Hochschule
München
University of
Applied Sciences

Wintersemester 2020/21

Embedded Systems (Studiengänge MBB, FAB)

Schriftliche Fernprüfung mit Videoaufsicht

Prüfer: Küpper

Bearbeitungszeit: 60 Minuten

Hilfsmittel:

- **Taschenrechner sind zugelassen.**
- **Alle schriftlichen Unterlagen sind erlaubt.**
- **Der PC darf während der Prüfung nur zur Anzeige des Aufgabenblatts genutzt werden.**

Schreiben Sie Ihren Namen, Vornamen und auch die Studiengruppe auf alle Lösungsblätter. Es werden nur handschriftliche Lösungen auf leeren, weißen DIN-A4-Blättern akzeptiert.

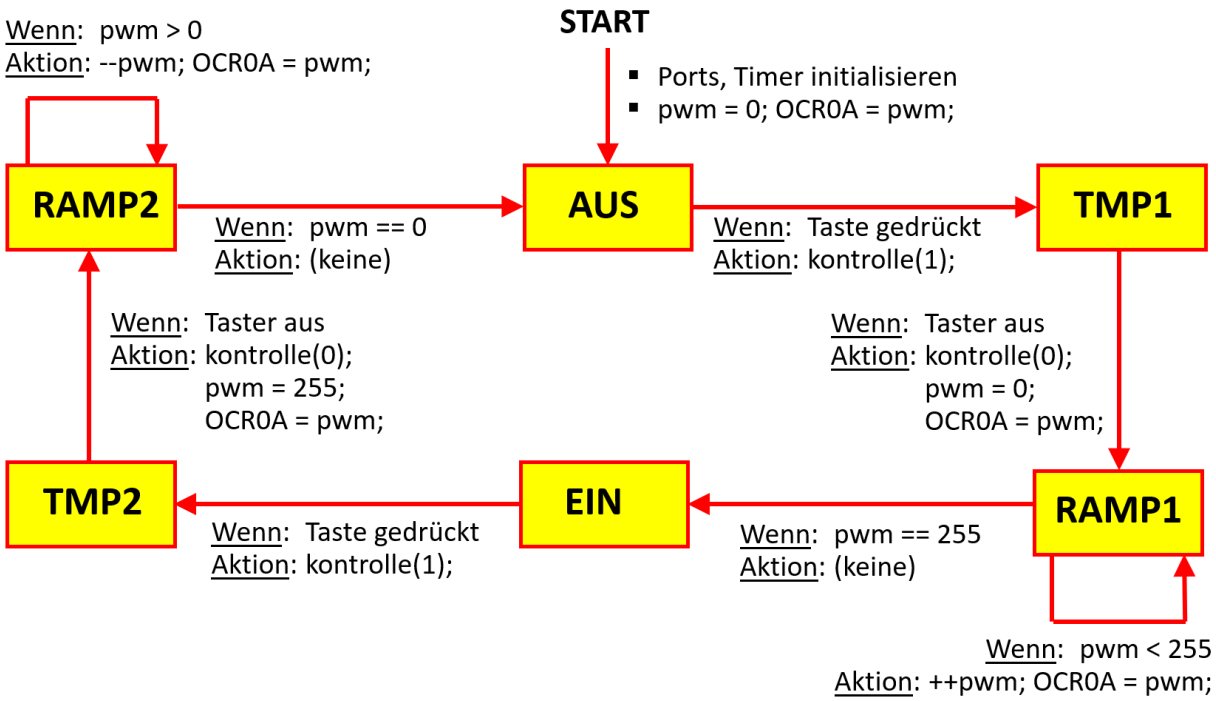
****** Viel Erfolg! ******

Aufgabe 1 von 3 (Mikrocontroller-Programmierung, ca. 30 Punkte ≙ 30 Minuten)

In dieser Aufgabe soll ein Scheinwerfer über einen Tastschalter ein- und ausgeschaltet werden. Das Ein- und Ausschalten des Scheinwerfers soll allerdings nicht schlagartig erfolgen. Vielmehr soll nach der Betätigung des Tastschalters das Licht langsam eingeblendet bzw. wieder ausgeblendet werden. Dazu wird am Anschluss PD6 des Mikrocontrollers ein geeignetes PWM-Signal ausgegeben.

