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

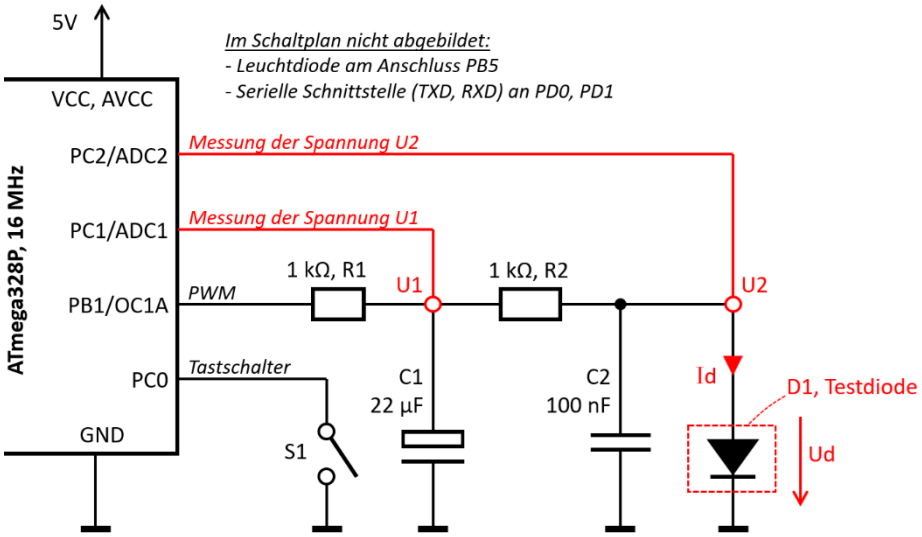
WiSe 2024/25
Viel Erfolg!!

1	2	3	4	5	6		Σ	N

Aufgabe 1 von 5: Programmierung (ca. 36 Punkte ≙ 36 Minuten)

Die folgende Mikrocontrollerschaltung (ATmega328P, 16 MHz) dient zur Messung von Diodenkennlinien. Das Signal am PWM-Ausgang des Mikrocontrollers wird durch einen Tiefpass geglättet (R1, C1) und über R2 an die zu prüfende Diode D1 weitergeleitet. U1 und U2 werden mit dem Analog-Digital-Wandler des Mikrocontrollers erfasst.

- Für die Diodenspannung gilt: $U_\mathrm{d} = U_2$
- Der Diodenstrom kann über das Ohmsche Gesetz ermittelt werden: $I_\mathrm{d} = (U_1 - U_2) \,/\, R_2$



Die Messung wird durch Betätigung des Tastschalters S1 gestartet. Dieser Tastschalter ist so angeschlossen, dass bei gedrücktem Taster an PC0 eine Spannung von 0 Volt anliegt. Wenn der Tastschalter dagegen nicht gedrückt ist, liegt an PC0 eine Spannung von 5 Volt an (Pullup-Widerstand an PC0 muss aktiviert sein, wie im Praktikum!).

