Hochschule München FK03	Embedded Systems, 90 Minuten	Prof. Dr.-Ing. T. Küpper
Zugelassene Hilfsmittel: alle eigenen, Taschenrechner	**Matr.-Nr.:** **Name, Vorname:** **Hörsaal:** **Unterschrift:**	

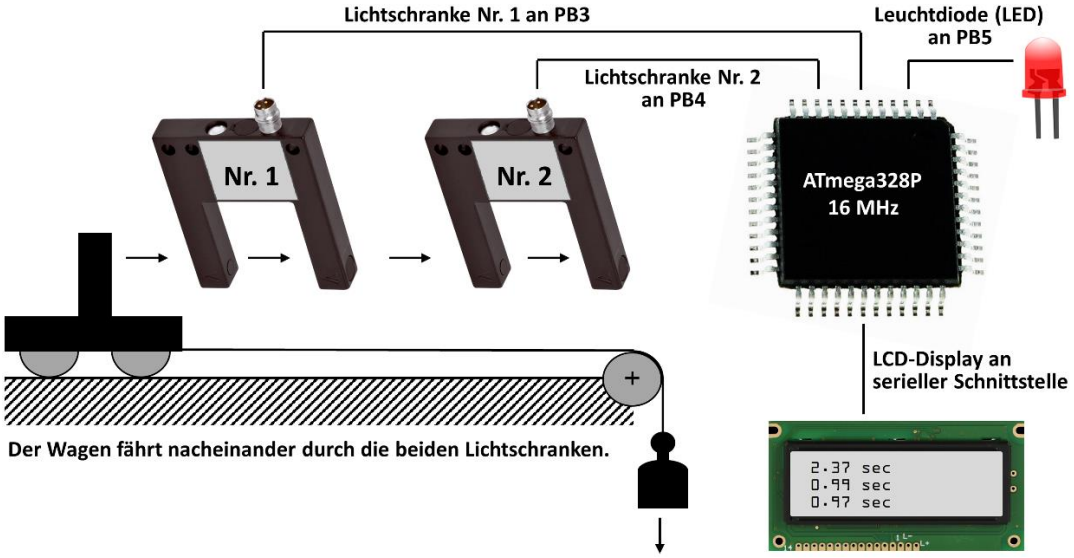
WiSe 2022/23

Viel Erfolg!!

