Hochschule München Fakultät 03	Komponenten & Programmierung von Automatisierungssystemen (Teil 1)		Prof. Dr. T. Küpper Prof. Dr. J. Höcht
Zugelassene Hilfsmittel: alle eigenen, Taschenrechner	**Matr.-Nr.:** **Hörsaal:**	**Name, Vorname:** **Unterschrift:**	

Viel Erfolg!!

1	2	3	4	HT	Σ	N

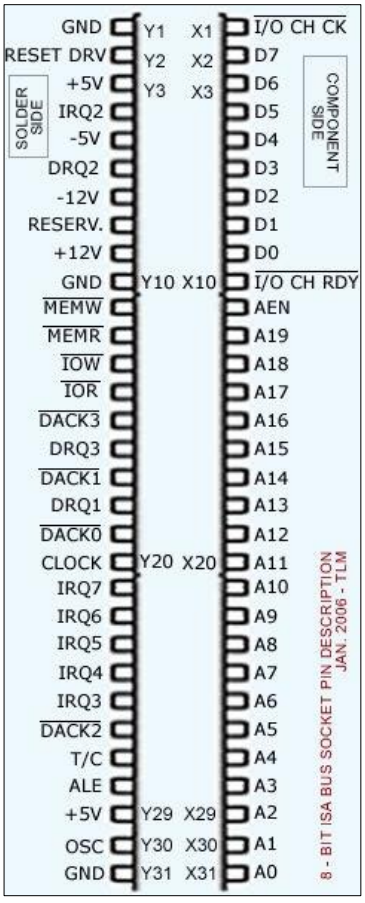
Aufgabe 1: Grundlagen, Mikroprozessorsysteme (ca. 15 Punkte ≙ 15 Minuten)

1.1. 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.

1.2. Die nebenstehende Abbildung zeigt die Kontakte eines „ISA-Steckplatzes“, dem Standard-Steckplatz in PCs der 80er- und 90er-Jahre.

- Wie viele Bits können über den Datenbus parallel übertragen werden? Markieren Sie die Kontakte, die zum Datenbus gehören mit grüner Farbe.

- Wie viele Bits können über den Adressbus parallel übertragen werden? Markieren Sie die Kontakte, die zum Adressbus gehören, mit blauer Farbe.



1.3. Wie viele unterschiedliche Adressen können über einen solchen ISA-Steckplatz eindeutig angesprochen werden? (Ergebnis als Dezimalzahl angeben!)

1.4. Wie viele Adressleitungen sind notwendig, um einen Speicherbereich von insgesamt 1 GByte eindeutig ansprechen zu können?

Aufgabe 2: 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.

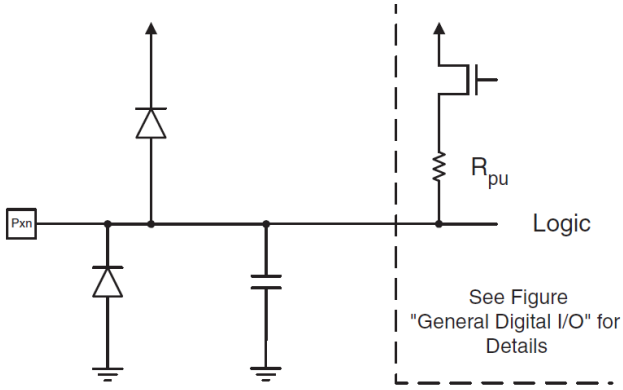
2.2. Nennen Sie einen Vorteil von statischem RAM gegenüber dynamischem RAM.

2.3. Nennen Sie einen Vorteil von dynamischem RAM gegenüber statischem RAM.

2.4. Nennen Sie jeweils einen typischen Einsatzbereich für statisches und dynamisches RAM.

Aufgabe 3: Mikrocontroller, elektrische Eigenschaften (ca. 15 Punkte ≙ 15 Minuten)

3.1. Die Abbildung zeigt, wie ein GPIO-Pin eines typischen Mikrocontrollers intern aufgebaut ist (Quelle: Datenblatt ATtiny45). Wozu dienen die beiden Dioden?



3.2. Wozu dient der Widerstand R_{PU}?

3.3. An einen GPIO-Pin eines mit V_{CC} = 5 V betriebenen Mikrocontrollers soll ein Relais angeschlossen werden. Die Spule zur Ansteuerung des Relais hat einen ohmschen Widerstand von 40 Ω und benötigt zum Einschalten eine Spannung im Bereich von 3,75…9 Volt. Ist es möglich, das Relais direkt über den GPIO-Pin des Mikrocontrollers zuverlässig zu betreiben? Beachten Sie dazu die folgenden Hinweise aus dem Datenblatt des Mikrocontrollers. (Begründung mit Stichworten und/oder kurzer Berechnung!)

