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

WiSe 2019/20

Viel Erfolg!!

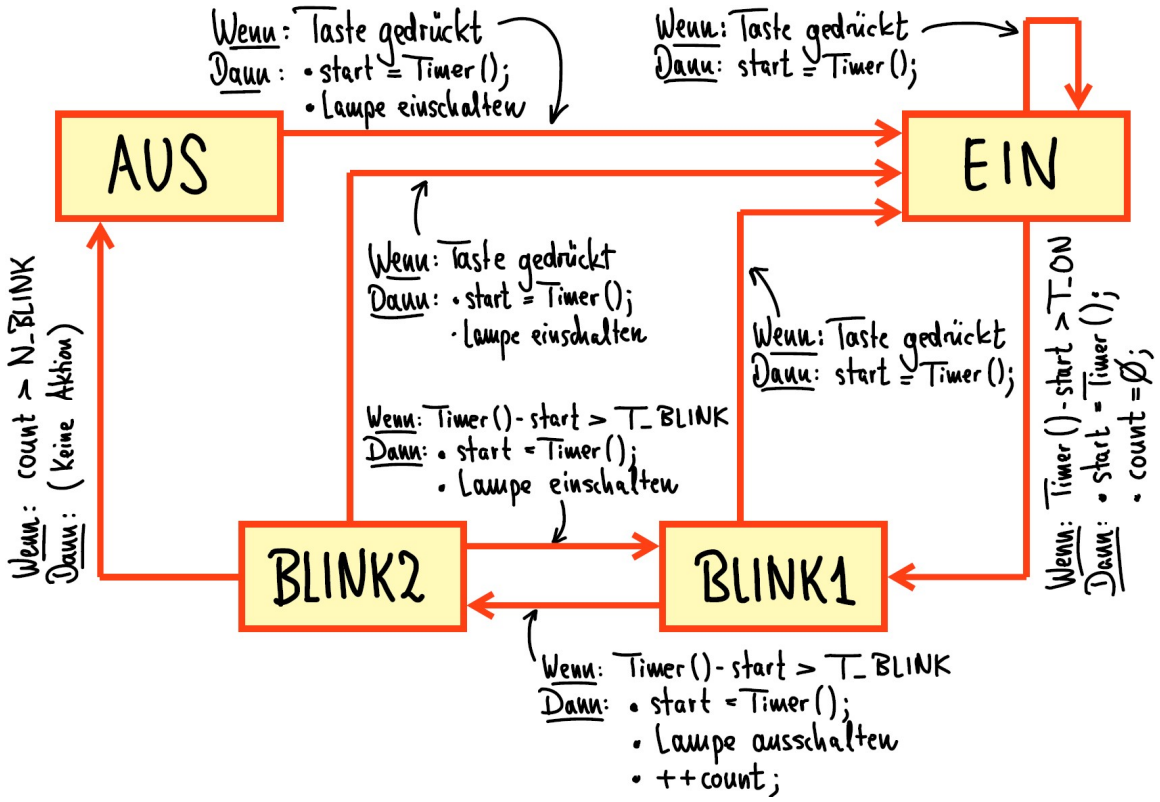
1	2	3	4	5	6		Σ	N

Aufgabe 1: Mikrocontroller-Programmierung (ca. 30 Punkte ≙ 30 Minuten)

In dieser Aufgabe soll die Software zur Steuerung einer Treppenhaus-Beleuchtung entwickelt werden.

- Es wird ein ATmega328P-Mikrocontroller verwendet.
- Das Treppenlicht wird über ein Relais am Anschluss PB2 ein- bzw. ausgeschaltet.
- Zwischen dem Anschluss PD2 und Masse ist ein Taster angeschlossen, durch Drücken des Tasters wird die Treppenhaus-Beleuchtung eingeschaltet. (Wenn der Taster gedrückt wird, dann verbindet er den Anschluss PD2 mit Masse/GND.)
- Zwischen dem Anschluss PD3 und Masse ist ein Schlüsselschalter angeschlossen, mit dem der Hausmeister oder das Reinigungspersonal die Beleuchtung dauerhaft einschalten kann. (Wenn der Schlüsselschalter eingeschaltet ist, dann verbindet er den Anschluss PD3 mit Masse/GND.)
- Der Schlüsselschalter wird erst in Unterpunkt 1.7 zur Software hinzugefügt!