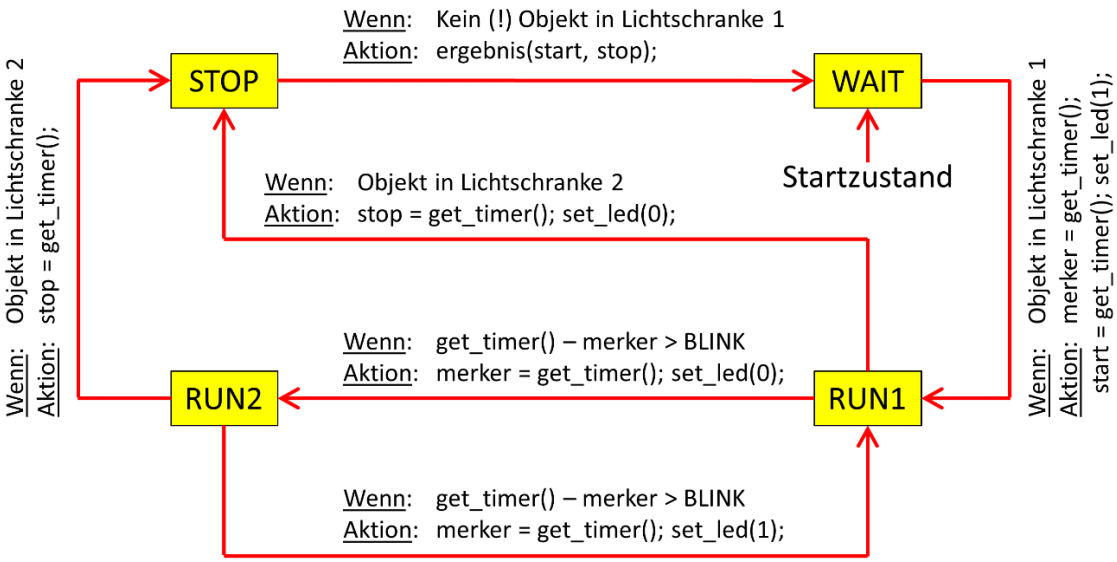
1	2	3	4	5	6		Σ	N

Aufgabe 1: Mikrocontroller-Programmierung, Timer für Physik-Experiment (ca. 35 Punkte ≙ 35 Minuten)

In einem Physik-Experiment rollt ein kleiner Wagen über einen Tisch. Mit zwei Lichtschranken soll die Zeitdauer gemessen werden, die der Wagen von „Position 1“ (Lichtschranke Nr. 1) bis „Position 2“ (Lichtschranke Nr. 2) benötigt. Die gemessene Zeit wird auf einem LCD-Display ausgegeben.



- Es wird ein ATmega328P-Mikrocontroller verwendet. Die Taktfrequenz beträgt 16 MHz.
- Lichtschranke 1 ist an PB3 angeschlossen, Lichtschranke 2 ist an PB4 angeschlossen.
- Das LCD-Display ist an die serielle Schnittstelle des Mikrocontrollers angeschlossen.
- Am Anschluss PB5 befindet sich eine Leuchtdiode (LED).
- Die beiden Lichtschranken verhalten sich exakt so, wie die mechanischen Taster im Embedded-Systems-Praktikum: Solange sich ein Objekt in der Lichtschranke befindet, wird der Anschluss PB3 bzw. PB4 mit Masse/GND verbunden. Genauso wie bei den Tastern im Praktikum, müssen auch die Pullup-Widerstände an PB3 und PB4 aktiviert werden.



```
// Timer für Physik-Experimente, T. Küpper, 01/2023
#define F_CPU 16000000UL
#include <inttypes.h>
#include <stdio.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

// Ports für Status-LED, Lichtschranke 1 und Lichtschranke 2
#define LED PB5
#define LICHT1 PB3
#define LICHT2 PB4

volatile uint32_t _timer_intern = 0;  // (Aufg. 1.7, Aufg. 1.8)

// Timer-Interrupt
ISR(TIMER1_COMPA_vect)
{
	_timer_intern++;
}

// Timer abfragen (Aufg. 1.6)
uint32_t get_timer(void)
{
	uint32_t result = 0;
	cli(); result = _timer_intern; sei();
	return result;
}

// 16-Bit-Timer (!) TC1 initialisieren (Aufg. 1.5)
void init_timer(void)
{
	OCR1AH = 0b00000010; OCR1AL = 0b01110001;
	TCCR1B = _BV(WGM12) + _BV(CS12);
	TIMSK1 = _BV(OCIE1A);
	sei();
}

// Serielle Schnittstelle initialisieren
#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1

void usart_init(uint16_t ubrr)
{
	UBRR0L = (uint8_t)(ubrr);
	UBRR0H = (uint8_t)(ubrr >> 8);		// Set baud rate
	UCSR0B = _BV( RXEN0) + _BV( TXEN0);	// Enable receiver & transmitter
	UCSR0C = _BV(UCSZ00) + _BV(UCSZ01);	// 8 data bits, 1 stop bit
}

// Ein einzelnes Zeichen über die Schnittstelle senden (Aufg. 1.9)
void usart_transmit(uint8_t data)
{
	while(!(UCSR0A & _BV(UDRE0))) { }
	UDR0 = data;
}
```

```
// Ergebnis auf serielle Schnittstelle schreiben (Aufg. 1.4)
char buf[32];

void ergebnis(uint32_t start, uint32_t stop)
{
	double tmp = (stop - start) / 100.0;
	sprintf(buf, "%8.2f sec\r\n", tmp);
	for(uint8_t i = 0; buf[i] != 0; ++i)
	{
		usart_transmit(buf[i]);
	}
}

// Befindet sich jetzt gerade ein Objekt in einer Lichtschranke?
// Parameter: nr == 1 oder 2 --> Nummer der Lichtschranke
// Rückgabe: 1 (es ist ein Objekt in der Lichtschranke) oder 0 (kein Objekt) (Aufg. 1.3)
int objekt_in_ls(int nr)
{









}

// LED an PB5 einschalten (on_off == 1) oder ausschalten (on_off == 0) (Aufg. 1.2)
void set_led(int on_off)
{









}

// Ein- und Ausgänge initialisieren, Pullups für Lichtschranken aktivieren (Aufg. 1.1)
void init_ports(void)
{









}
```

```c
// Endlicher Automat, Liste der Zustände
#define STATE_WAIT 0
#define STATE_RUN1 1
#define STATE_RUN2 2
#define STATE_STOP 3

// Hauptprogramm
#define BLINK 10

int main(void)
{
	usart_init(MYUBRR);
	init_ports();
	init_timer();

	uint8_t state = STATE_WAIT;
	uint32_t merker = 0, start = 0, stop = 0;

	while (1)  // (Aufg. 1.10)
	{
```