1.1 Wozu dient die Definition von F_CPU? Was passiert, wenn hier ein falscher Wert angegeben wird?

1.2 **Der Timer/Counter1 ist ein 16-Bit-Zähler**. In welchem Modus wird der Timer/Counter1 hier betrieben? Bis zu welchem Maximalwert zählt der Timer/Counter1 in diesem Modus? Welche Frequenz wird am PWM-Ausgang des Mikrocontrollers ausgegeben?

```c
#define F_CPU 16000000UL     /***  Aufgabe 1.1  ***/
#include <util/delay.h>
#include <avr/io.h>
#include <stdint.h>
#include <stdio.h>

// USART initialisieren, vergl. Datenblatt, 24.6., S. 230
#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1
void USART_Init(uint16_t ubrr)
{
	UBRR0  = ubrr;
	UCSR0B = _BV( RXEN0) + _BV( TXEN0);
	UCSR0C = _BV(UCSZ00) + _BV(UCSZ01);
}

// Ein einzelnes Zeichen senden, vergl. Datenblatt, 24.7., S. 231
void USART_Transmit(uint8_t data)
{
	while(!(UCSR0A & _BV(UDRE0))) { }
	UDR0 = data;
}

// Ein einzelnes Zeichen empfangen, vergl. Datenblatt, 24.8., S. 233
uint8_t USART_Receive(void)
{
	while (!(UCSR0A & _BV(RXC0))) { }
	return UDR0;
}

// Eine komplette Textzeile inkl. Zeilenumbruch senden.
void USART_Line(const char *text)
{
	uint16_t idx;
	for(idx = 0; text[idx] != 0; idx++) { USART_Transmit(text[idx]); }
	USART_Transmit(13); USART_Transmit(10);  // Zeilenumbruch
}

// PWM-Signal an PB1/OC1A initialisieren          /***  Aufgabe 1.2  ***/
void init_pwm(void)
{
	TCCR1A = _BV(COM1A1) | _BV(WGM10) | _BV(WGM11);
	TCCR1B = _BV(CS10) | _BV(WGM12);
}

// AD-Wandler initialisieren
void init_adc(void)
{
	ADCSRA = _BV(ADEN) | _BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0);      /***  Aufgabe 1.3  ***/
	ADMUX = _BV(REFS0);
}

// Ports initialisieren: PB1, PB5 = Ausgänge (PWM, LED); PC0 = Taster mit Pullup  /***  Aufgabe 1.4  ***/
void init_ports(void)
{




}

// Ist der Taster aktuell gedrückt (Rückgabe 1) oder nicht (Rückgabe 0)?      /***  Aufgabe 1.5  ***/
uint8_t taster(void)
{




}
```

```
// LED an PB5 einschalten (onoff == 1) oder ausschalten (onoff == 0)    /***  Aufgabe 1.6  ***/
void set_led(uint8_t onoff)
{






}

// Analogwert von ADC1 (falls nr == 1) bzw. ADC2 (falls n2 == 2) einlesen.    /***  Aufgabe 1.7  ***/
uint16_t read_adc(uint8_t nr)
{
	if(nr != 1 && nr != 2) { return 0; }  // ... ungültig!

	// ADMUX-Register belegen: Eingang ADC1 bzw. ADC2 auswählen















	// ADSC-Bit setzen, um Analog-Digital-Wandlung zu starten.






	// Warten, bis die Wandlung beendet ist.






	// Messwert zurückgeben.
	return ADC;  // uint16_t
}

// Aus den Ergebnissen der Analog-Digital-Wandlung (ADC1 und ADC2) werden
// die Diodenspannung ux und der Diodenstrom ix berechnet. Beide Werte werden
// auf der seriellen Schnittstelle ausgegeben.
void write_results(double adc1, double adc2)
{
	double avcc = 4.75;  // tatsächliche Betriebsspannung ist etwas kleiner als 5.00 Volt
	double r2 = 994.0;   // 1000-Ohm-Widerstand, mit Multimeter genau gemessen

	double u1 = avcc * adc1 / 1024.0;
	double u2 = avcc * adc2 / 1024.0;
	double ux = u2;                          // Diodenspannung in Volt
	double ix = 1000.0 * (u1 - u2) / r2;   // Diodenstrom in mA

	static char buf[32];
	sprintf(buf, "%5.3f %8.3f", ux, ix);
	USART_Line(buf);
}
```

```c
#define STATE_INIT  0
#define STATE_LOOP1 1
#define STATE_LOOP2 2
#define STATE_NEXT  3
#define STATE_DELAY 4

// Hauptprogramm
int main(void)
{
	double adc1 = 0, adc2 = 0;
	uint16_t count = 0, sum = 0;
	uint8_t state = STATE_INIT;
	OCR1A = 0;

	init_adc();
	init_pwm();
	init_ports();
	USART_Init(MYUBRR);

	while (1)
	{
		_delay_us(500);  // Achtung: 500 Mikrosekunden, nicht Millisekunden!

		switch(state)		/***  Aufgabe 1.8  ***/
		{
```

1.3 Bei der Initialisierung des Analog-Digital-Wandlers (ADC) werden die ADPS2-, ADPS1- und ADPS0-Bits gesetzt. Wozu dienen diese Bits? Warum ist es sinnvoll, sie genau auf diese Werte zu setzen?

