Hochschule München FK03	Komponenten und Programmierung		Prof. Dr.-Ing. T. Küpper
Zugelassene Hilfsmittel: alle eigenen, Taschenrechner	**Matr.-Nr.:**	**Name, Vorname:**	
	Hörsaal:	**Unterschrift:**	

SS 2017

Viel Erfolg!!

1	2	3	4	5	6		Σ	N

Aufgabe 1: Programmierung (ca. 26 Punkte ≙ 26 Minuten)

Es soll die Software zur Steuerung einer Klimaanlage entwickelt werden.

– KLIMAANLAGE –

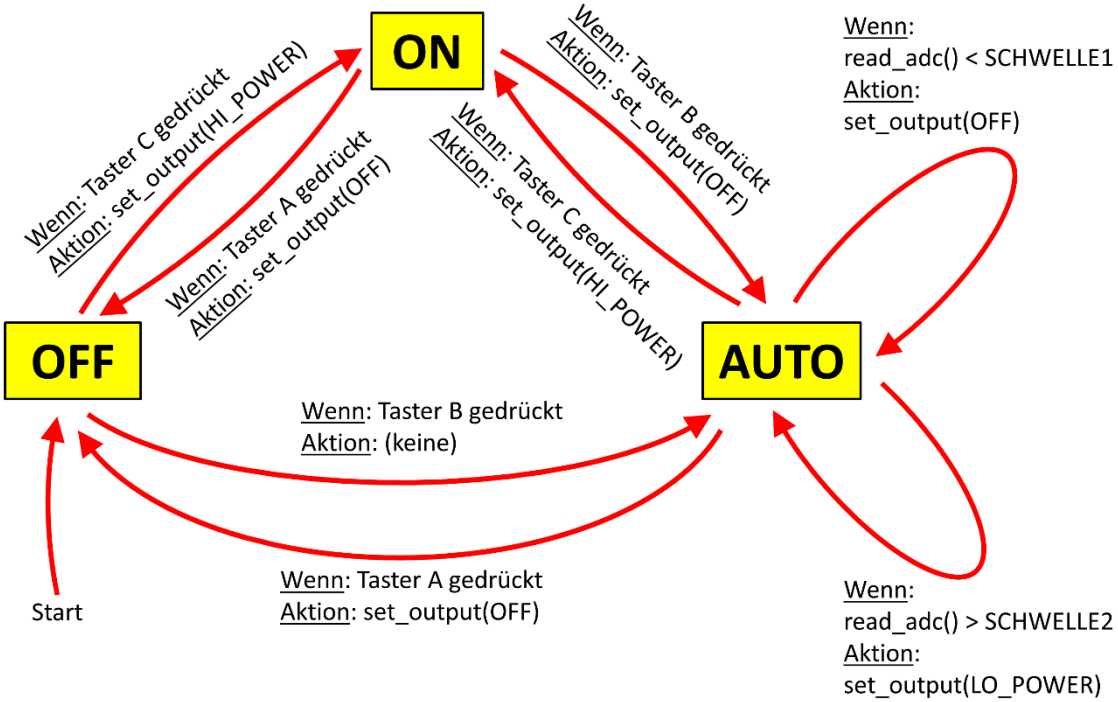
A AUS

C EIN

B AUTOMATIK

- Es wird ein ATmega328P-Mikrocontroller verwendet.
- Der Motor der Klimaanlage wird über den Anschluss PB1 angesteuert: Wenn an PB1 eine logische „1“ ausgegeben wird, dann läuft die Klimaanlage. Durch Ausgabe eines PWM-Signals an PB1 kann die Klimaanlage mit reduzierter Leistung betrieben werden.
- An den Anschlüssen PC0, PC1 und PC2 sind die Taster „A“, „B“ und „C“ angeschlossen (siehe nebenstehende Abbildung). Das Drücken eines Tasters führt zu einer logischen „0“ an dem entsprechenden Eingang (wie im Praktikum).
- Im Automatikmodus geht die Klimaanlage je nach aktueller Temperatur an oder aus. Am Eingang des Analog-Digital-Wandlers ist dazu ein Temperatursensor angeschlossen.
- Hinweis: In diesem Programm wird der 16-Bit-Timer/Counter1 verwendet und nicht der 8-Bit-Timer/Counter0.