1.1. Schreiben Sie den Quelltext der Funktion `void init_ports(void)` zur Initialisierung der Ein- und Ausgänge des Mikrocontrollers.

1.2. Schreiben Sie den Quelltext der Funktion `void set_led(int on_off)` zum Ein- und Ausschalten der Leuchtdiode (LED) an PB5.

1.3. Schreiben Sie den Quelltext der Funktion `int objekt_in_ls(int nr)`. Mit dieser Funktion wird abgefragt, ob sich in der angegebenen Lichtschranke (nr == 1 oder 2) gerade ein Objekt befindet (Rückgabe == 1) oder nicht (Rückgabe == 0).

1.4. Welcher Text wird über die serielle Schnittstelle gesendet, falls die Funktion ergebnis() folgendermaßen aufgerufen wird? `ergebnis(1000, 1150);`

1.5. Wie oft wird die Funktion `ISR(TIMER1_COMPA_vect)` pro Sekunde aufgerufen, nachdem der 16-Bit-Timer TC1 durch Aufruf der Funktion `void init_timer(void)` initialisiert wurde?

1.6. Warum sind die Aufrufe von `cli();` und `sei();` in der Funktion `uint32_t get_timer(void)` erforderlich?

1.7. Nach wie vielen Tagen hat die Variable `_timer_intern` ihren maximal möglichen Wert erreicht? Und was passiert danach?

1.8. Was würde passieren, wenn die Variable `_timer_intern` nicht als `volatile` deklariert wäre?

1.9. Wozu dient die (leere) while-Schleife in der Funktion `void usart_transmit(uint8_t data)`?

1.10. Schreiben Sie die fehlenden Befehle des Hauptprogramms `int main(void)` auf Seite 4.

1.11. Manchmal passiert es, dass der Wagen zwar durch die erste Lichtschranke fährt, aber die zweite Lichtschranke nicht mehr erreicht: zum Beispiel, weil er der Wagen vom Tisch fällt oder weil die Umlenkrolle klemmt. Dann wartet der Mikrocontroller vergeblich auf die zweite Lichtschranke und die Messung „bleibt hängen“.

Darum wird das System erweitert: Es wird ein Taster an den Mikrocontroller angeschlossen, mit dem eine laufende Messung abgebrochen werden kann. Bei Betätigung des Tasters wird auf der seriellen Schnittstelle der Text „Cancel“ ausgegeben und das System wartet auf den Beginn einer neuen Messung (Lichtschranke 1).

(Wenn gerade keine Messung läuft, hat der Taster keine Funktion.)

Zeichnen Sie in das Zustandsdiagramm auf der ersten Seite die notwendigen Erweiterungen, die für diesen Taster benötigt werden. (In dieser Unteraufgabe 1.11. sollen also keine C-Befehle geschrieben werden!)