1.4 Schreiben Sie den Quelltext der Funktion void init_ports(void) zur Initialisierung der Ein- und Ausgänge.

1.5 Wie lauten die fehlenden Befehle in der Funktion taster()? Diese Funktion gibt eine 1 zurück, falls der Tastschalter gerade eingeschaltet (gedrückt) ist, ansonsten ist die Rückgabe gleich 0. Die Funktion soll also nicht warten, bis der Tastschalter wieder losgelassen wird, sondern das Ergebnis sofort zurückgeben.

1.6 Schreiben Sie den Quelltext der Funktion void set_led(int on_off) zum Schalten der Leuchtdiode (LED) an PB5.

1.7 Die Funktion read_adc(nr) liest einen Analogwert von PC1/ADC1 (falls nr == 1) bzw. von PC2/ADC2 (falls nr == 2) und gibt das Ergebnis des AD-Wandlers als Rückgabewert zurück (10-Bit-Wert, Bereich 0 … 1023). Ergänzen Sie die fehlenden Befehle in dieser Funktion.

1.8 Der Ablauf des Messvorgangs wird durch einen endlichen Automaten beschrieben (siehe letzte Seite!). Ergänzen Sie das Hauptprogramm main() so, dass der Ablauf des Programms diesem endlichen Automaten entspricht.

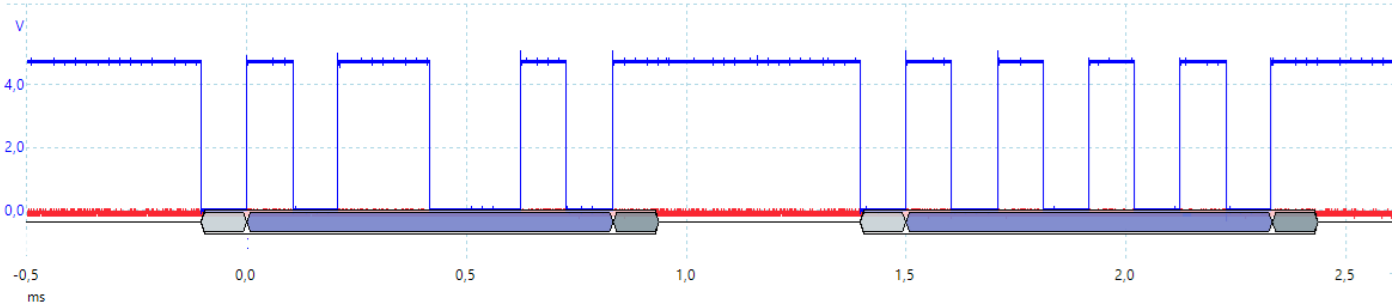
1.9 Das vom Mikrocontroller ausgegebene PWM-Signal wird durch ein Tiefpassfilter geglättet (Kondensator C1, Widerstand R1). Berechnen Sie die Eckfrequenz f_0 dieses Filters. Es gilt: $f_0 = 1 \,/\, (2\pi \cdot R \cdot C)$

1.10 Ermitteln Sie mithilfe des endlichen Automaten die Dauer eines vollständigen Messvorgangs (vom Drücken des Tastschalters bis zum Ende der Messung)? Gehen Sie davon aus, dass die Verzögerungen durch den _delay_us()-Befehl zu Beginn der Hauptschleife so groß sind, dass dagegen alle anderen Verzögerungen vernachlässigbar klein sind (z. B. die AD-Wandlungen).

1.11 Schauen Sie sich die Oszilloskop-Aufnahme eines vollständigen Messvorgangs an (siehe letzte Seite!). Wie lange dauert der Messvorgang tatsächlich?

Aufgabe 2 von 5: Serielle Schnittstelle (ca. 10 Punkte ≙ 10 Minuten)

Ein Mikrocontroller des Typs ATmega328P sendet zwei Zeichen über seine serielle Schnittstelle. Der dazugehörige Spannungsverlauf auf der TXD-Leitung ist in der folgenden Abbildung dargestellt:



2.1 Bestimmen Sie die beiden Zeichen (im ASCII-Format) die vom Mikrocontroller gesendet wurden. Geben Sie für beide Zeichen sowohl den Zeichen-Code als Hexadezimalzahl als auch das dazugehörige ASCII-Zeichen an. Die Parameter der seriellen Schnittstelle sind folgendermaßen gewählt:

- Baudrate: 9600 bps
- 1 Startbit, 8 Datenbits, 1 Stoppbit

2.2 Messdaten im Umfang von 1000 Bytes werden über die serielle Schnittstelle gesendet (Daten der Schnittstelle siehe 2.1). Wie lange dauert diese Übertragung mindestens? (Bitte kurze Berechnung durchführen!)

