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

WS 2016/17

Viel Erfolg!!

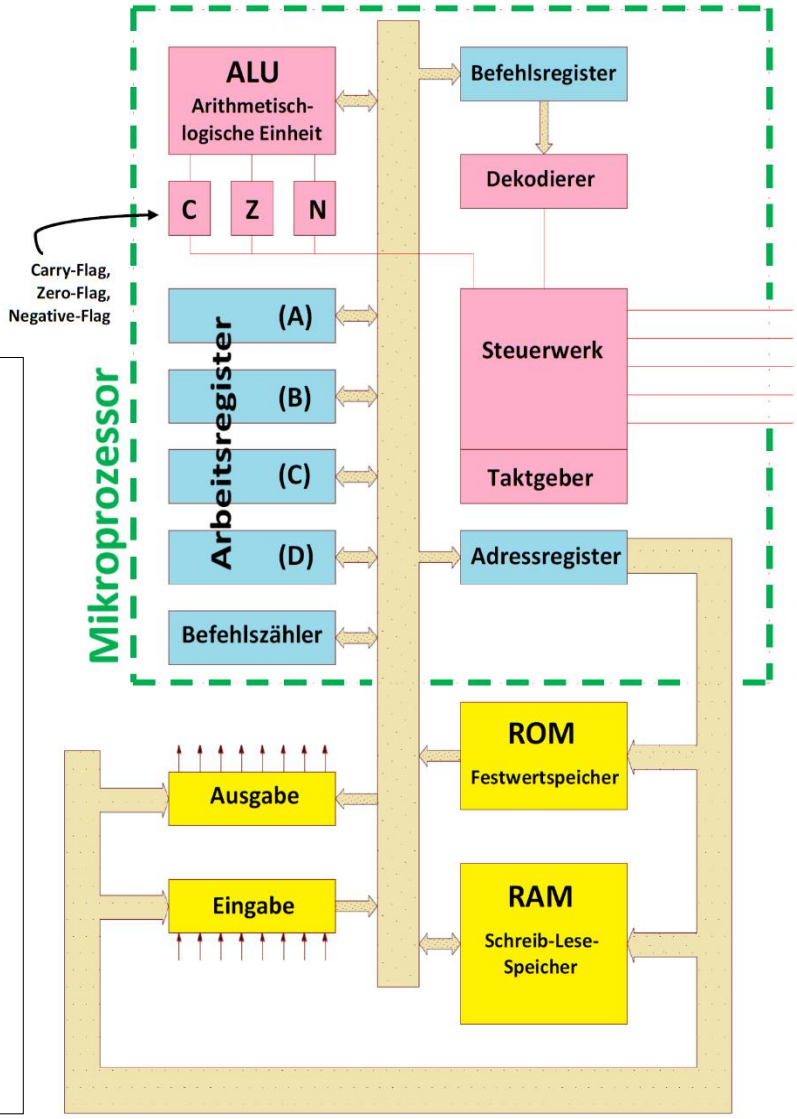
1	2	3	4	5	6		Σ	N

Aufgabe 1: Grundlagen, Mikroprozessorsysteme (ca. 10 Punkte ≙ 10 Minuten)

1.1. Die nebenstehende Abbildung zeigt den (vereinfachten) Aufbau eines typischen Mikroprozessorsystems.

Markieren Sie in der Abbildung den Datenbus und den Adressbus.

1.2. Erläutern Sie in wenigen Stichworten die Aufgaben von Datenbus, Adressbus und Steuerleitungen (Steuerbus). Arbeiten diese Busse unidirektional oder bidirektional?



Datenbus:

Adressbus:

Steuerleitungen (Steuerbus):

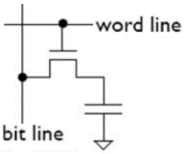
1.3. Erläutern Sie den Unterschied zwischen der von-Neumann-Architektur und der Harvard-Architektur.

1.4. In einem 8-Bit-Prozessor haben viele Register – und insbesondere die Arbeitsregister – eine Größe von 8 Bit. Nur Befehlszähler und Adressregister sind oft deutlich größer (z. B. 16 oder 20 Bit). Warum ist dies so?