Der Ablauf der Steuerung wird durch den folgenden endlichen Automaten beschrieben:



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

1.2. Programmieren Sie die beiden Funktionen `int taster_gedrueckt(void)` und `int schluessel_ein(void)`. Beide Funktionen geben eine 1 zurück, falls der betreffende Taster/Schalter gerade eingeschaltet ist, ansonsten ist die Rückgabe gleich 0. (Die Funktonen sollen also nicht warten, bis der Taster/Schalter wieder geöffnet wird, sondern das Ergebnis sofort zurückgeben. Es soll auch nichts „entprellt“ werden...)

1.3. In der Funktion `void relais_schalten(int ein_aus)` gibt es den folgenden Befehl: `PORTB &= ~_BV(RELAIS);` Wozu dient dieser Befehl? Warum wäre es keine gute Idee, stattdessen einfach nur `PORTB = 0;` zu schreiben?

1.4. In welchem Modus wird Timer 1 betrieben? Wie oft pro Sekunde wird die Funktion `ISR(TIMER1_COMPA_vect)` aufgerufen? Hinweis: Beachten Sie die Tabellen aus dem Datenblatt des Mikrocontrollers, die gegen Ende der Aufgabe 1 abgedruckt sind!

1.5. Wie lange bleibt die Treppenhaus-Beleuchtung eingeschaltet, nachdem der Taster losgelassen wird? Was passiert danach (und wie oft und wie schnell)?

1.6. Vervollständigen Sie das Hauptprogramm „main“ im abgebildeten C-Quelltext.

1.7. Zwischen dem Anschluss PD3 und Masse ist ein Schlüsselschalter angeschlossen, mit dem der Hausmeister oder das Reinigungspersonal die Beleuchtung dauerhaft einschalten kann.

Wenn der Schlüsselschalter eingeschaltet ist, dann soll die Treppenhaus-Beleuchtung ständig eingeschaltet bleiben. Nach dem Ausschalten des Schlüsselschalters soll die Treppenhaus-Beleuchtung nicht direkt ausgehen, sondern solange weiterleuchten, wie dies auch ganz normal nach dem Drücken des Tasters der Fall ist.

Ergänzen Sie den endlichen Automaten auf der ersten Seite, damit der Schlüsselschalter die gewünschte Funktion erhält. (Sie müssen in Unterpunkt 1.7. keinen C-Quelltext programmieren!)

