          ;
          LDI       R20,1<<OCF1A
          OUT       TIFR1,R20 ;clear OCF1A flag
          RET
```

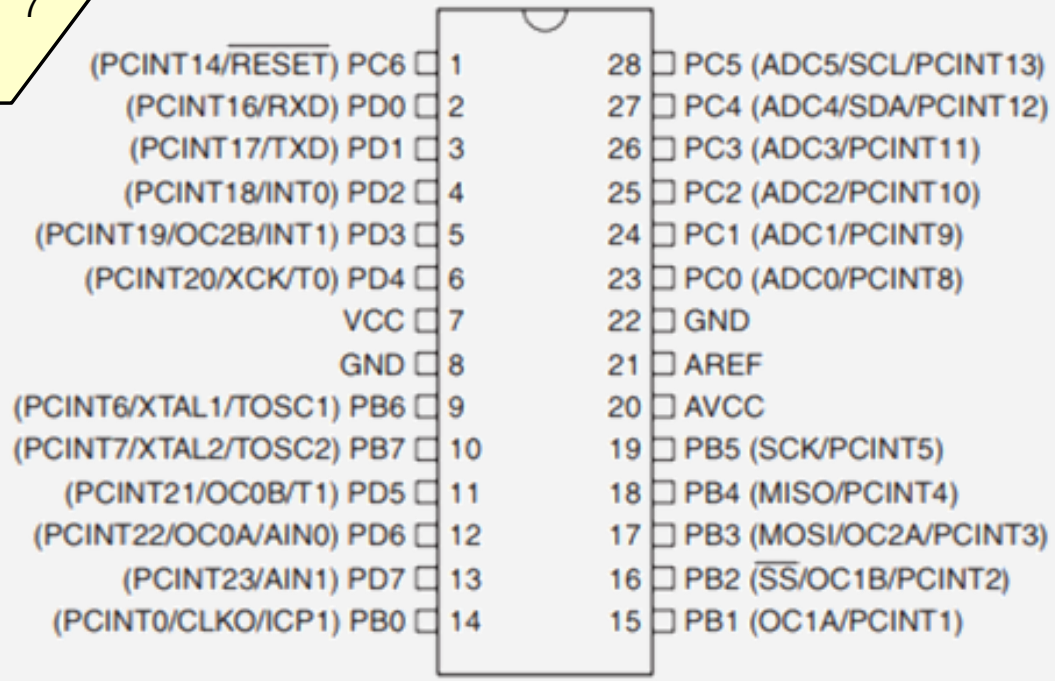
C