- Es wird ein ATmega328P-Mikrocontroller verwendet und mit dem Atmel Studio 7 programmiert.
- Der Scheinwerfer wird über ein PWM-Signal am **Anschluss PD6** angesteuert.
- Zwischen dem **Anschluss PD2** und Masse ist ein Taster angeschlossen, durch Drücken des Tasters wird der Scheinwerfer ein- und wieder ausgeschaltet. (Wenn der Taster gedrückt wird, dann verbindet er den Anschluss PD2 mit Masse/GND. Es soll der im Mikrocontroller eingebaute Pullup-Widerstand verwendet werden.)
- Am **Anschluss PB2** wird ein zusätzliches „Kontrollsignal“ ausgegeben!

Der Ablauf Mikrocontroller-Programms 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. Schreiben Sie den Quelltext der Funktion `int8_t taster(void)`. Diese Funktion gibt 1 zurück, falls der Taster gerade gedrückt ist, sonst wird 0 zurückgegeben. (Die Funkton soll also nicht warten, bis der Taster wieder losgelassen wird, sondern das Ergebnis sofort zurückgeben.)

1.3. In `void kontrolle(uint8_t on)` gibt es den Befehl `PORTB &= ~_BV(KONTROLLSIGNAL);` Wozu dient dieser Befehl? Warum ist es keine gute Idee, einfach nur `PORTB = 0;` zu schreiben?

1.4. In welchem Modus („Waveform Generation Mode“?) wird der Timer 0 betrieben? Hinweis: Beachten Sie die Tabellen aus dem Datenblatt, die auf der letzten Seite abgedruckt sind!

1.5. Der Scheinwerfer wird durch die Betätigung des Tasters eingeschaltet. Wie lange dauert es nach dem Loslassen des Tasters, bis der Scheinwerfer die maximale Helligkeit erreicht hat?

1.6. Warum würde die Bedienung des Scheinwerfers ohne den Befehl `_delay_ms(10);` im Hauptprogramm nicht korrekt funktionieren?