Tipp: Ein neuer Zustand und drei neue Zustandsübergänge reichen schon aus…

```
// ----------------------------------------------------------------------------
// Steuerung einer Treppenhaus-Beleuchtung, WS 2019/20
// ----------------------------------------------------------------------------
#define F_CPU 16000000UL
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>

// Anschlüsse von Taster, Schlüsselschalter und Relais
#define TASTER PD2
#define SCHLUESSEL PD3
#define RELAIS PB2

// Wie lange ist die Lampe an?
#define T_ON  100000UL
#define T_BLINK 2500UL
#define N_BLINK 10

// Globale Variable: Wieviel Zeit ist seit dem Programmstart vergangen?
volatile uint32_t _timer_intern = 0;

// ----------------------------------------------------------------------------
// Interrupt-Funktion, wird durch den Timer 1 aufgerufen
// ----------------------------------------------------------------------------
ISR(TIMER1_COMPA_vect)
{
    _timer_intern++;
}

// ----------------------------------------------------------------------------
// Wieviel Zeit ist seit dem Programmstart vergangen?
// ----------------------------------------------------------------------------
uint32_t Timer(void)
{
    uint32_t result = 0;
    cli(); result = _timer_intern; sei();
    return result;
}

// ----------------------------------------------------------------------------
// Timer 1 initialisieren
// ----------------------------------------------------------------------------
void init_timer(void)  // *** Aufgabe 1.4 ***
{
    TCCR1B  = 0b00001011;
    OCR1A   = 24;
    TIMSK1  = _BV(OCIE1A);
    sei();
}

// ----------------------------------------------------------------------------
// PB2 = Relais (Ausgang); PD2 = Taster (Eingang mit Pullup);
// PD3 = Schlüsselschalter (Eingang mit Pullup)
// ----------------------------------------------------------------------------
void init_ports(void)  // *** TODO: Aufgabe 1.1 ***
{









}
```

```
// ----------------------------------------------------------------------------
// Wird der Taster gerade gedrückt?
// ----------------------------------------------------------------------------
int taster_gedrueckt(void)  // *** TODO: Aufgabe 1.2 ***
{









}

// ----------------------------------------------------------------------------
// Ist der Schlüsselschalter gerade eingeschaltet?
// ----------------------------------------------------------------------------
int schluessel_ein(void)  // *** TODO: Aufgabe 1.2 ***
{









}

// ----------------------------------------------------------------------------
// Relais ein- (1) oder ausschalten (0)
// ----------------------------------------------------------------------------
void relais_schalten(int ein_aus)
{
    if(ein_aus == 1) PORTB |=  _BV(RELAIS);
    if(ein_aus == 0) PORTB &= ~_BV(RELAIS);  // *** Aufgabe 1.3 ***
}

// Zustände, die die Treppenhaus-Beleuchtung durchläuft
#define ZUSTAND_AUS    1
#define ZUSTAND_EIN    2
#define ZUSTAND_BLINK1 3
#define ZUSTAND_BLINK2 4
#define ZUSTAND_PRIO   5

// ----------------------------------------------------------------------------
// Hauptprogramm
// ----------------------------------------------------------------------------
int main(void)
{
    uint32_t start = 0;
    uint8_t  count = 0;
    uint8_t  state = ZUSTAND_AUS;  // Zunächst ist das Licht aus!

    init_ports();
    init_timer();          // *** TODO: Aufgabe 1.6 ***
```

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			OCIEB	OCIEA	TOIE
Access			R/W			R/W	R/W	R/W
Reset			0			0	0	0

Bit 0 – TOIE: Overflow 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 1 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV Flag, located in TIFR1, is set.

Aufgabe 2: C-Programmierung auf Mikrocontrollern, Verschiedenes (ca. 20 Punkte ≙ 20 Minuten)

Das folgende Programm für den Mikrocontroller ATmega328P dient zur Ausgabe eines sinusförmigen Spannungsverlaufs mittels PWM. Im Gegensatz zum Programm aus dem Embedded-Systems-Praktikum wird hier der Timer/Counter1 verwendet, bei dem es sich um einen 16-Bit-Zähler (!!) handelt. (Datenblatt: siehe Seite 6!)