```
#include <avr/io.h>
void  delay1ms();
int main(){
 DDRB |= 1<<5;
 while (1) {
 delay1ms();
 PORTB ^= (1<<5); //toggle PB5
 }
}
void  delay1ms()
{
 TCNT1 = 0;
 OCR1A = 10000-1;
 TCCR1A = 0;     //WGM=0100 (CTC)
 TCCR1B = 0x09;  //N = 1
 while((TIFR1&(1<<OCF1A))==0)
 { } //wait until OCF1A is set
 TCCR1B = 0; //stop timer1
 TIFR1 = 1<<OCF1A;//clear flag
}
```

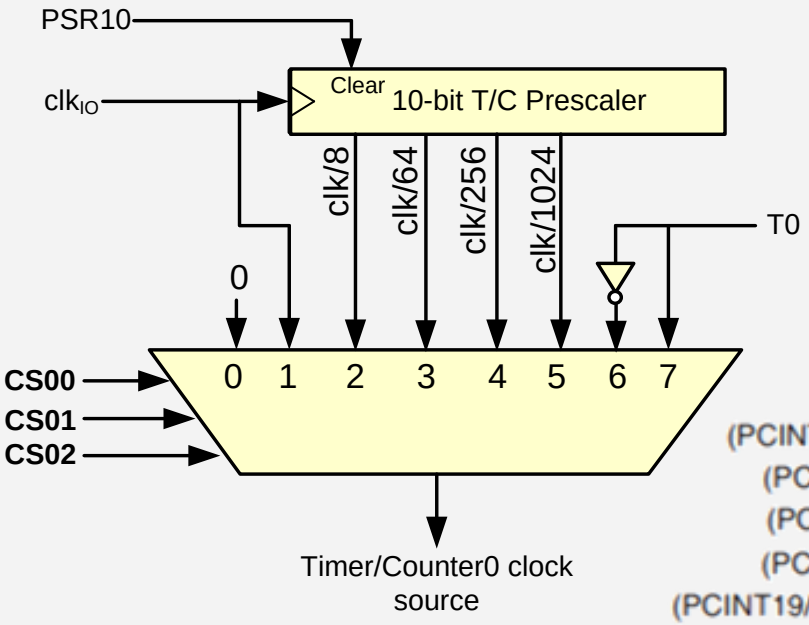
Counting

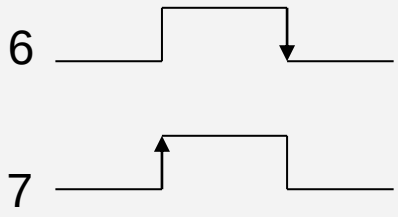

PSR10
clk$_{IO}$
Clear
10-bit T/C Prescaler
clk/8
clk/64
clk/256
clk/1024
T0
0
0 1 2 3 4 5 6 7
CS00
CS01
CS02