Aufgabe 2: C-Programmierung auf Mikrocontrollern, Verschiedenes (ca. 15 Punkte ≙ 15 Minuten)

2.1. Wie lauten die Ergebnisse der folgenden Berechnungen? (Bitte als Dualzahl/Binärzahl schreiben!)

a)
```
uint8_t a1 = 0b11111111;
a1 &= ~_BV(1);
```

b)
```
uint8_t a2 = 0b11111111;
a2 |= _BV(1);
```

c)
```
uint8_t a3 = 0b11111111;
a3 -= ~_BV(1);
```

d)
```
uint8_t a4 = 0b11111111;
a4 += ~_BV(1);
```

e)
```
uint8_t a5 = (1 << 3);
```

f)
```
uint8_t a6 = (3 << 1);
```

g)
```
uint8_t a7 = _BV(1) | _BV(0);
```

h)
```
uint8_t a8 = 1 + 0;
```

2.2. Der Zustand eines Tasters am Eingang PB1 eines ATmega328P-Mikrocontrollers soll eingelesen werden. Die folgenden Befehle funktionieren allerdings nicht:

```
// Hinweis: Die Variable "taster_gedrueckt" ist weiter oben definiert.
if((PORTB & _BV(1)) == 1)
{
    // Taster nicht gedrückt
    taster_gedrueckt = 0;
}
else
{
    // Taster gedrückt
    taster_gedrueckt = 1;
}
```

a) Beschreiben Sie in einigen Stichworten, wo das Problem liegt.
b) Korrigieren Sie den Fehler.

2.3. Erläutern Sie in einige Stichworten den Unterschied zwischen
a) einem „von-Neumann-Rechner“ und
b) einem „Harvard-Rechner“.

2.4. Was versteht man unter der „modifizierten Harvard-Architektur“ bei einem Mikrocontroller des Typs ATmega-328P, der im Praktikum verwendet wurde?

Aufgabe 3: Serielle Schnittstelle (ca. 15 Punkte ≙ 15 Minuten)

3.1. Auf der Arduino-Platine, die im Praktikum verwendet wurde, befindet sich ein ATmega328P-Mikrocontroller. Es ist problemlos möglich, Daten über die serielle Schnittstelle des Microcontrollers an einen PC zu senden. Dies funktioniert sogar, obwohl moderne PCs über gar keine „klassischen“ seriellen Schnittstellen mehr verfügen sondern nur über einige USB-Anschlüsse.

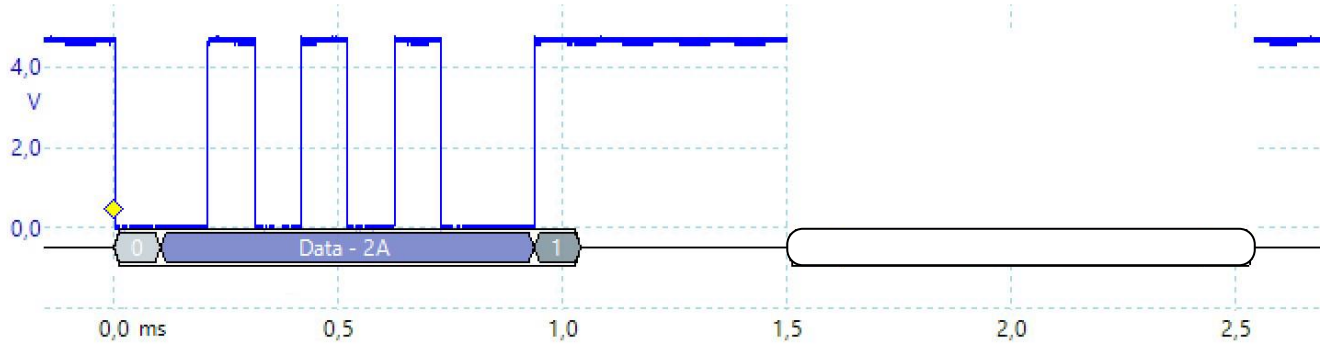
Beschreiben Sie in wenigen Stichworten (oder mit einer passenden Skizze!), wie die serielle Datenübertragung vom Mikrocontroller bis zur PC-Applikation funktioniert.