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>

// Sinuswerte von 0°...358° in Schritten von 2°
const uint16_t sin_tab[] PROGMEM =
{
    512 , 530 , 548 , 565 , 583 , 601 , 618 , 636 , 653 , 670 , 687 , 703 ,
    720 , 736 , 752 , 768 , 783 , 798 , 812 , 827 , 840 , 854 , 867 , 880 ,
    892 , 903 , 915 , 925 , 936 , 945 , 955 , 963 , 971 , 979 , 986 , 992 ,
    998 , 1003, 1008, 1012, 1015, 1018, 1020, 1022, 1023, 1023, 1023, 1022,
    1020, 1018, 1015, 1012, 1008, 1003, 998 , 992 , 986 , 979 , 971 , 963 ,
    955 , 945 , 936 , 925 , 915 , 903 , 892 , 880 , 867 , 854 , 840 , 827 ,
    812 , 798 , 783 , 768 , 752 , 736 , 720 , 703 , 687 , 670 , 653 , 636 ,
    618 , 601 , 583 , 565 , 548 , 530 , 512 , 494 , 476 , 459 , 441 , 423 ,
    406 , 388 , 371 , 354 , 337 , 321 , 304 , 288 , 272 , 256 , 241 , 226 ,
    212 , 197 , 184 , 170 , 157 , 144 , 132 , 121 , 109 , 99  , 88  , 79  ,
    69  , 61  , 53  , 45  , 38  , 32  , 26  , 21  , 16  , 12  , 9   , 6   ,
    4   , 2   , 1   , 1   , 1   , 2   , 4   , 6   , 9   , 12  , 16  , 21  ,
    26  , 32  , 38  , 45  , 53  , 61  , 69  , 79  , 88  , 99  , 109 , 121 ,
    132 , 144 , 157 , 170 , 184 , 197 , 212 , 226 , 241 , 256 , 272 , 288 ,
    304 , 321 , 337 , 354 , 371 , 388 , 406 , 423 , 441 , 459 , 476 , 494 ,
    0  // <--- Ende-Markierung
};

uint16_t sin_index = 0;  // Aktuelle Position in der Tabelle

// Nächsten Sinuswert aus Sinus-Tabelle lesen, PWM-Tastgrad aktualisieren
ISR(TIMER1_OVF_vect)
{
    // Hinweis: read_word statt read_byte, weil im Gegensatz zum
    // Praktikum hier mit 16-Bit-Werten (!!) gearbeitet wird...
    uint16_t pwm = pgm_read_word_near(sin_tab + sin_index);
    if(!pwm) { sin_index = 0; pwm = pgm_read_word_near(sin_tab + sin_index); }

    OCR1A = pwm;  // Hinweis: OCR1A ist ein 16-Bit-Register (!!)
    sin_index++;
}

// PWM-Ausgabe und Timer-Overflow-Interrupt aktivieren
void init_pwm_and_interrupt(void)
{
    // *** TODO: Aufgabe 2.2, korrektes DDRx-Register setzen ***




    TCCR1A = _BV(COM1A1) + _BV(WGM10) + _BV(WGM11);
    TCCR1B = _BV(WGM12) + _BV(CS12);
    TIMSK1 = _BV(TOIE1);
    sei();
}

// Hauptprogramm
int main(void)
{
    init_pwm_and_interrupt();
    while(1) { }
}
```

When the OC1A or OC1B is connected to the pin, the function of the COM1x[1:0] bits is dependent of the WGM1[3:0] bits setting. The table below shows the COM1x[1:0] bit functionality when the WGM1[3:0] bits are set to a Normal or a CTC mode (non-PWM).

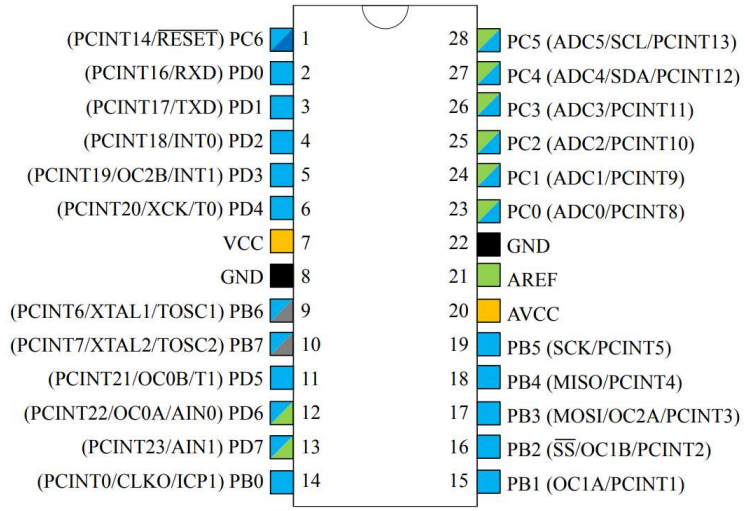
Table 20-3. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on Compare Match.
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level).
1	1	Set OC1A/OC1B on Compare Match (Set output to high level).

2.1. An welchem Anschluss (Pin) des Mikrocontrollers wird das PWM-Signal ausgegeben? Markieren Sie den Anschluss in der folgenden Grafik.

Pin-out

Figure 5-1. 28-pin PDIP



2.2. In der Funktion `void init_pwm_and_interrupt(void)` muss noch ein bestimmtes DDRx-Register eingestellt werden, damit überhaupt ein PWM-Signal am Ausgang des Mikrocontrollers ausgegeben wird. Fügen Sie den fehlenden Quelltext zur Funktion `void init_pwm_and_interrupt(void)` hinzu.

2.3. Vervollständigen Sie die folgenden Aussagen:

- Der Timer/Counter1 zählt in diesem Programm immer wieder von seinem minimalen Wert [ ] bis hin zu seinem maximalen Wert [ ].
- Es wird ein sinusförmiges Signal mit einer Frequenz von [ ] Hz ausgegeben.

2.4. Wozu dient die folgende Programmzeile in der Funktion `ISR(TIMER1_OVF_vect)`?

