Interrupts

- An interrupts is an external or internal event that interrupts the microcontroller to inform that a device needs its service.

interrupt vs polling

- micro controller can serve many devices. there are two ways to do that interrupt or polling
- In polling the micro controller continuously monitors the status of a given device ,when the condition is met it performs specific service.after that it moves to the next device until every one is serviced.
- in interrupt whenever any device needs its service the device notifies by sending an interrupt.

Advantages of interrupt.

- We can assign priority in case of interrupts.
- In interrupt micro controller can also ignore (mask) a device request
- Polling method waste time of micro controller

Sequence of event when interrupt happens.

- Current instruction execution is completed
- The address of next instruction is stored on stack
- Address of ISR is loaded into PC
- The processor executes the ISR
- The ISR execution is indicated by RETI instruction.
- The processor loads the PC with the value stored on the stack

note>>

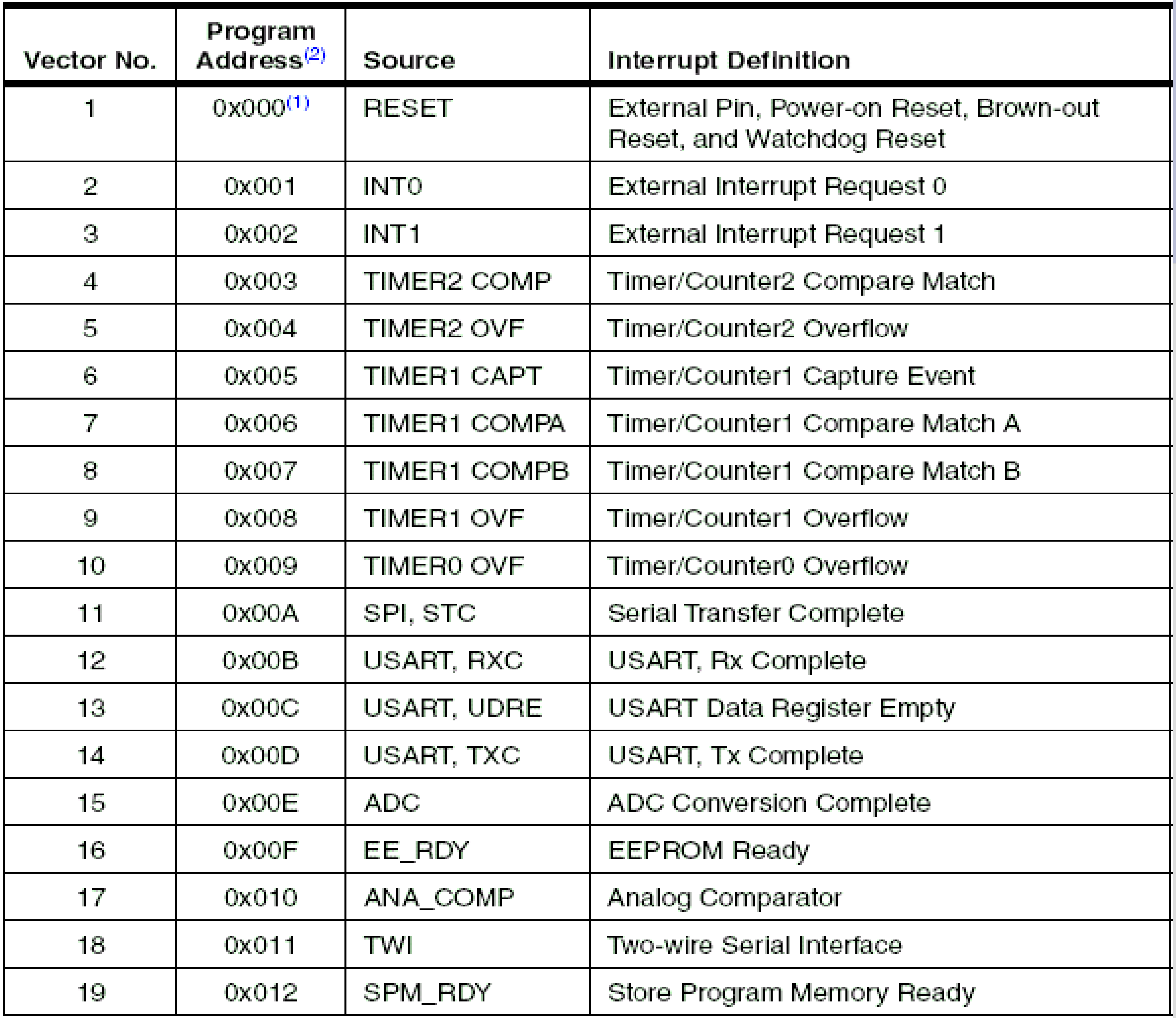
- Interrupt can occur at any time so status register(SREG) needs to be saved.
- One method :use register

- Second :use stack
- Note>>> normally after the interrupt occurs & is being serviced , global interrupt is disabled

How the priorities are assigned?