Timer/Counter0 clock source

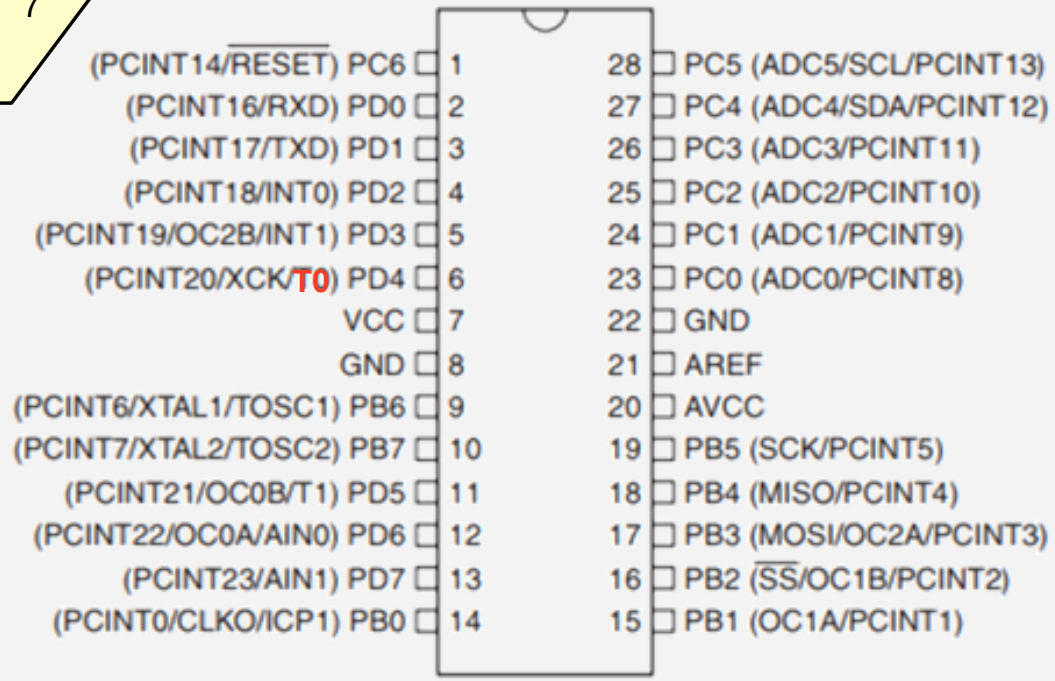

6
7


(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

Counting


PSR10
clk_IO
Clear
10-bit T/C Prescaler
clk/8
clk/64
clk/256
clk/1024
T0
0
0 1 2 3 4 5 6 7
CS00
CS01
CS02
Timer/Counter0 clock source


6
7


(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

Example 2

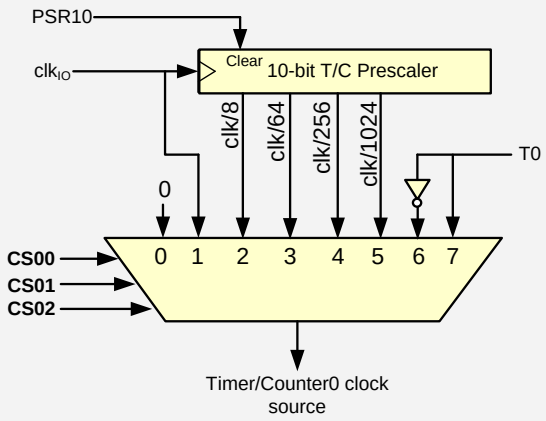
Assuming that clock pulses are fed into pin T0, write a program for counter 0 in normal mode to count the pulses on falling edge and display the state of the TCNT0 count on PORTC.

Assembly