```
if(!pwm) { sin_index = 0; pwm = pgm_read_word_near(sin_tab + sin_index); }
```

2.5. Die Variable `sin_index` wird innerhalb einer Interrupt-Funktion verwendet/geschrieben/gelesen. Warum muss sie dennoch nicht als `volatile` deklariert werden?

2.6. Im Quelltext findet sich die folgende Zuweisung: `TCCR1A = _BV(COM1A1) + _BV(WGM10) + _BV(WGM11);`

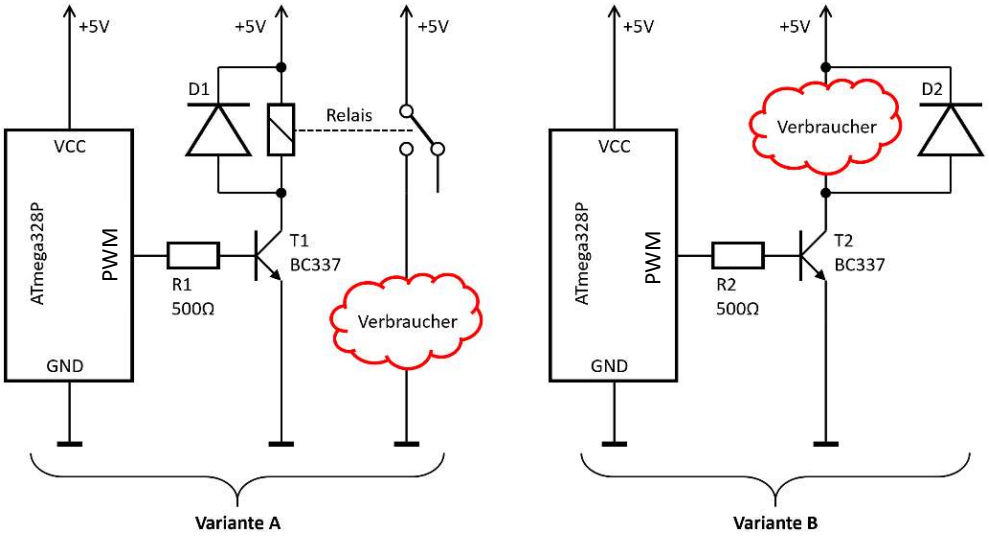
Hätte man stattdessen auch `TCCR1A = _BV(COM1A1) | _BV(WGM10) | _BV(WGM11);` schreiben können?

Zeigen Sie ein konkretes Beispiel für einen Ausdruck, der mit „+“ bzw. mit „|“ unterschiedliche Ergebnisse liefert. Wie lauten die beiden Ergebnisse in Ihrem Beispiel?

2.7. Das letzte Element des Vektors `const uint16_t sin_tab[] PROGMEM` hat den Wert null. Was wäre die Folge, wenn beim Schreiben des Programms dieses Element entfernt (vergessen…) würde?

2.8. In welchem Speicherbereich sind die Werte des Vektors `const uint16_t sin_tab[] PROGMEM` abgelegt? Ist so etwas bei einem Mikrocontroller, der auf der Harvard-Architektur basiert, möglich? Auf welcher Architektur basiert der hier verwendete Mikrocontroller des Typs ATmega328P?

2.9. Das vom abgebildeten Programm generierte PWM-Signal wird an die folgenden Schaltungen angeschlossen:



Als Verbraucher wird jeweils ein kleiner Gleichstrommotor verwendet. Sie dürfen annehmen, dass die Spannungs- und Stromwerte in dieser Schaltung grundsätzlich „passen“ und für den Motor ausreichend groß sind. Was passiert bei der linken, was passiert bei der rechten Schaltung?

Aufgabe 3: Mikrocontroller, Speicher (ca. 8 Punkte ≙ 8 Minuten)