ASCII-Tabelle

DEZ	HEX	CHAR	DEZ	HEX	CHAR	DEZ	HEX	CHAR	DEZ	HEX	CHAR	DEZ	HEX	CHAR	DEZ	HEX	CHAR	DEZ	HEX	CHAR	DEZ	HEX	CHAR
32	20		33	21	!	34	22	"	35	23	#	36	24	$	37	25	%	38	26	&	39	27	'
40	28	(	41	29	)	42	2A	*	43	2B	+	44	2C	,	45	2D	-	46	2E	.	47	2F	/
48	30	0	49	31	1	50	32	2	51	33	3	52	34	4	53	35	5	54	36	6	55	37	7
56	38	8	57	39	9	58	3A	:	59	3B	;	60	3C	<	61	3D	=	62	3E	>	63	3F	?
64	40	@	65	41	A	66	42	B	67	43	C	68	44	D	69	45	E	70	46	F	71	47	G
72	48	H	73	49	I	74	4A	J	75	4B	K	76	4C	L	77	4D	M	78	4E	N	79	4F	O
80	50	P	81	51	Q	82	52	R	83	53	S	84	54	T	85	55	U	86	56	V	87	57	W
88	58	X	89	59	Y	90	5A	Z	91	5B	[	92	5C	\	93	5D	]	94	5E	^	95	5F	_
96	60	`	97	61	a	98	62	b	99	63	c	100	64	d	101	65	e	102	66	f	103	67	g
104	68	h	105	69	i	106	6A	j	107	6B	k	108	6C	l	109	6D	m	110	6E	n	111	6F	o
112	70	p	113	71	q	114	72	r	115	73	s	116	74	t	117	75	u	118	76	v	119	77	w
120	78	x	121	79	y	122	7A	z	123	7B	{	124	7C	|	125	7D	}	126	7E	~	127	7F	

Aufgabe 3 von 5: C-Programmierung mit Mikrocontrollern (ca. 14 Punkte ≙ 14 Minuten)

3.1 Wie lauten die Ergebnisse der folgenden Berechnungen mit 8-Bit-Zahlen? Ergebnisse als Binärzahl schreiben!

a) 0b01010101 & 0b00111100 =

b) 0b11001100 | 0b10101010 =

c) 0b11110000 << 2 =

d) 0b00111011 >> 3 =

3.2 Ein 16-Bit-System addiert zwei Zahlen: 0xFFEF + 0x0012. Wie lautet das 16-Bit-Ergebnis als Hexadezimal-Zahl? Markieren Sie, ob ein Überlauf auftritt (Hinweis: Alle Register, inkl. Ergebnisregister, sind 16 Bit groß.)

3.3 Ein Mikrocontroller speichert Daten in einem EEPROM. Warum ist es wichtig, die Anzahl der Schreibzyklen zu minimieren?

3.4 Ein 8-Bit-Mikrocontroller verwendet die Variable zeit_ms im Hauptprogramm und auch im Timer-Interrupt:

```
volatile uint16_t zeit_ms = 0;

ISR(TIMER1_COMPA_vect)  // Timer-Interrupt-Routine
{
    zeit_ms++;
}

void main()  // Hauptprogramm
{
    uint16_t aktuelle_zeit;

    while(1)
    {
        // Lese aktuelle Zeit
        aktuelle_zeit = zeit_ms;

        // Weitere Verarbeitung der aktuellen Zeit ...
    }
}
```