```
        CBI     DDRD,4      ;make T0 (PD4) input
        LDI     R20,0xFF
        OUT     DDRC,R20    ;make PORTC output
        LDI     R20,0x00
        OUT     TCCR0A,R20
        LDI     R20,0x06
        OUT     TCCR0B,R20  ;counter, falling edge
AGAIN:
        IN      R20,TCNT0
        OUT     PORTC,R20   ;PORTC = TCNT0
        RJMP    AGAIN               ;keep doing it
```

C

```
#include <avr/io.h>
int main()
{
 DDRD &= ~(1<<4);
 DDRC = 0xFF;
 TCCR0A = 0;     //WGM=0000
                 (Normal)
 TCCR0B = 0x06; //CS=6
                 (Count on fall)
 while (1)
 {
   PORTC = TCNT0; //read TCNT0
 }
}
```



COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	TCCR0A
FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	TCCR0B

Example 2

Assuming that clock pulses are fed into pin T1. Write a program for Counter1 in CTC mode to make PORTC.0 high every 100 pulses.

Assembly

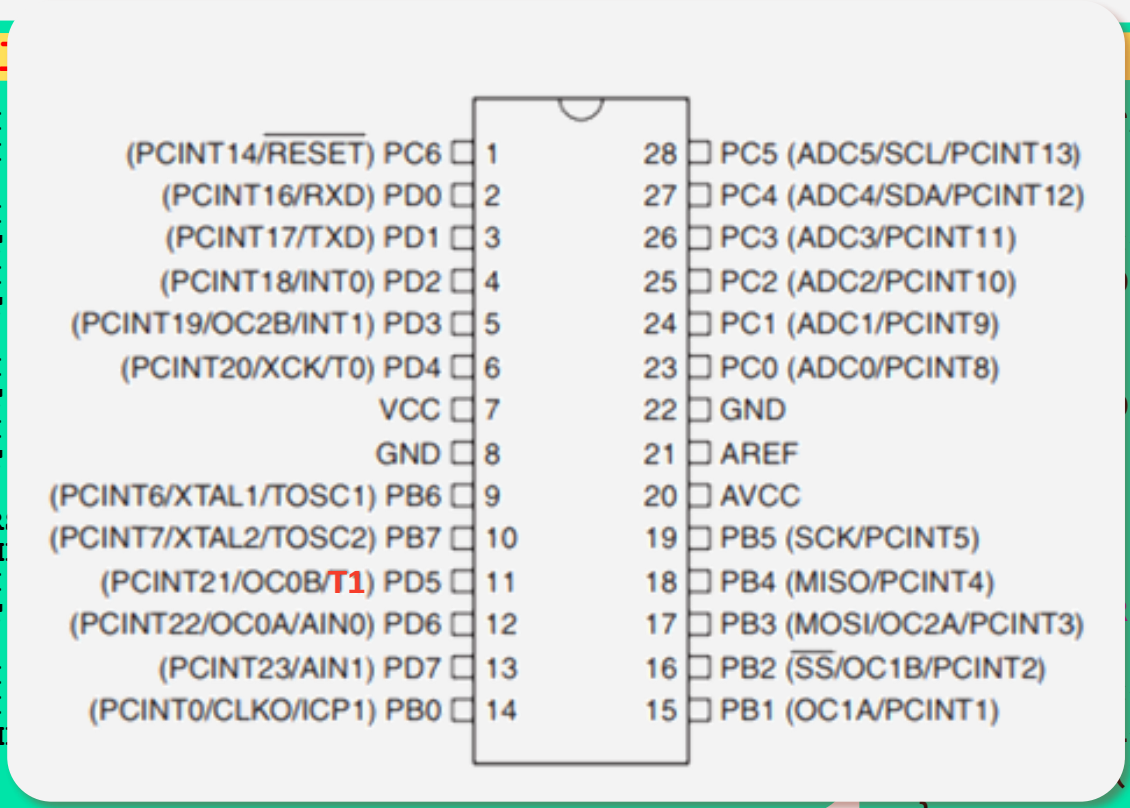
```
        CBI     DDRD,5      ;make T1 (PD5) input
        SBI     DDRC,0      ;PC0 as an output