Notes:
1. Typical values at 25°C.
2. "Min" means the lowest value where the pin is guaranteed to be read as high.
3. "Max" means the highest value where the pin is guaranteed to be read as low.
4. Although each I/O port can sink more than the test conditions (10 mA at V_{CC} = 5V, 5 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for all ports, should not exceed 60 mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (10 mA at V_{CC} = 5V, 5 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOH, for all ports, should not exceed 60 mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.

3.4. Zeichnen Sie eine Schaltung mit einem NPN-Transistor, die geeignet ist, das o. g. Relais über den GPIO-Pin des Mikrocontrollers zuverlässig ein- und auszuschalten.

3.5. Beim Ausschalten des Relais können in der Relais-Spule hohe Spannungsspitzen induziert werden. Ergänzen Sie – falls noch nicht geschehen – Ihre Schaltung um eine Schutzschaltung, sodass diese Spannungsspitzen keine Schäden verursachen können. Beschreiben Sie in wenigen Stichworten die Funktion der Schutzschaltung.

3.6. Berechnen Sie den Strom, der vom GPIO-Pin geliefert werden muss, um mit Ihrer Schaltung das Relais einzuschalten. Für die Daten des Transistors bitte einfach typische Werte annehmen!

Aufgabe 4: Programmierung (ca. 30 Punkte ≙ 30 Minuten)

Mit einem Mikrocontroller des Typs ATtiny45, der mit einer Taktfrequenz von 8 MHz betrieben wird, soll ein Rechteckgenerator mit den folgenden Funktionen aufgebaut werden:

- Das Rechtecksignal wird am Anschluss PB0 ausgegeben. Es können Frequenzen von 10 kHz, 100 kHz und 1 MHz ausgegeben werden, der Tastgrad („duty cycle“) beträgt immer 50%.
- An PB1, PB2 und PB3 sind über passende Vorwiderstände drei Leuchtdioden (LEDs) angeschlossen. Bei einer Frequenz von 10 kHz leuchtet LED1 an PB1, bei 100 kHz leuchtet LED2 an PB2 und bei 1 MHz leuchtet LED3 an PB3.
- An den Eingang PB4 ist ein Taster angeschlossen: PB4 geht auf LOW, wenn der Taster gedrückt wird, PB4 geht auf HIGH, wenn der Taster wieder losgelassen wird. Direkt nach dem Start wird zunächst eine Frequenz von 10 kHz ausgegeben, nach Betätigung des Tasters 100 kHz, nach erneutem Tastendruck 1 MHz, dann wieder 10 kHz usw…

4.1. Schreiben Sie auf Seite 6 die Funktion **init_port()**, welche die Anschlüsse PB0, PB1, PB2 und PB3 zu Ausgängen und den Anschluss PB4 zu einem Eingang macht.

4.2. Sie haben im Praktikum den „Fast-PWM-Modus“ kennengelernt: Ein Zähler („Timer 0“) zählt immer wieder von 0 bis 255. Beim Zählerstand 0 wird der PWM-Ausgang des Controllers auf HIGH gesetzt, bei Erreichen des in OCR0A angegebenen Werts wechselt der PWM-Ausgang wieder auf LOW. Am PWM-Ausgang ergibt sich ein Rechtecksignal mit konstanter Frequenz, das Tastverhältnis („duty cycle“) ist über das Register OCR0A einstellbar.

Es gibt aber auch einen anderen Fast-PWM-Modus, der wie folgt abläuft: Ein Zähler („Timer 0“) zählt automatisch immer wieder von 0 bis zu dem Wert, der in OCR0A angegeben ist (also nicht bis 255!) und wechselt dann sofort wieder auf 0. Immer wenn der Zähler auf 0 springt, wird der PWM-Ausgang umgeschaltet („getoggelt“). Am PWM-Ausgang ergibt sich ein Rechtecksignal mit einem konstantem Tastgrad von 50%, die Frequenz ist über das Register OCR0A einstellbar.

Dieser „andere“ Fast-PWM-Modus soll hier benutzt werden!