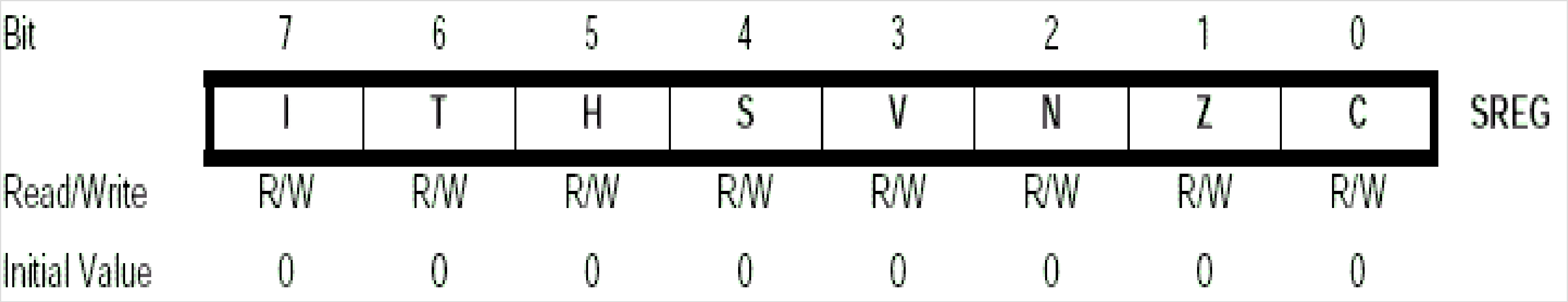
- These are fixed for avr
- The priorities of interrupt is determined by the way interrupts vectors are assigned.
- An interrupt vector at lower memory address is having a higher priority

```
; PROGRAM FOR 2 FREQUENCY GENERATION BY PRESSING KEYS
;*************************************************************************************
        .INCLUDE "../8515def.inc"       ;
        .TEXT
        .ORG 0X00
        .GLOBAL _START
;**************************************************************************************
        _START:
        RJMP RESET_HANDLER              ;
        RJMP EXT_INT0_HANDLER           ;
        RJMP EXT_INT1_HANDLER           ;
        RJMP TIM_CAPT_HANDLER           ;
        RJMP TIM1_COMA_HANDLER          ;
        RJMP TIM1_COMB_HANDLER          ;
        RJMP TIM1_OVF_HANDLER           ;
        RJMP TIM0_OVF_HANDLER           ;
        RJMP SPI_STC_HANDLER            ;
        RJMP UART_RXCINT0_HANDLER       ;
        RJMP UART_DRE_HANDLER           ;
        RJMP UART_TXC_HANDLER           ;
        RJMP ANA_COMP_HANDLER           ;
;******************************************************************************************
        RESET_HANDLER           :       rjmp main
        EXT_INT0_HANDLER        :       reti
        EXT_INT1_HANDLER        :       reti
        TIM_CAPT_HANDLER        :       reti
        TIM1_COMA_HANDLER       :       reti
        TIM1_COMB_HANDLER       :       reti
        TIM1_OVF_HANDLER        :       reti
```

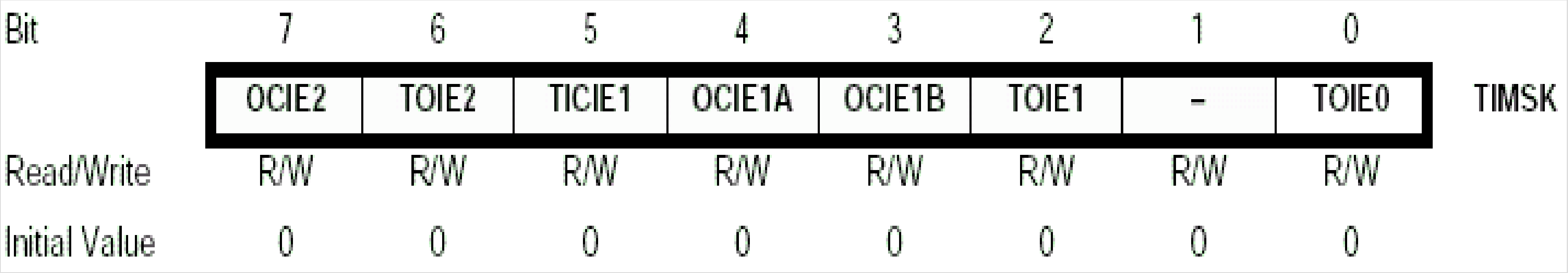


Vector No.	Program Address[2]	Source	Interrupt Definition
1	0x000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	SPM_RDY	Store Program Memory Ready