        LDI     R20,0x00
        OUT     TCCR1A,R20
        LDI     R20,0x0E    ;CS=6 (falling edge counter)
        OUT     TCCR1B,R20  ;CTC
AGAIN:
        LDI     R20,0
        OUT     OCR1AH,R20    ;TEMP = 0
        LDI     R20,99
        OUT     OCR1AL,R20    ;ORC1L = R20, OCR1H = TEMP
L1:
        SBRS    TIFR1,OCF1A
        RJMP    L1              ;keep doing it
        LDI     R20,1<<OCF1A    ;clear OCF1A flag
        OUT     TIFR1, R20

        SBI     PORTC,0             ;PC0 = 1
        CBI     PORTC,0             ;PC0 = 0
        RJMP    AGAIN               ;keep doing it
```

C

```
#include <avr/io.h>
int main()
{
 DDRD &= ~(1<<5);
 DDRC |= 1<<0;
 TCCR1A = 0;     //WGM=0100
                 (CTC)
 TCCR1B = 0x0E;  //CS=6 (Count
                 on fall)
 OCR1A = 99;
 while (1)
 {
  while((TIFR1&(1<<OCF1A)) ==
0);
  TIFR1 = (1<<OCF1A);
  PORTC |= (1<<0);  //PC0 = 1
  PORTC &= ~(1<<0); //PC0 = 0
 }
}
```

Example 2

Assuming that clock pulses are fed into pin T1. Write a program for Counter1 in CTC mode to make PORTC.0 high every 100 pulses.



Assembl

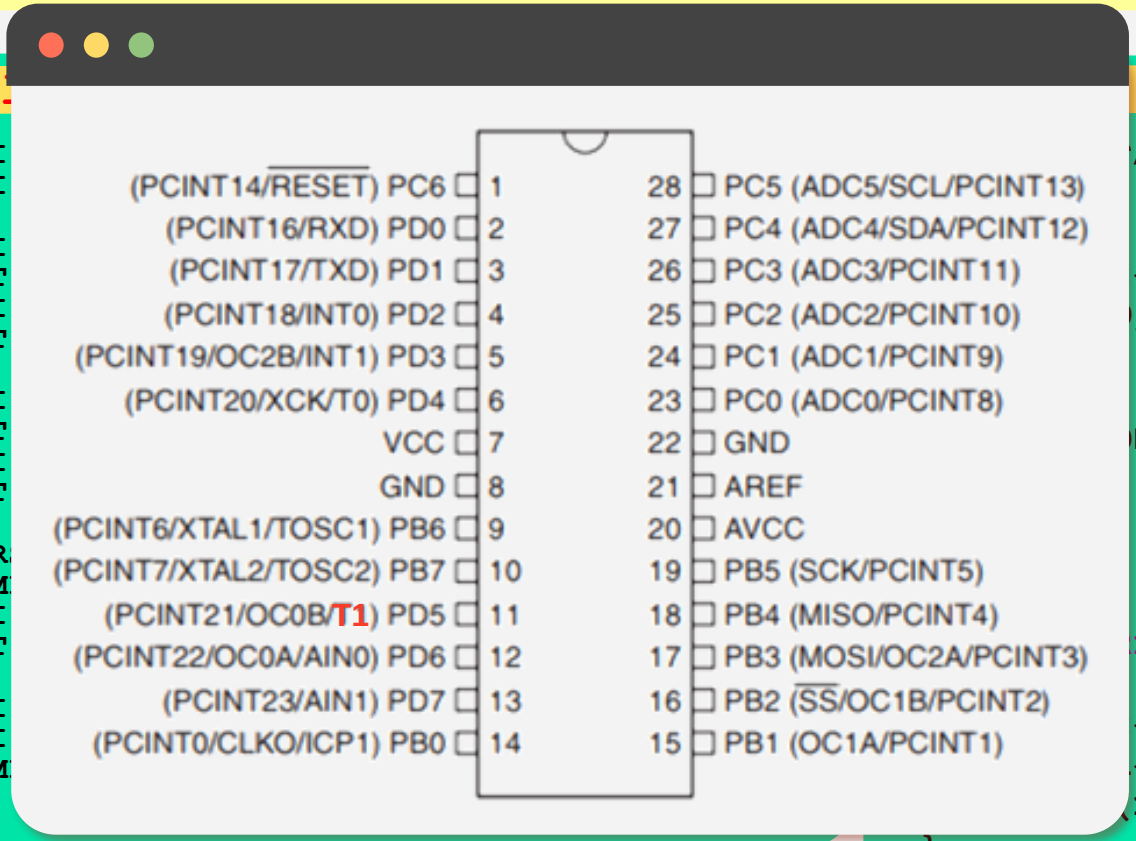
```
        CBI
        SBI

        LDI
        OUT
        LDI
        OUT
AGAIN:
        LDI
        OUT
        LDI
        OUT
L1:
        SBR
        RJM
        LDI
        OUT

        SBI
        CBI
        RJM
```

```
/io.h>


<5);
;
    //WGM=0100
    (CTC)
E;  //CS=6 (Count
    on fall)