Das Verhalten der Steuerung wird durch einen endlichen Automaten beschrieben:



1.1. Nachdem der Timer/Counter1 durch Aufruf der Funktion „init_timer“ (siehe Seite 4) initialisiert wurde, zählt er automatisch immer wieder von null bis zu einem bestimmten Maximalwert. Wie groß ist dieser Maximalwert?

20.14.1. TC1 Control Register A

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM1[3:0] = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode)

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

Table 20-7. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
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)

1.2. Welche Frequenz wird an PB1 ausgegeben, wenn das PWM-Signal aktiv ist?

1.3. Welcher Tastgrad wird nach dem Aufruf von „set_output(LO_POWER)“ an PB1 ausgegeben?

28.9.1. ADC Multiplexer Selection Register

MUX[3:0]	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	Temperature sensor

1.4. Wozu dient die Zuweisung an „PORTC“ in der Funktion „init_buttons“?

1.5. Der AD-Wandler wird durch Aufruf der Funktion „init_adc“ initialisiert. An welchem Eingang muss das Analogsignal (= der Temperatursensor) angeschlossen werden?

1.6. Die Funktion „read_adc“ dient zum Einlesen eines Analogwerts durch den Analog-Digital-Wandler. In welchem Wertebereich (min…max) kann sich der Rückgabewert dieser Funktion befinden?

1.7. Programmieren Sie den fehlenden Rest des Hauptprogramms (Funktion „main“).

```
// Mikrocontroller läuft mit Takt = 16 MHz  <--- !!!
#include <avr/io.h>

// Externe Anschlüsse
#define KLIMAANLAGE PB1
#define TASTER_A    PC0
#define TASTER_B    PC1
#define TASTER_C    PC2

// Temperatur-Schwellenwerte für Automatikmodus
#define SCHWELLE1 0.4
#define SCHWELLE2 0.6
```

```
// Timer/Counter1: PWM-Modus einschalten
void init_timer1(void)
{
    TCCR1A = _BV(WGM11) | _BV(WGM10) | _BV(COM1A1);
    TCCR1B = _BV(WGM12) | _BV(CS11);
}

// Taster "A" an PC0, Taster "B" an PC1, Taster "C" an PC2
void init_buttons(void)
{
    PORTC = _BV(TASTER_A) | _BV(TASTER_B) | _BV(TASTER_C);
}

// Analog-Digital-Wandler initialisieren
void init_adc(void)
{
    ADMUX  = _BV(REFS0) | _BV(MUX0) | _BV(MUX1) | _BV(MUX2);
    ADCSRA = _BV(ADEN) | _BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0);
}

// Einen Wert vom Analog-Eingang einlesen
double read_adc(void)
{
    ADCSRA |= _BV(ADSC);           // Wandlung starten
    while(ADCSRA & _BV(ADSC)) { }  // Auf das Ende der Wandlung warten
    return ADC / 1023.0;           // Ergebnis normieren und zurückgeben
}

// Klimaanlage auf hohe/mittere Leistung schalten bzw. ausschalten
#define OFF      0
#define LO_POWER 1
#define HI_POWER 2

void set_output(int power)
{
    if(power == OFF     ) { DDRB &= ~_BV(KLIMAANLAGE);                }
    if(power == LO_POWER) { DDRB |=  _BV(KLIMAANLAGE); OCR1A =  256; }
    if(power == HI_POWER) { DDRB |=  _BV(KLIMAANLAGE); OCR1A = 1023; }
    // Hinweis: Timer/Counter1 ist ein 16-Bit-Timer,
    // OCR1A hat eine Größe von 16 Bits!
}

// Hauptprogramm, endlicher Automat mit drei Zuständen
#define STATE_OFF  0
#define STATE_ON   1
#define STATE_AUTO 2

int main(void)
{
    init_timer1();
    init_buttons();
    init_adc();

    int state = STATE_OFF;
    set_output(OFF);
```

```
while (1)
{
    switch(state)
    {
    case STATE_OFF:
```

Aufgabe 2: Mikrocontroller, Speicher (ca. 10 Punkte ≙ 10 Minuten)

2.1. Ursprünglich war der ROM-Speicher in PCs mittels EPROMs aufgebaut. Diese sind inzwischen ausnahmslos durch Flash-Speicherbausteine ersetzt worden, aber warum? Nennen Sie zwei Vorteile von Flash-Speicherbausteinen im Vergleich zu EPROMs.