1.7. Beschreiben Sie kurz in eigenen Worten, wann das Kontrollsignal an PB2 ausgegeben wird.

1.8. Programmieren Sie den fehlenden Rest des Hauptprogramms `int main(void)`.

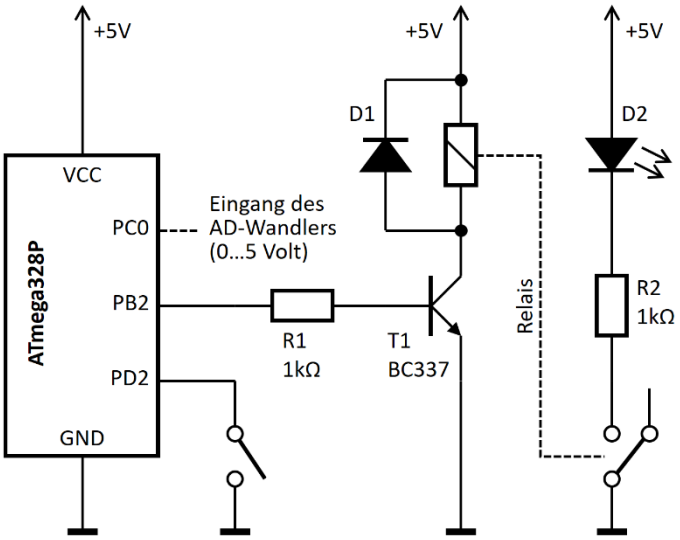
```
#define F_CPU 16000000UL
#include <inttypes.h>
#include <avr/io.h>
#include <util/delay.h>

#define EXTLED PD6          // Scheinwerfer an PD6 (PWM)
#define TASTER PD2          // Taster an PD2
#define KONTROLLSIGNAL PB2  // Kontrollsignal an PB2

// Funktion void init_ports(void); Ein- und Ausgabeports inkl. Pullup initialisieren
*** AUFGABE 1.1 ***

// Funktion int8_t taster(void): Taster gedrückt (1) oder nicht (0)?
*** AUFGABE 1.2 ***

// Kontrollsignal am Anschluss PB2 ein- (1) oder ausschalten (0)
void kontrolle(uint8_t on)
{
    if(on)
        PORTB |= _BV(KONTROLLSIGNAL);
    else
        PORTB &= ~_BV(KONTROLLSIGNAL);
}

// Timer 0 initialisieren
void init_timer0(void)
{
    TCCR0A = _BV(COM0A1) + _BV(WGM00);
    TCCR0B = _BV(CS00) + _BV(CS01);
}

#define Z_AUS   1
#define Z_TMP1  2
#define Z_RAMP1 3
#define Z_EIN   4
#define Z_TMP2  5
#define Z_RAMP2 6

// Hauptprogramm
int main(void)
{
    init_ports();
    init_timer0();

    int8_t state = Z_AUS;
    uint8_t pwm = 0; OCR0A = pwm;

    while(1)
    {
        _delay_ms(10);

        *** AUFGABE 1.8 ***
```

Aufgabe 2 von 3 (Hardware, GPIO-Ports, ca. 10 Punkte ≙ 10 Minuten)

2.1. Die nebenstehende Abbildung zeigt den Anschluss eines Relais an den Ausgang eines Mikrocontrollers.

Was würde passieren, wenn beim Aufbau dieser Schaltung aus Versehen die Freilaufdiode falsch herum (verpolt) eingebaut würde (kurze Erläuterung/Begründung erforderlich)?

2.2. Warum ist es in der Regel nicht möglich, ein Relais direkt mit dem Mikrocontroller-Ausgang zu betreiben (also ohne Transistor)?



2.3. Wird die Freilaufdiode D1 nur beim Einschalten des Relais, nur beim Ausschalten des Relais oder in beiden Fällen benötigt (kurze Erläuterung/Begründung erforderlich)?

2.4. Ist die Freilaufdiode notwendig, wenn anstelle des Relais eine Leuchtdiode (LED) durch den Transistor T1 geschaltet werden soll (kurze Erläuterung/Begründung erforderlich)?

2.5. Beschreiben Sie kurz die Funktion eines „Pullup-Widerstands“ in eigenen Worten.

Aufgabe 3 von 3 (C-Programmierung auf Mikrocontrollern, ca. 10 Punkte ≙ 10 Minuten)

3.1. Was ist die Bedeutung des Schlüsselworts `volatile`; wann bzw. wozu wird es benötigt?

3.2. Zur Ermittlung der Programmlaufzeit seit dem Start des Mikrocontrollers haben wir die folgende Funktion geschrieben:

```
uint32_t Millis(void)
{
    uint32_t result = 0;
    cli(); result = _millis_intern; sei();
    return result;
}
```

Wozu sind sie beiden Befehle `cli();` und `sei();` notwendig (kurze Erläuterung/Begründung erforderlich)?

3.3. Geben Sie zwei Beispiele für die Datentypen, die in der Include-Datei `#include <inttypes.h>` definiert sind (bitte jeweils mit einer kurzen Beschreibung des Datentyps).

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

a) 0b00000010 | 0b00000010 = ?

b) 0b00000010 + 0b00000010 = ?

c) _BV(3) | _BV(4) = ?

d) _BV(3) + _BV(4) = ?

19.9.1. TC0 Control Register A

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: TCCR0A
Offset: 0x44
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x24

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(1)(2)
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

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: TCCR0B
Offset: 0x45
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x25

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)