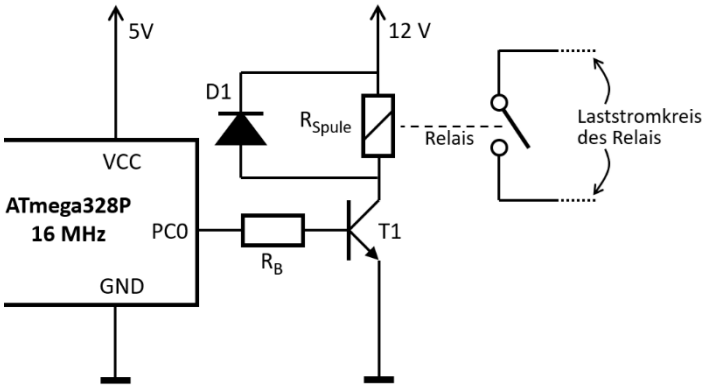
a) Beschreiben Sie, warum die Variable zeit_ms als volatile deklariert sein muss.

b) Erklären Sie, welche Probleme auftreten könnten, wenn die Interrupts während des Lesezugriffs im Hauptprogramm nicht deaktiviert werden.

c) Ergänzen Sie das Hauptprogramm so, dass die Interrupts während des Lesezugriffs sicher deaktiviert werden.

Aufgabe 4 von 5: Ansteuerung von Transistoren und Relais (ca. 10 Punkte ≙ 10 Minuten)

Am Ausgang PC0 eines Mikrocontrollers ist ein Relais angeschlossen. Zur Ansteuerung der Relais-Spule wird ein NPN-Transistor und eine separate 12-Volt-Spannungsquelle verwendet. Die Spannung am Ausgang PC0 beträgt 4,5 V wenn dieser eingeschaltet ist. Für die Basis-Emitter-Spannung des Transistors kann in diesem Fall $U_{\mathrm{BE}} = 0{,}5\ \mathrm{V}$ angenommen werden.



4.1 Berechnen Sie einen geeigneten Vorwiderstand für die Basis des Transistors, wenn der Spulenstrom $I_{\mathrm{Spule}} = 100\ \mathrm{mA}$ beträgt und die Großsignalverstärkung des Transistors $B = 150$ ist. Wählen Sie dabei zum sicheren Einschalten des Transistors einen sinnvollen Übersteuerungsfaktor.

4.2 Überprüfen Sie mit einer kurzen Berechnung, ob ein Vorwiderstand von $R_{\mathrm{B}} = 4{,}7\ \mathrm{k\Omega}$ zwischen dem Ausgang des Mikrocontrollers und der Basis des Transistors geeignet ist, um I_{B} bereitzustellen.

4.3 Nun soll ein anderes Relais mit $R_{\mathrm{Spule}} = 0{,}6\ \mathrm{k\Omega}$ verwendet werden. Überprüfen Sie, ob die Schaltung mit dem Vorwiderstand $R_{\mathrm{B}} = 4{,}7\ \mathrm{k\Omega}$ aus Unterpunkt 4.2 den neuen Anforderungen genügt.

4.4 Im Laststromkreis des Relais wird nun eine LED angeschlossen. Diskutieren Sie, ob es mit dieser Schaltung möglich ist, die LED-Helligkeit per PWM einzustellen.

4.5 Wann bzw. wozu werden „Freilaufdioden“ verwendet?

Aufgabe 5 von 5: Verschiedenes (ca. 10 Punkte ≙ 10 Minuten)

5.1 Was ist der Unterschied zwischen einem Mikroprozessor und einem Mikrocontroller? Nennen Sie zwei wesentliche Merkmale, die sie voneinander abgrenzen.

5.2 Warum ist es einfacher, den RAM eines Mikrocontrollers mit einem externen SRAM-Baustein zu erweitern, anstatt einen DRAM-Baustein zu verwenden?

5.3 Wie erkennt ein externer SRAM-Baustein, ob der Mikrocontroller an einer angegebenen Speicheradresse Daten lesen oder schreiben möchte?