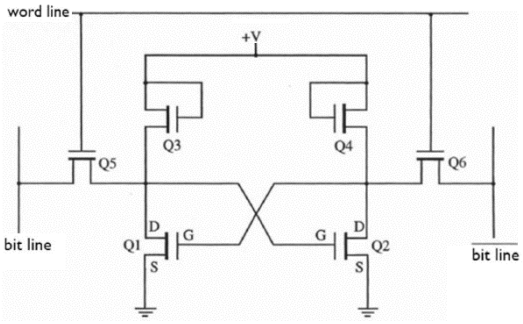
1.5. Wie viele Adressleitungen sind mindestens erforderlich, um einen Speicherbereich von insgesamt 4 GByte eindeutig ansprechen zu können?

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

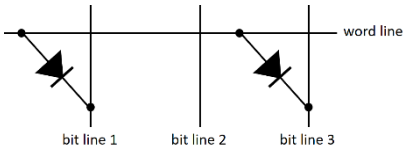
2.1. Betrachten Sie die folgenden Schaltungen von drei unterschiedlichen Speicherzellen. Kreuzen Sie jeweils an, um welchen Typ von Speicher es sich dabei handelt.



❒ EPROM
❒ EEPROM
❒ M-ROM
❒ dynamischer RAM-Speicher
❒ statischer RAM-Speicher



❒ EPROM
❒ EEPROM
❒ M-ROM
❒ dynamischer RAM-Speicher
❒ statischer RAM-Speicher



❒ EPROM
❒ EEPROM
❒ M-ROM
❒ dynamischer RAM-Speicher
❒ statischer RAM-Speicher

2.2. Welche Art von Speicherbausteinen besitzt ein Quarzfenster im Chip-Gehäuse? Wozu dient dieses Fenster?

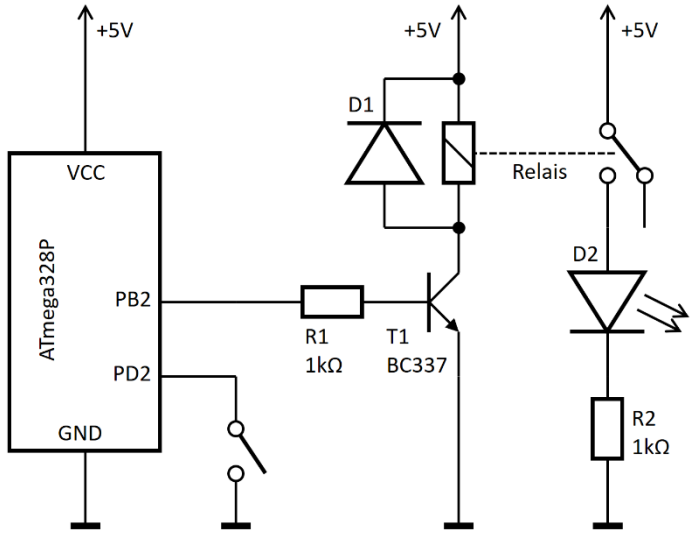
2.3. Wozu benötigen dynamische RAM-Speicherzellen einen regelmäßigen „Refresh-Vorgang“?

2.4. Wozu dient bei der Arbeit mit Mikrocontrollern der sog. Bootloader?

2.5. Beschreiben Sie in Stichworten die Schritte, die bei der Verwendung eines Bootloaders nacheinander ablaufen.

Aufgabe 3: GPIO-Ports (ca. 6 Punkte ≙ 6 Minuten)

3.1. Die Abbildung zeigt den Anschluss eines Relais an einen Mikrocontroller. Berechnen Sie den Strom, der vom Ausgang PB2 geliefert werden muss, um das Relais einzuschalten. (Daten des Transistors: $B = 150, U_{BE} = 0{,}7\ \mathrm{V}$. Die Spannung am eingeschalteten Mikrocontroller-Ausgang PB2 beträgt 4,5 V.)



3.2. Das Relais benötigt zum Einschalten einen Spulenstrom von 50 mA. Ist der Basisstrom des Transistors T1 groß genug, damit dieser Spulenstrom tatsächlich fließen kann? (Begründung oder kurze Rechnung erforderlich!)