Various interrupt registers

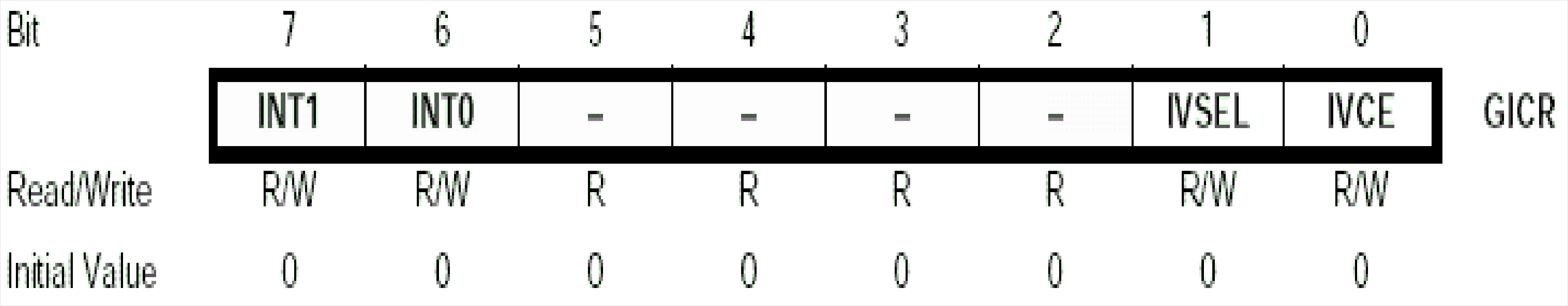


Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	



Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registers



Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timers

•Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes

•One 16 bit timers with seperate prescaler ,compare mode and capture mode

Purpose

- To calculate time delays

- To be use as the counters
- Its advantage is that the input clock and operationof the timer is independent of the program execution.

Why different timers?

- The answer is simple to have right timer for the right application

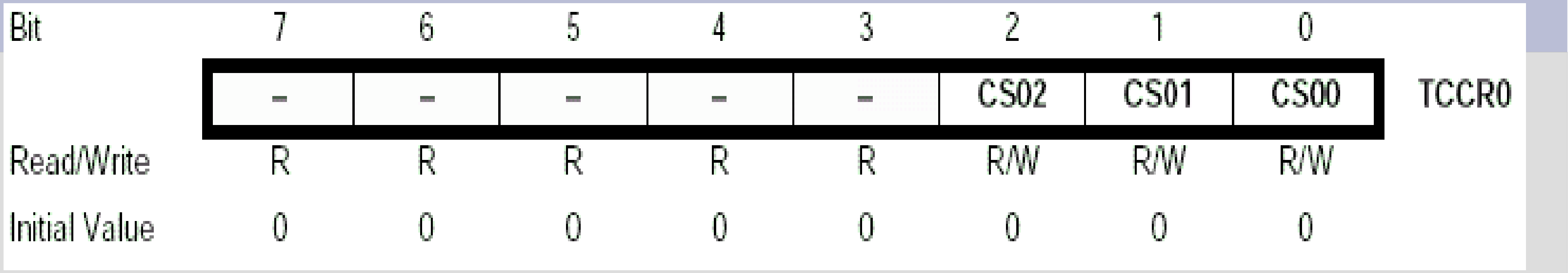
- A timer with 16-bit resolution is certainly more flexible to use than one with 8-bit resolution.

- for many application it is sufficient to have 8 bit resolution using higher resolution means higher program load & higher cost.