3.2. Die Datenübertragungsrate einer seriellen Schnittstelle beträgt 19200 Bits pro Sekunde, es werden acht Daten- und ein Startbit gesendet. Wie lange dauert die Übertragung eines MP4-Videos mit einer Größe von 1.000.000 Bytes über diese Schnittstelle minimal, wenn
a) ein Stoppbit gesendet wird?
b) zwei Stoppbits gesendet werden?

3.3. Durch einen Fehler bei der Konfiguration des Senders werden 2 Stoppbits gesendet (korrekt wäre nur 1 Stoppbit gewesen). Beschreiben Sie den Effekt dieses Fehlers auf den Ablauf der seriellen Datenübertragung.
a) Funktioniert die Datenübertragung noch?
b) Wer erkennt ggf. den Fehler?

3.4. Über die serielle Schnittstelle eines ATmega328P wird zunächst das Zeichen * (Stern, ASCII-Code = 42_{DEZ}) übertragen. Anschließend wird das Zeichen @ gesendet, der ASCII-Code (als Dezimalzahl) dieses Zeichens ist 96_{DEZ}. Zeichnen Sie den Spannungsverlauf, der sich am Anschluss TXD/PD1 (= Ausgang der seriellen Schnittstelle) beim Senden des zweiten Zeichens (@, ASCII-Code = 96_{DEZ}) ergibt, in das vorbereitete Diagramm.

Die Übertragungsparameter sind: 9,6 kbit/s; 8 Datenbits; kein Paritätsbit; 1 Stoppbit.



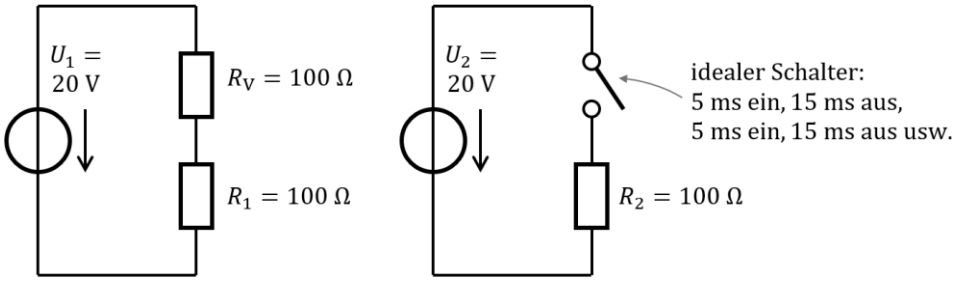
3.5. Was ändert sich am Spannungs-Verlauf im Vergleich zu Unterpunkt 3.4, wenn die Übertragung über eine serielle Schnittstelle nach RS232-Standard erfolgt. (Kurze Beschreibung oder geeignete Skizze angeben!)

Aufgabe 4: Elektronik, PWM (ca. 15 Punkte ≙ 15 Minuten)

4.1. Der Verbraucher $R_1 = 100\ \Omega$ ist über einen Vorwiderstand $R_V = 100\ \Omega$ an die Spannungsquelle $U_1 = 20\ \mathrm{V}$ angeschlossen.

a) Welche Leistung P_1 wird vom Verbraucher R_1 aufgenommen?

b) Berechnen Sie den Wirkungsgrad η_1 (= Leistung P_1 bezogen auf die Gesamtleistung, welche von der Spannungsquelle U_1 abgegeben wird).