1&(1<<OCF1A)) ==

<OCF1A);
<<0);  //PC0 = 1
1<<0); //PC0 = 0
 }
}
```

Timer 0 | Timer 1 | Timer 2

Example 2

Assuming that clock pulses are fed into pin T1. Write a program for Counter1 in CTC mode to make PORTC.0 high every 100 pulses.



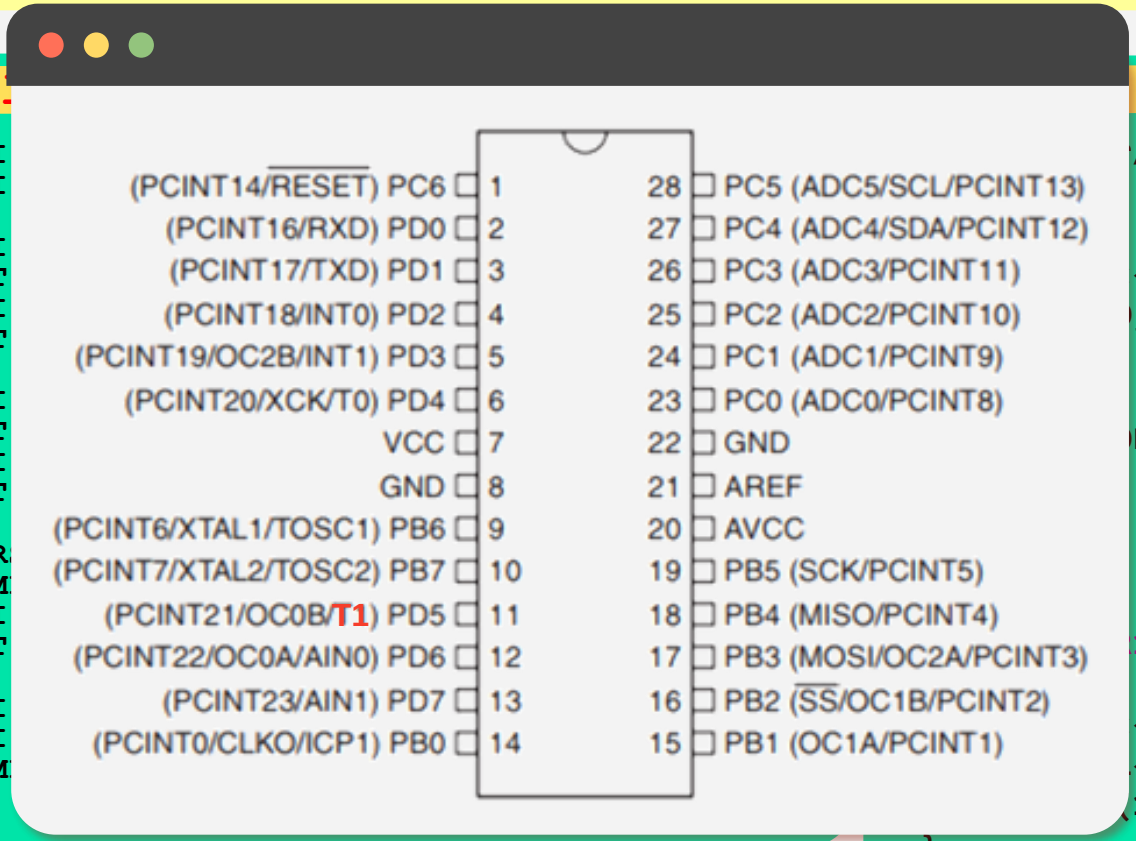
Assemb

```
        CBI
        SBI

        LDI
        OUT
        LDI
        OUT
AGAIN:
        LDI
        OUT
        LDI
        OUT
L1:
        SBR
        RJM
        LDI
        OUT

        SBI
        CBI
        RJM
```

```
/io.h>


<5);
;
    //WGM=0100
    (CTC)
E;  //CS=6 (Count
    on fall)