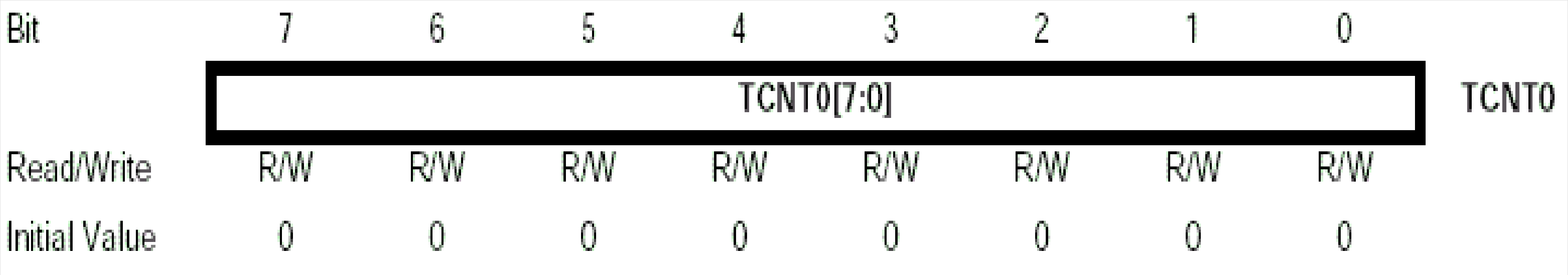
Different modes of operation

- Normal mode
- Capture mode
- Pwm mode

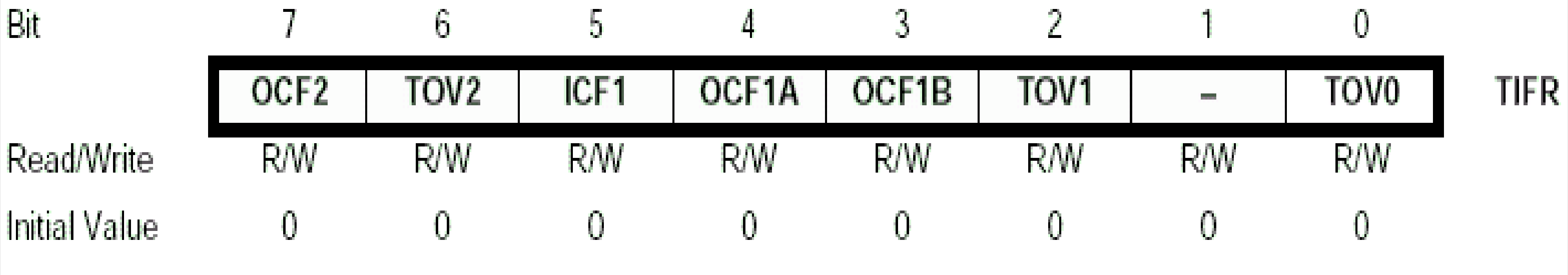
Registers



Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	



Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	



Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