3.3. Wozu dient die Diode D1?

Aufgabe 4: C-Programmierung von Mikrocontrollern (ca. 10 Punkte ≙ 10 Minuten)

Betrachten Sie das folgende C-Programm für einen Mikrocontroller des Typs ATmega328P:

```
1.  #define F_CPU 16000000UL
2.  #include <avr/io.h>
3.  #include <util/delay.h>
4.
5.  int main(void)
6.  {
7.      DDRB = 0b00100000;
8.
9.      while (1)
10.     {
11.         PORTB = 0b11111111;
12.         _delay_ms(500);
13.
14.         PORTB = 0b00000000;
15.        _delay_ms(500);
16.     }
17. }
```

4.1. Wozu dient die `#define`-Anweisung in der ersten Zeile? Was passiert, wenn diese Zeile fehlt?

4.2. Wozu dient die Zuweisung `DDRB = 0b00100000;` in Zeile 7?

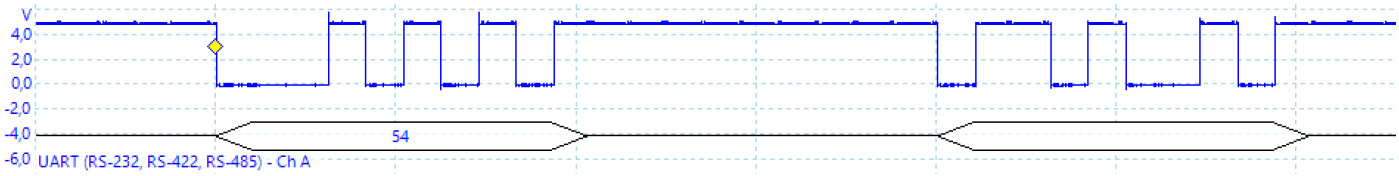
4.3. Beschreiben Sie das Verhalten einer Leuchtdiode, die über einen Vorwiderstand von 1 kΩ an den Anschluss PB5 des Mikrocontrollers angeschlossen ist.

4.4. Beschreiben Sie das Verhalten einer Leuchtdiode, die ohne Vorwiderstand direkt an PB0 angeschlossen ist.

4.5. Regelmäßig wiederkehrende Aufgaben können statt in einer `while`-Schleife mit *`_delay_ms`* -Aufrufen auch durch einen Timer-Interrupt gesteuert werden. Nennen Sie zwei Vorteile, wenn ein Timer-Interrupt verwendet wird.

Aufgabe 5: Serielle Schnittstelle eines Mikrocontrollers (ca. 8 Punkte ≙ 8 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

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

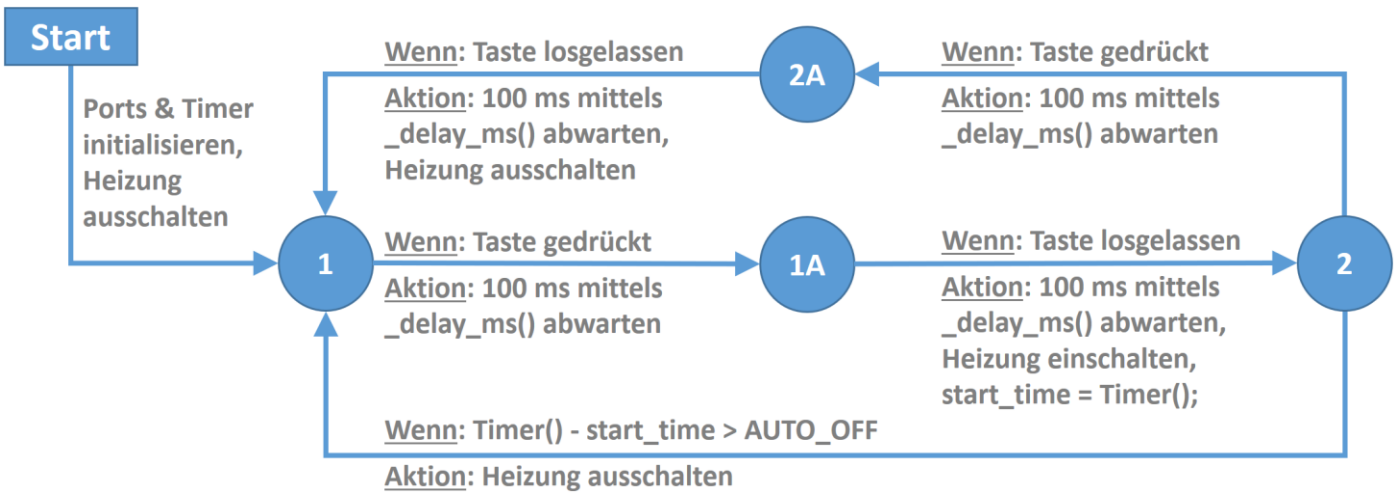
5.2. Welchen Wert hat das zweite Datenbyte? (Ergebnis bitte binär und hexadezimal hinschreiben!)