3.1. Ein Speicherbaustein ist intern mit 16 Bitleitungen und 16 Wortleitungen aufgebaut. Wie viele Bits können in diesem Baustein gespeichert werden? (Hinweis: An jeder Adresse ist ein Bit gespeichert.)

3.2. Wie viele Adressleitungen sind beim Speicherbaustein aus Unterpunkt 3.1 nach außen geführt?

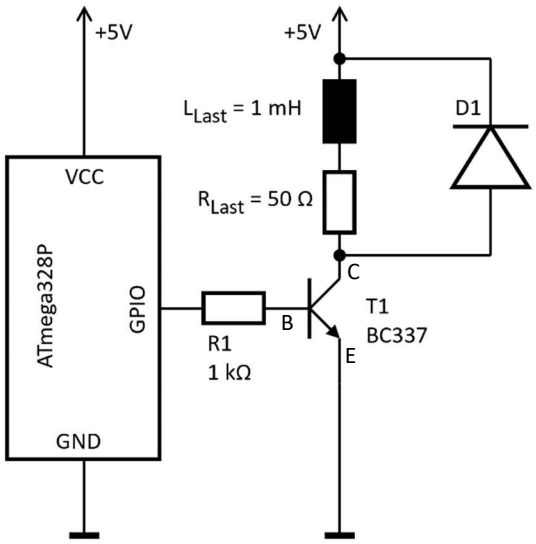
3.3. Wodurch unterscheiden sich Mikroprozessoren und Mikrocontroller? Nennen Sie zwei Unterschiede!

3.4. Zur Erweiterung des RAM-Speichers kann bei Bedarf ein externer SRAM-Baustein an den Mikrocontroller angeschlossen werden. Warum ist dies einfacher zu realisieren als eine Speichererweiterung mittels DRAM?

3.5. Zur Speichererweiterung wird ein separater SRAM-Baustein an einen Mikrocontroller angeschlossen. Woher „weiß“ dieser SRAM-Baustein, ob an einer vom Mikrocontroller gesendeten Speicheradresse Daten geschrieben oder gelesen werden sollen?

Aufgabe 4: GPIO-Ports (ca. 12 Punkte ≙ 12 Minuten)

Die Abbildung zeigt die Ansteuerung eines Verbrauchers mithilfe eines Transistors. Für die Basis-Emitter-Spannung des eingeschalteten Transistors gilt: U_{BE} = 0,6 Volt. Die Spannung am GPIO-Port des Mikrocontrollers beträgt entweder 0 Volt (aus) oder 4,6 Volt (ein). Der Stromverstärkungsfaktor („Großsignalverstärkung“) des Transistors ist B = 150.



4.1. Welcher Laststrom fließt durch den Transistor, wenn dieser dauerhaft eingeschaltet ist? Gehen Sie davon aus, dass der Transistor den Laststrom wie ein „idealer Schalter“ einschaltet (also ohne jeglichen Widerstand oder Spannungsverlust zwischen Kollektor und Emitter).