2.2. Welche Art von ROM-Speicher besitzt ein (durchsichtiges) Quarzglas-Fenster auf seinem IC-Gehäuse? Wozu dient dieses Quarzglas-Fenster?

2.3. 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.)

2.4. Wie viele Adressleitungen sind beim Speicherbaustein aus Unterpunkt 2.3 nach außen geführt?

2.5. Bei einem Mikrocontroller reicht der interne RAM-Speicher nicht aus, daher soll ein externer RAM-Baustein angeschlossen werden. Warum ist es aus schaltungstechnischer Sicht einfacher, zu diesem Zweck statisches RAM einzusetzen (und kein dynamisches RAM)?

2.6. Nennen Sie einen Nachteil von statischem RAM im Vergleich zu dynamischem RAM.

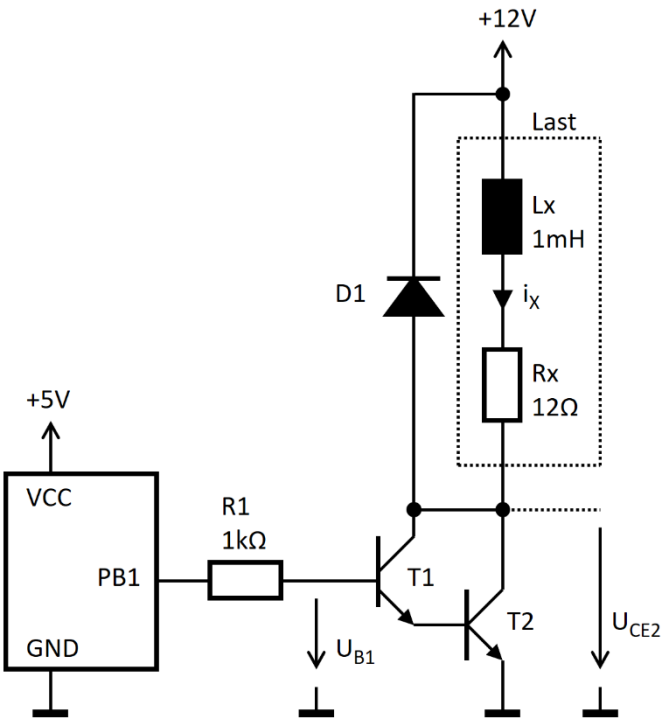
Aufgabe 3: GPIO-Ports (ca. 10 Punkte ≙ 10 Minuten)

Ein Verbraucher (= Reihenschaltung von Rx und Lx) soll über einen GPIO-Port ein- und ausgeschaltet werden. Die geplante Schaltfrequenz ist allerdings zu hoch für ein mechanisches Relais, und der Laststrom i_x ist zu groß für einen einstufigen Transistorverstärker.

Eine mögliche Lösung zeigt die nebenstehende Schaltung.