Aufgabe 6: Programmierung (ca. 23 Punkte ≙ 23 Minuten)

In dieser Aufgabe soll die Software zur Steuerung einer Heckscheibenheizung entwickelt werden.

- Es wird ein ATmega328P-Mikrocontroller verwendet.
- Die Heizung ist (über ein Relais) mit dem Anschluss PB0 verbunden (Schaltung ähnlich wie in Aufgabe 3).
- Zwischen dem Anschluss PB5 und Masse ist ein Taster angeschlossen (ebenfalls ähnlich wie in Aufgabe 3).
- Durch Drücken des Tasters wird die Heizung eingeschaltet.
- Durch erneutes Drücken des Tasters wird die Heizung wieder ausgeschaltet.
- Die Heizung geht spätestens nach 120 Sekunden automatisch wieder aus, auch wenn kein Taster betätigt wird.

Der Ablauf der Steuerung kann durch den folgenden endlichen Automaten beschrieben werden:



6.1. Ergänzen Sie die Funktion `init_timer` so, dass die Interrupt-Funktion `ISR(TIMER0_COMPA_vect)` 100 mal pro Sekunde aufgerufen wird. Also nicht – wie im Praktikum – 1000 mal pro Sekunde! Sie werden feststellen, dass sich für `OCR0A` keine exakte Lösung ergibt. Wählen Sie stattdessen einen möglichst genauen, gerundeten Wert. Beachten Sie die folgenden Ausschnitte aus dem Datenblatt des Controllers:

19.9.1. TC0 Control Register A

Bit	7	6	5	4	3	2	1	0
	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Table 19-9. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

19.9.2. TC0 Control Register B

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Table 19-10. Clock Select Bit Description

CS02	CS01	CS00	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)
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.

19.9.3. TC0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
						OCIEB	OCIEA	TOIE
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCIEB: Timer/Counter0, Output Compare B Match Interrupt Enable
When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCF0B bit is set in TIFR0.

Bit 1 – OCIEA: Timer/Counter0, Output Compare A Match Interrupt Enable
When the OCIE0A bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in TIFR0.

Bit 0 – TOIE: Timer/Counter0, Overflow Interrupt Enable
When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in TIFR0.

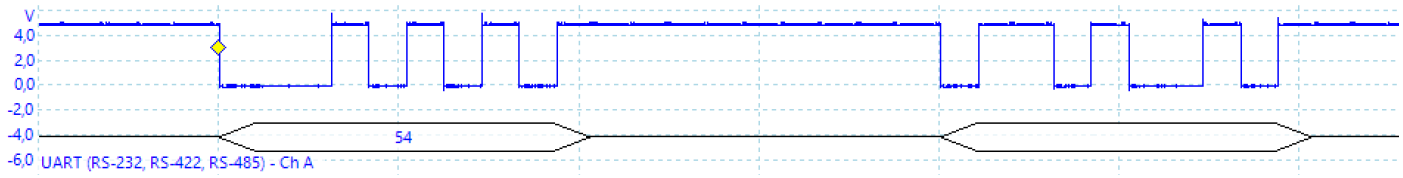
6.2. Ergänzen Sie die Funktion `main` so, dass der Ablauf dem oben dargestellten endlichen Automaten entspricht.