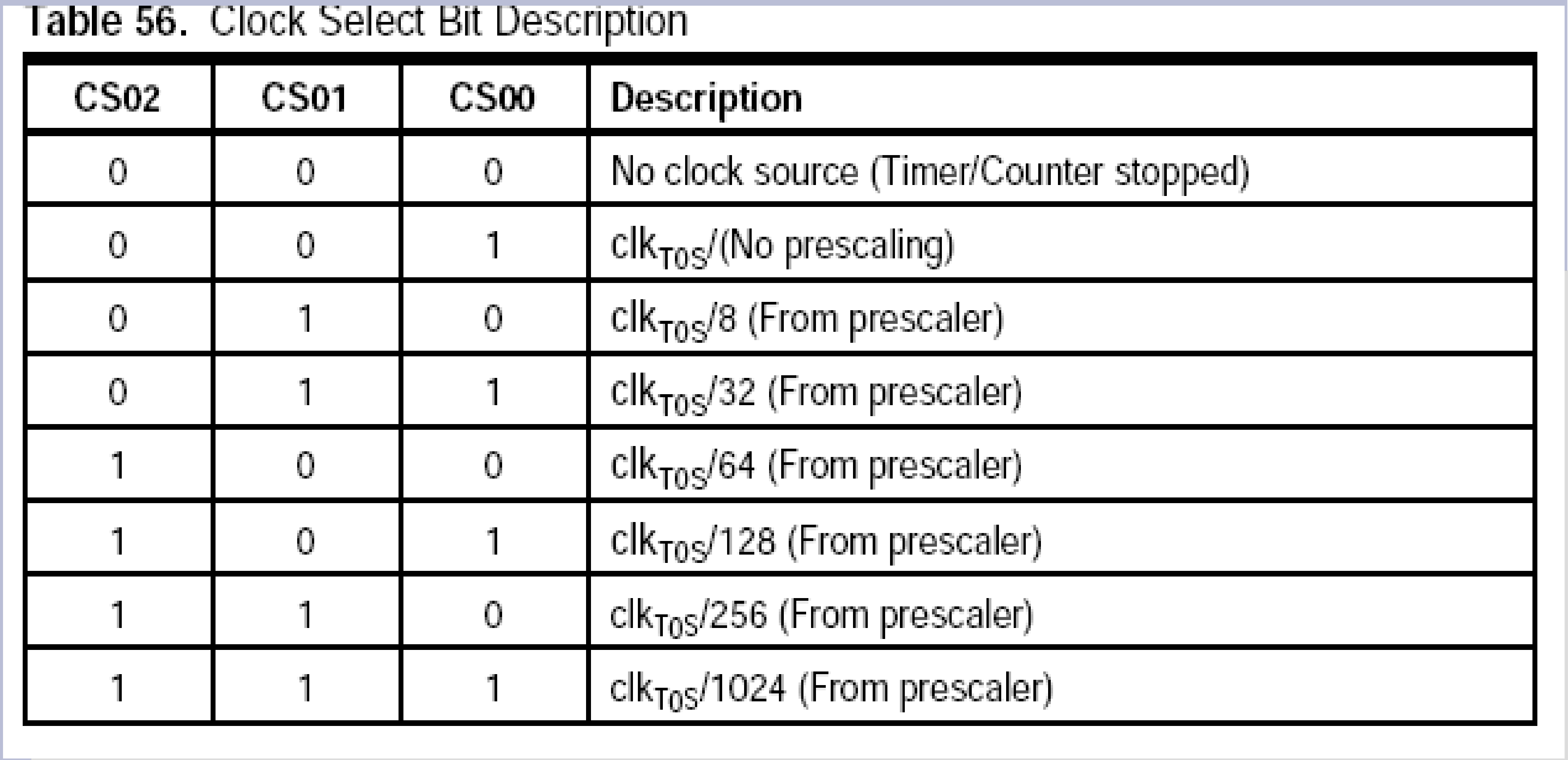
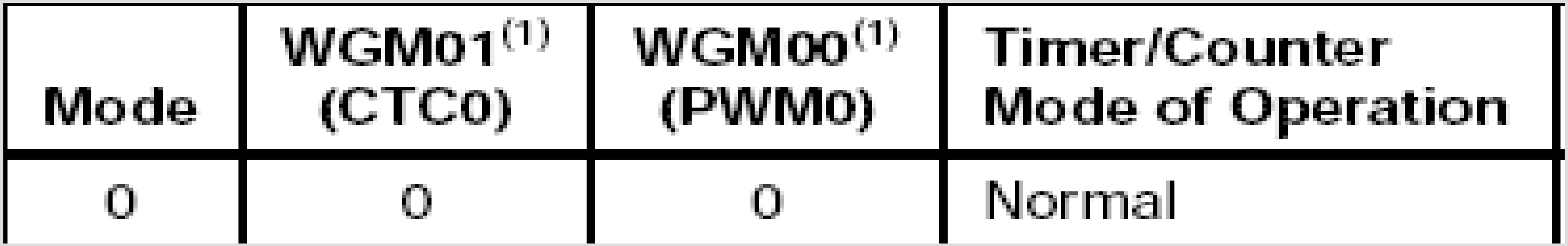
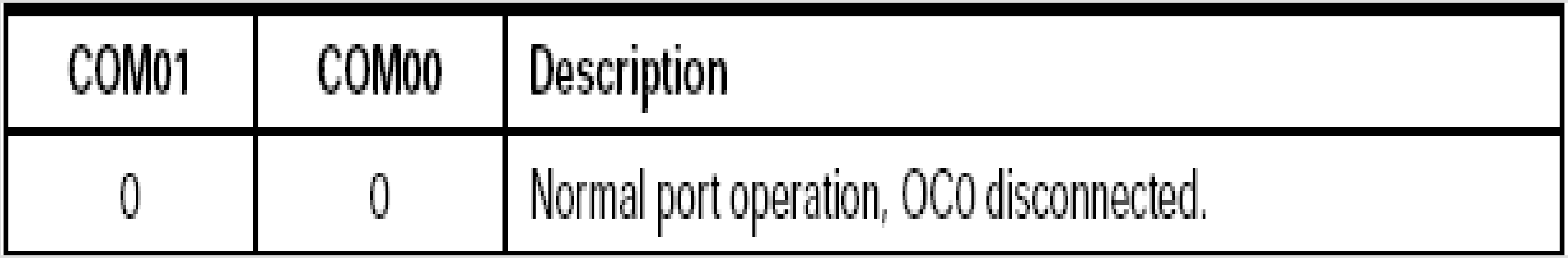


Table 56. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)



Mode	WGM01[1] (CTC0)	WGM00[1] (PWM0)	Timer/Counter Mode of Operation
0	0	0	Normal



COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.