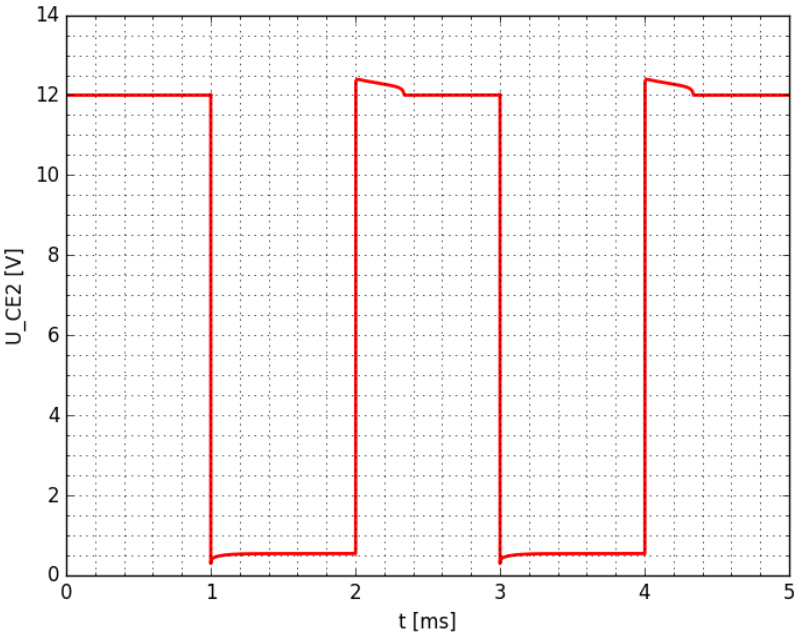
Die Transistoren T1 und T2 sind direkt miteinander verbunden (sog. Darlington-Schaltung). Diese Kombination von T1 und T2 verhält sich „von außen betrachtet“ wie ein einzelner Transistor mit einer sehr hohen Stromverstärkung $B = B_1 \cdot B_2$.

Die Spannung u_{B1} am Eingang der Darlington-Schaltung ist allerdings höher als die Basis-Emitter-Spannung eines einzelnen Transistors. Auch die Spannung u_{CE2} ist im eingeschalteten Zustand etwa höher als die Kollektor-Emitter-Spannung eines einzelnen Transistors.



3.1. Die folgende Abbildung zeigt den zeitlichen Verlauf der Spannung u_{CE2}, der mit einem Oszilloskop aufgenommen wurde. Zu welchen Zeitpunkten t_1 und t_2 wird der Verbraucher eingeschaltet? Zu welchen Zeitpunkten t_3 und t_4 wird er wieder ausgeschaltet?

3.2. Wie groß ist der Endwert des Stroms i_X, wenn der Verbraucher eingeschaltet ist?



3.3. Für die Transistoren T1 und T2 gelten die folgenden technischen Daten: $U_{BE1} = U_{BE2} = 0{,}6V$; $B_1 = 150$; $B_2 = 30$. Wie groß ist der Basisstrom i_{B1} des Transistors T1, wenn der Verbraucher eingeschaltet ist? Welcher maximale Laststrom $i_{x,max}$ würde bei diesem Basisstrom i_{B1} höchstens fließen können?

3.4. Zeichnen die den ungefähren zeitlichen Verlauf der Spannung u_{CE2} ins Oszilloskopbild, der sich bei fehlender Diode D1 ergeben würde. (Gehen Sie zur Vereinfachung davon aus, das alle Bauelemente dies „überleben“ würden!)

Aufgabe 4: Grundlagen (ca. 10 Punkte ≙ 10 Minuten)

4.1. Nennen Sie jeweils einen Vorteil, der sich ergibt, wenn ein Rechner nach der Von-Neumann-Architektur bzw. nach der Harvard-Architektur aufgebaut wird. Welche der beiden Architekturen eignet sich eher für einen Universal-PC, welche für ein Mikrocontroller-basiertes Steuergerät („Embedded System“)?

	Vorteil	Universal-PC / Embedded System?
Von-Neumann		
Harvard		

4.2. Was versteht man im Zusammenhang mit den AVR-Mikrocontrollern, die Sie im Praktikum programmiert haben, unter der „modifizierten“ Harvard-Architektur?

4.3. Wie heißen die drei Busse, die sich auf der Hauptplatine jedes typischen Mikroprozessorsystems befinden? Erläutern Sie in wenigen Stichworten, wozu jeder dieser drei Busse dient.

Aufgabe 5: C-Programmierung auf Mikrocontrollern (ca. 14 Punkte ≙ 14 Minuten)

In einem C-Programm für den Mikrocontroller ATmega328P, der mit 16 MHz Taktfrequenz betrieben wird, soll die Anzahl der Millisekunden seit dem Programmstart ermittelt werden. Dazu wird der Timer/Counter0 so eingerichtet, dass er 1000 Interrupts pro Sekunde auslöst. Die Anzahl der Millisekunden seit dem Programmstart können über die Funktion „Millis“ abgefragt werden.