5.4 Wozu dient der Bootloader auf einem Mikrocontroller?

5.5 Nennen Sie in kurzen Stichpunkten die Abläufe, die beim Einsatz eines Bootloaders Schritt für Schritt stattfinden.

Auszüge aus dem ATmega328P-Datenblatt

Timer/Counter1

20.14.1. TC1 Control Register A (TCCR1A)

Bit	7	6	5	4	3	2	1	0
	COM1A[1:0]		COM1B[1:0]				WGM1[1:0]	
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 (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 20-4. Compare Output Mode, Fast PWM

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 20-7. 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)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

ADC - Analog to Digital Converter

28.9.2. ADC Control and Status Register A (ADCSRA)

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – ADEN: ADC Enable

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

Bit 6 – ADSC: ADC Start Conversion

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

Table 28-5. ADPSn: ADC Prescaler Select [n = 2:0]

ADPS[2:0]	Division Factor
000	2
001	2
010	4
011	8
100	16
101	32
110	64
111	128

By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is selected by the ADC Prescaler Select bits in the ADC Control and Status Register A (ADCSRA.ADPS).

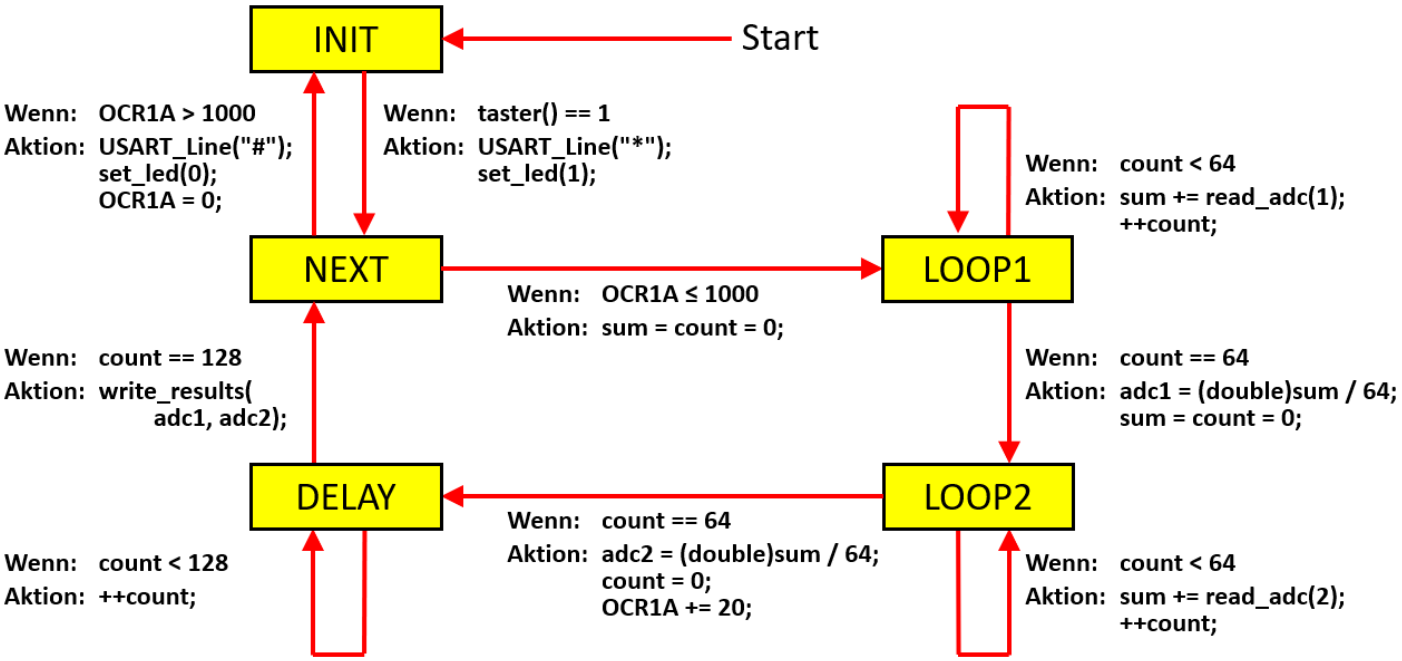
28.9.1. ADC Multiplexer Selection Register (ADMUX)

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

Table 28-4. Input Channel Selection

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
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	1.1V (V_{BG})
1111	0V (GND)

<u>zu Aufgabe 1</u>: Endlicher Automat



<u>zu Aufgabe 1</u>: Oszilloskop-Aufnahme eines vollständigen Messvorgangs, es ist der zeitliche Verlauf der Diodenspannung U_d dargestellt.