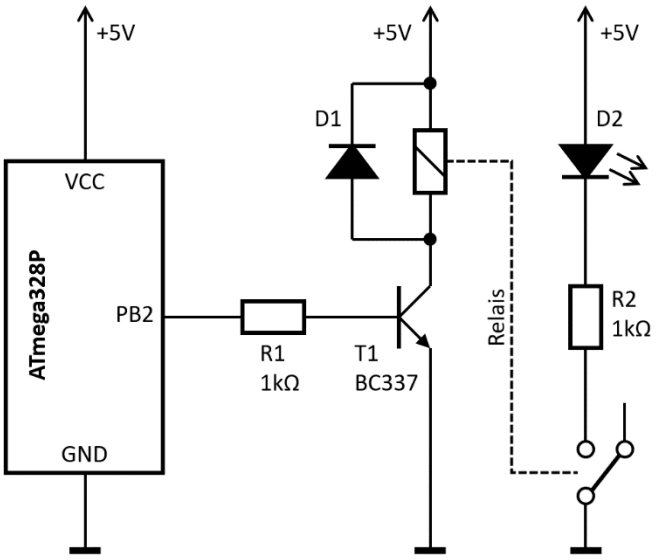


Der Verbraucher $R_2 = 100\ \Omega$ ist über einen idealen Schalter an die Spannungsquelle $U_2 = 20\ \mathrm{V}$ angeschlossen. Der Schalter wird 5 ms eingeschaltet, 15 ms ausgeschaltet, dann wieder 5 ms eingeschaltet, 15 ms ausgeschaltet und immer so weiter …

c) Welche mittlere (!) Leistung $\overline{P_2}$ wird vom Verbraucher R_2 aufgenommen?

d) Berechnen Sie den Wirkungsgrad η_2 (= Leistung $\overline{P_2}$ bezogen auf die Gesamtleistung, welche von der Spannungsquelle U_2 abgegeben wird).

4.2. Im Praktikum wurde zur Ansteuerung eines <u>mechanischen</u> Relais die folgende Transistorschaltung verwendet.
<u>Annahmen</u>: Die Spannung am (eingeschalteten) Anschluss PB2 sinkt durch die Belastung auf 4,5 Volt ab;
die Basis-Emitter-Spannung beträgt 0,5 Volt; der Verstärkungsfaktor des Transistors ist B = 50.



a) Welcher Strom I_{B2} fließt am Anschluss PB2 des Mikrocontrollers, wenn das Relais eingeschaltet ist?

b) Es wird ein Relais eingesetzt, welches zum Einschalten einen Spulenstrom von $I_{\mathrm{ein}} = 50\ \mathrm{mA}$ benötigt. (Der ohmsche Widerstand der Relais-Spule beträgt ca. 100 Ω.) Zeigen Sie durch eine kurze Rechnung, ob dieses Relais mit der abgebildeten Transistorschaltung verwendet werden kann oder nicht.

c) Wozu ist die Diode D1 erforderlich?

d) Ist die abgebildete Schaltung geeignet, um die Helligkeit der Leuchtdiode D2 mittels PWM zu regulieren? (Kurze Begründung erforderlich!)

Aus dem Datenblatt des Mikrocontrollers ATmega328P

20.14.1. TC1 Control Register A

Name: TCCR1A

Bit	7	6	5	4	3	2	1	0
	COM1	COM1	COM1	COM1			WGM11	WGM10
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

20.14.2. TC1 Control Register B

Name: TCCR1B

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Table 20-6. Waveform Generation Mode Bit Description

Mode	WGM13	WGM12 (CTC1)(1)	WGM11 (PWM11)(1)	WGM10 (PWM10)(1)	Timer/ Counter Mode of Operation	TOP	Update of OCR1x at	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	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	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	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Table 20-7. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)

20.14.12. Timer/Counter 1 Interrupt Mask Register

Name: TIMSK1

Bit	7	6	5	4	3	2	1	0
			ICIE			OCIE1B	OCIE1A	TOIE
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

Bit 1 – OCIE1A: Output Compare A Match Interrupt Enable

When this bit is written to '1', and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCFA Flag, located in TIFR1, is set.

20.14.8. Output Compare Register 1 A Low byte

Name: OCR1AL

Bit	7	6	5	4	3	2	1	0
	OCR1AL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCR1AL[7:0]: Output Compare 1 A Low byte

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1x pin.

20.14.9. Output Compare Register 1 A High byte

Name: OCR1AH

Bit	7	6	5	4	3	2	1	0
	OCR1AH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCR1AH[7:0]: Output Compare 1 A High byte