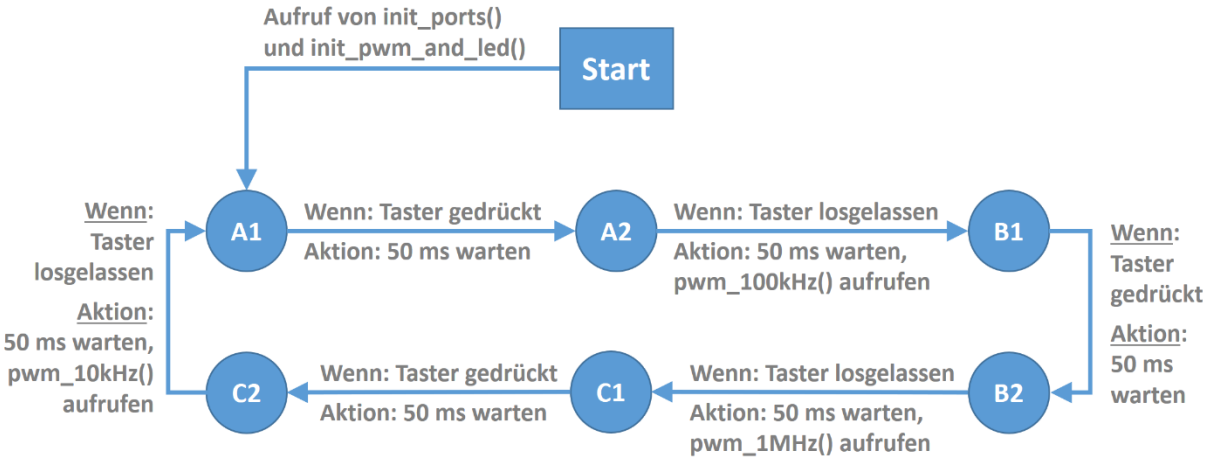
Schreiben Sie auf Seite 6 die Funktion **init_pwm_and_led()**, welche den „anderen“ Fast-PWM-Modus startet, eine Frequenz von 10 kHz ausgibt und die Leuchtdiode LED1 einschaltet.

Tipp: Beachten Sie die Tabellen 11-5 und 11-6 mit den dazugehörenden Erläuterungen im Datenblatt des Mikrocontrollers. Beachten Sie auch den Hinweis zum WGM02-Bit ganz am Ende von Seite 73!

4.3. Nachdem der Fast-PWM-Modus gestartet wurde, kann durch Verändern des Registers OCR0A und ggf. des Prescalers die Frequenz des Rechtecksignals neu eingestellt werden. Implementieren Sie die Funktionen **pwm_100kHz()** zum Einstellen von 100 kHz und Einschalten von LED2, **pwm_1MHz()** zum Einstellen von 1 MHz und Einschalten von LED3 und **pwm_10kHz()** zum erneuten Einstellen von 10 kHz und Einschalten von LED1.

4.4. Vervollständigen Sie auf Seite 8 das Hauptprogramm **main()**, sodass das Verhalten des Rechteckgenerators exakt dem abgebildeten Zustandsautomaten entspricht (siehe Abbildung auf der nächsten Seite).

4.5. Wozu dienen die Verzögerungen von jeweils 50 ms beim Zustandswechsel?




```
/*********************************************************************/
/*** Einfacher Rechteckgenerator (10 kHz, 100 kHz, 1 MHz) mit ATtiny45 ***/
/*********************************************************************/
#define F_CPU 8000000UL
#include <util/delay.h>
#include <avr/io.h>

/* Symbolische Konstanten für die verwendeten Anschlüsse */
#define PWM_OUT 0  /* PB0 */
#define LED1 1     /* PB1 */
#define LED2 2     /* PB2 */
#define LED3 3     /* PB3 */
#define TASTER 4   /* PB4 */

/* Alle möglichen Zustände des Zustandsautomaten */
#define A1 1
#define A2 2
#define B1 3
#define B2 4
#define C1 5
#define C2 6

/* Anschlüsse initialisieren: PB0...PB3 = Ausgänge, PB4 = Eingang */
void init_port(void)
{




}

/* Fast-PWM-Modus aktivieren, 10 kHz einstellen, LED1 einschalten */
void init_pwm_and_led(void)
{




}
```

```
/* 10 kHz einstellen, LED1 einschalten */
void pwm_10kHz(void)
{









}

/* 100 kHz einstellen, LED2 einschalten */
void pwm_100kHz(void)
{









}

/* 1 MHz einstellen, LED3 einschalten */
void pwm_1MHz(void)
{









}

/*****************************************************************/
/*** Hauptprogramm, als Zustandsautomat implementiert          ***/
/*****************************************************************/
int main(void)
{
    /* Verschiedene Initialisierungen... */
    uint8_t state = A1;
    init_port();
    init_pwm_and_led();
```

```
    /* Zustandsautomat... */
    while(1)
    {
        switch(state)
        {
            case A1: if((PINB & _BV(TASTER)) == 0)
                     {
```