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

// Globale Variable: Millisekunden seit Programmstart
volatile uint32_t _millis_intern = 0;

ISR(TIMER0_COMPA_vect)  // Interrupt-Frequenz = 1 kHz
{
    _millis_intern++;
}
```

```
// Millisekunden seit Programmstart abfragen
uint32_t Millis(void)
{
    uint32_t result = 0;
    cli(); result = _millis_intern; sei();
    return result;
}

// Timer initialisieren, Interrupt-Frequenz = 1 kHz
void init_timer(void)
{
    TCCR0A = _BV(WGM01);              // CTC-Modus (siehe Tab. 19-9, S. 140)
    TCCR0B = _BV(CS01) + _BV(CS00);   // Prescaler = 64 (Tab. 19-10, S. 142)
    OCR0A  = 249;                     // Vergleichswert für Timer
    TIMSK0 = _BV(OCIE0A);             // Compare Match Interrupt (siehe S. 143)
    sei();                            // Interruptsystem des Controllers ein
}
```

5.1. Warum muss die Variable „_millis_intern“ als „volatile“ deklariert werden?

5.2. Wozu dienen die Befehle „cli()“ und „sei()“ in der Funktion „Millis“?

5.3. Nach welcher Zeit erreicht die Variablem „_millis_intern“ ihren Maximalwert und es kommt zu einem Überlauf?

5.4. Nach welcher Zeit würde es zu einem Überlauf kommen, wenn „_millis_intern“ als uint64_t definiert wäre?

5.5. Mit welcher Frequenz werden die Interrupts ausgelöst, wenn das Register „OCR0A“ auf den Wert 49 (statt auf den Wert 249) gesetzt wird?

5.6. Geben Sie die Ergebnisse der folgenden Operationen im Binärformat (!) an.

```
uint8_t var1 = _BV(0) | _BV(3);
```

var1 =

```
uint8_t var2 = _BV(0) + _BV(3);
```

var2 =

```
uint8_t var3 = 0b11111111 & ~_BV(1);
```

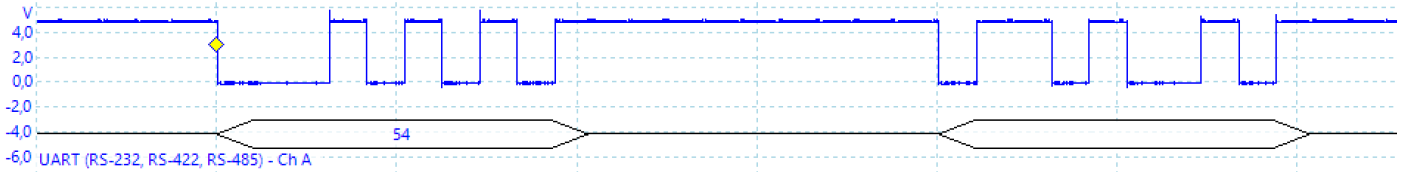
var3 =

```
uint8_t var4 = (1 << 2);
```

var4 =

Aufgabe 6: Serielle Schnittstelle (ca. 10 Punkte ≙ 10 Minuten)

Auf der seriellen Schnittstelle (Anschluss „TXD“) eines Mikrocontrollers wird das folgende Signal ausgegeben:



Es handelt sich um die Übertragung von zwei Datenbytes. Das erste Byte hat den Wert 54_{HEX} (hexadezimal). Der Wert des zweiten Datenbytes soll in dieser Aufgabe ermittelt werden.

Übertragungsparameter: 9,6 kbit/s, 1 Startbit, 1 Stoppbit, kein Paritätsbit

6.1. Markieren Sie in der Abbildung (bei beiden Datenbytes) die Positionen aller Start- und Stoppbits!

6.2. Welchen Wert hat das zweite Datenbyte? (Ergebnis bitte binär und hexadezimal hinschreiben!)

6.3. Nennen Sie zwei Beispiele, wie es zu einem „Framing-Error“ bei der seriellen Datenübertragung kommen kann.