4.2. Welchen Basisstrom benötigt der Transistor mindestens, um den Laststrom aus Unterpunkt 4.1 einzuschalten?

4.3. Wie groß ist der tatsächlich fließende Basisstrom (beim eingeschalteten Transistor) in dieser Schaltung?

4.4. Warum ist die Diode D1 unbedingt erforderlich?

4.5. Wie groß ist der Strom durch die Diode D1 unmittelbar nach dem Abschalten des Transistors?

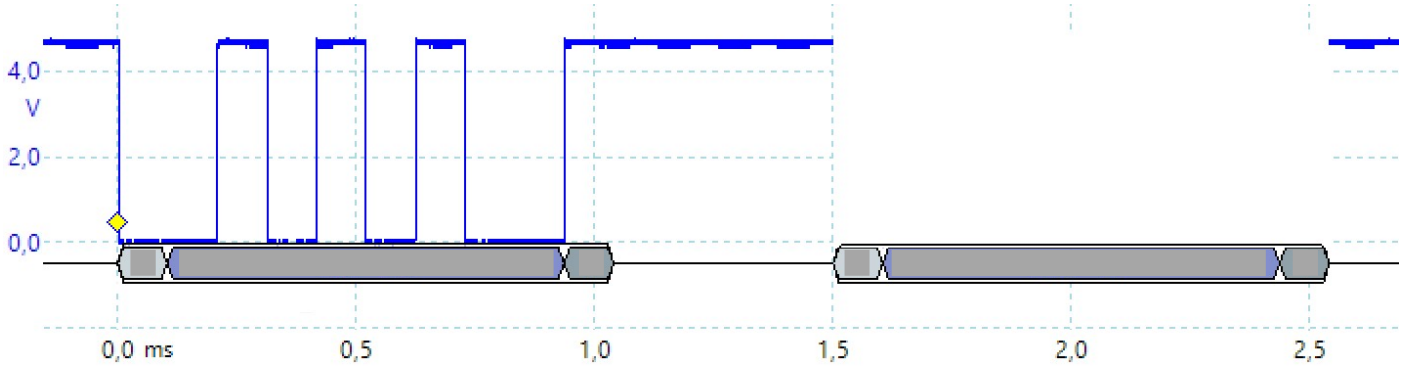
4.6. Innerhalb des Mikrocontrollers befinden sich am GPIO-Port zwei Schutzdioden. Zeichnen Sie diese (internen) Schutzdioden in die oben abgebildete Schaltung.

4.7. Wozu dienen die (internen) Schutzdioden aus Unterpunkt 4.6?

Aufgabe 5: Serielle Schnittstelle (ca. 10 Punkte ≙ 10 Minuten)

5.1. 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 35_{DEZ}. Die Übertragungsparameter sind: 9,6 kbit/s; 8 Datenbits; kein Paritätsbit; 1 Stoppbit.

Zeichnen Sie den Spannungsverlauf, der sich am Anschluss TXD/PD1 (= Ausgang der seriellen Schnittstelle) beim Senden des zweiten Zeichens (#, ASCII-Code = 35_{DEZ}) ergibt, in das vorbereitete Diagramm.



5.2. Wie lange dauert die Übertragung einer jpg-Datei mit einer Größe von 500.000 Bytes über eine serielle Schnittstelle minimal? Die Datenübertragungsrate der Schnittstelle betrage 9600 Bits pro Sekunde, es werden acht Daten-, ein Start- und ein Stoppbit gesendet.

5.3. Durch einen Fehler bei der Konfiguration des Senders werden 2 Stoppbits gesendet (korrekt wäre allerdings nur 1 Stoppbit gewesen). Beschreiben Sie den Effekt dieser Fehl-Konfiguration auf den Ablauf der seriellen Datenübertragung. Funktioniert die Datenübertragung noch? Wer erkennt ggf. den Fehler?