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

// Anschlüsse für Relais und Taster
#define RELAIS PB0
#define TASTER PB5

// Nach wie vielen Hundertstel-Sekunden geht die Heizung automatisch aus?
#define AUTO_OFF 12000UL

// Mögliche Zustände der Heizungssteuerung
#define STATE_1  1
#define STATE_1A 2
#define STATE_2  3
#define STATE_2A 4

// Globale Variable: Hundertstel-Sekunden seit Programmstart
volatile uint32_t _timer_intern = 0;

// Timer-Interrupt, Frequenz = 100 Hz
ISR(TIMER0_COMPA_vect)
{
    _timer_intern++;
}

// Hundertstel-Sekunden seit Programmstart abfragen
uint32_t Timer(void)
{
    uint32_t result = 0;
    cli(); result = _timer_intern; sei();
    return result;
}

// Aufgabe 6.1: Timer initialisieren: Interrupt-Frequenz = 100 Hz
void init_timer(void)
{

    TCCR0A =                                    // CTC-Modus einschalten

    TCCR0B =                                    // Passenden Prescaler einstellen

    OCR0A  =                                    // Vergleichswert für Timer --> gerundet!!!

    TIMSK0 =                                    // Output Compare Match Interrupt aktivieren

                                                // Interruptsystem des Controllers einschalten

}

// Taster (PB5) = Eingang mit Pullup, Relais (PB0) = Ausgang
void init_ports(void)
{
    DDRB = _BV(RELAIS); PORTB = _BV(TASTER);
}

// Relais an- (on_off = 1) oder ausschalten (on_off = 0)
void set_relay(uint8_t on_off)
{
    if(on_off)
        PORTB |= _BV(RELAIS);
    else
        PORTB &= ~_BV(RELAIS);
}
```

```
// Ist der Taster gedrückt (Rückgabe = 1) oder nicht (Rückgabe = 0)?
uint8_t key_pressed(void)
{
    if(PINB & _BV(TASTER))
        return 0;
    else
        return 1;
}

// Aufgabe 6.2: Hauptprogramm
int main(void)
{
    // Alle notwendigen Variablen definieren und ggf. auf Startwerte setzen
    uint32_t start_time = 0;
    uint8_t state = STATE_1;

    // TODO: Ports & Timer-Interrupt initialisieren, Heizung ausschalten









    // TODO: Hauptschleife, Ablauf entspricht dem abgebildeten endlichen Automaten
    while(1)
    {
        switch(state)
        {
        case STATE_1:
            if(key_pressed())
            {
```

(Fortsetzung Aufgabe 6.2)

Aufgabe 7: Praktikum zur seriellen Datenübertragung bei Prof. Höcht (ca. 13 Punkte ≙ 13 Minuten)



7.1. Betrachten Sie erneut den oben abgebildeten Signalverlauf aus Aufgabe 5: Das am „TXD-Ausgang“ des Mikrocontrollers ausgegebene Signal wechselt ständig zwischen 0 Volt und 5 Volt. Eine Datenübertragung über die serielle PC-Schnittstelle (RS-232-Schnittstelle) folgt grundsätzlich demselben Muster, allerdings werden unterschiedliche Spannungen verwendet.

Zeichnen Sie den Spannungsverlauf auf der „TXD-Leitung“ einer seriellen PC-Schnittstelle in das vorbereitete Diagramm. Geben Sie konkrete Spannungswerte an! (Es sollen dieselben Daten übertragen werden.)

7.2. Was versteht man im Zusammenhang mit serieller Datenübertragung unter einem „Overrun-Error“?

7.3. Was versteht man im Zusammenhang mit serieller Datenübertragung unter einem „Framing-Error“?

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

7.5. Beschreiben Sie die einzelnen Schritte, die bei einer seriellen Datenübertragung mit vollständigem Hardware-Handshake (also mit den Leitungen RTS, CTS, DSR, DTR) durchlaufen werden müssen, wenn ein PC ein Kommando an einen Plotter senden soll.