1&(1<<OCF1A)) ==

<OCF1A);
<<0);  //PC0 = 1
1<<0); //PC0 = 0
 }
}
```

Example 2

Assuming that clock pulses are fed into pin T1. Write a program for Counter1 in CTC mode to make PORTC.0 high every 100 pulses.



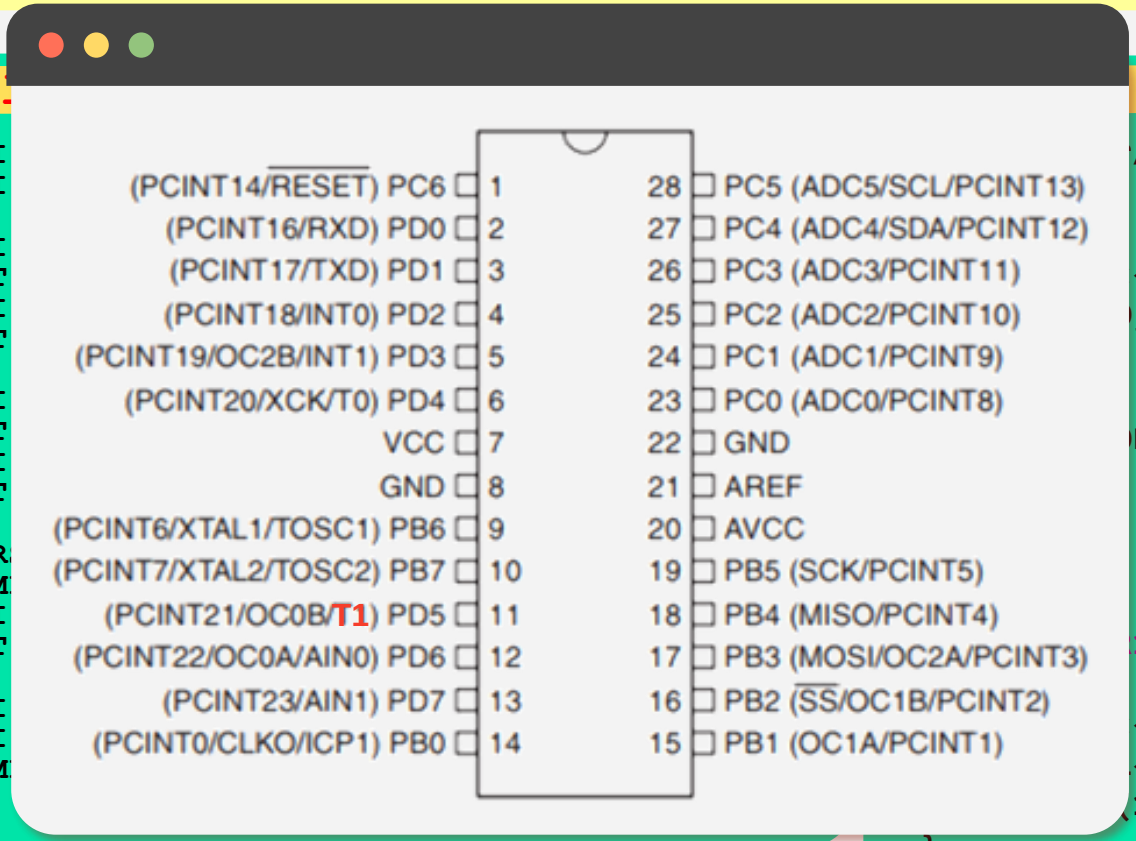
Assemb

```
        CBI
        SBI

        LDI
        OUT
        LDI
        OUT
AGAIN:
        LDI
        OUT
        LDI
        OUT
L1:
        SBR
        RJM
        LDI
        OUT

        SBI
        CBI
        RJM
```

```
/io.h>


<5);
;
    //WGM=0100
    (CTC)
E;  //CS=6 (Count
    on fall)



1&(1<<OCF1A)) ==

<OCF1A);
<<0);  //PC0 = 1
1<<0); //PC0 = 0
 }
}
```

Example 2

Assuming that clock pulses are fed into pin T1. Write a program for Counter1 in CTC mode to make PORTC.0 high every 100 pulses.



Assemb

```
        CBI
        SBI

        LDI
        OUT
        LDI
        OUT
AGAIN:
        LDI
        OUT
        LDI
        OUT
L1:
        SBR
        RJM
        LDI
        OUT

        SBI
        CBI
        RJM
```

```
/io.h>


<5);
;
    //WGM=0100
    (CTC)
E;  //CS=6 (Count
    on fall)



1&(1<<OCF1A)) ==

<OCF1A);
<<0);  //PC0 = 1
1<<0); //PC0 = 0
 }
}
```

Thanks!

Do you have any questions?

razeghizade@gmail.com

www.pudica.ir

CREADITS: This presentation was created by M.Razeghizadeh
Please keep this slide for attribution