Fachhochschule Gelsenkirchen
Physikalische Technik

Arbeitsunterlagen zum Praktikum Mikroprozessortechnik II

Versuch 2: AD/DA-Wandlung

Inhaltsverzeichnis

Seite

Abbildungsverzeichnis

1 Einleitung

In diesem Versuch bilden das Einlesen von analogen Spannungen und die Ausgabe von Analogwerten den Schwerpunkt.

Die Versuchsaufgabe besteht darin, schrittweise einen Funktionsgenerator aufzubauen. Es soll eine Sägezahn-, eine Sinus- und eine Rechteckfunktion ausgegeben werden.
Die Ausgangsfunktion soll dadurch entstehen, dass im EEPROM-Bereich gespeicherte Funktionswerte periodisch als pulsweitenmodulierte Ausgangsspannung ausgegeben werden. Dazu sollen jeweils 16 Stützpunkte auf 16 aufeinander folgenden EEPROM-Adressen abgelegt sein und die Stützpunkte für die ausgewählte Funktion bilden. Zur Glättung des PWM-Signals wird ein RC-Tiefpass an den Ausgangsport angeschlossen.
Die Auswahl der Funktion geschieht durch Betätigung der an einen Digitaleingang angeschlossenen Taster. Die Frequenz der ausgegebenen Funktion soll mit Hilfe einer analogen Eingangsspannung über ein Potentiometer einstellbar sein. Die Frequenz soll dadurch geändert werden, dass die Kurvenstützpunkte bei kleiner Eingangsspannung schnell hintereinander und bei hoher Spannung langsamer ausgegeben werden.

Im Folgenden wird erklärt, wie man mit dem ATmega16:

- analoge Eingangsspannungen einliest,
- analoge Spannungen mit Hilfe der Pulsweitenmodulation ausgibt und
- Werte im EEPROM-Speicherbereich ablegt und während des Programmablaufs ausliest.

Entsprechende Codebeispiele werden angegeben.

2 AD-Umsetzung mit dem ATmega16

Im ATmega16 ist ein 10-Bit Analog/Digital Konverter eingebettet, die Umsetzung erfolgt nach dem Prinzip der sukzessiven Approximation. Der Eingang des AD - Umsetzers ist mit einem 8-Kanal Analog-Multiplexer verbunden, so dass man acht verschiedene Eingangspotenziale einlesen kann. Das Bezugspotenzial aller Eingänge ist 0V (GND). Die Eingänge des Multiplexers sind mit dem Port A verbunden.

Der DA-Umsetzer ist zwischen zwei Betriebsarten umstellbar. Die erste Betriebsart führt nach der Aktivierung ein einmaliges Einlesen einer Spannung durch (Single Conversion). In der zweiten Betriebsart werden nach einmaliger Aktivierung fortwährend Werte eingelesen (Free Running mode).

Zunächst wird die Funktionsweise anhand des Blockschaltbildes erklärt. Bevor auf einige Details näher eingegangen wird und die bauteilspezifischen I/O-Register und Bits erklärt werden

2.1 Blockschaltbild des AD-Umsetzers

Das Blockschaltbild des ADC ist in Abbildung 1 zu sehen. Der ADC wird von einer separaten Eingangsspannung am Pin AV_{CC} versorgt. AV_{CC} darf nicht mehr als ± 0,3 V von der Versorgungsspannung des Atmega16 abweichen. Die Referenzspannungen von wahlweise 2,56 V oder AV_{CC} werden intern bereitgestellt. Außerdem kann eine externe Referenzspannung angeschlossen werden. Die Referenzspannung wird durch die Bits REFSn im Register ADMUX (Kap. 2.7.1) ausgewählt.

Der ADC setzt die analoge Eingangsspannung in einen 10-Bit Digitalwert durch sukzessive Approximation um.

Der analoge Eingangskanal wird durch Setzen der entsprechenden Bits MUXn im Register ADMUX ausgewählt. Jeder der ADC Eingangsanschlüsse, genauso wie GND kann als Eingangssignal ausgewählt werden. Außerdem ist es möglich, die Differenzspannung zwischen bestimmten Eingangskanälen als Eingang auszuwählen oder die Eingangsspannung zu verstärken (siehe Tabelle 2). Der ADC wird durch Setzen des „ADC Enable Bit“, ADEN im Register ADCSR, aktiviert (Kap. 2.7.2). Spannungsreferenz- und Eingangskanalauswahl bleiben ohne Auswirkung bis ADEN gesetzt wird.

Der ADC beinhaltet eine „Sample and Hold“ Schaltung, die sicherstellt, dass die Eingangsspannung während der AD-Umsetzung konstant bleibt..

Der ADC erzeugt ein 10-Bit Ergebnis, welches in den ADC Datenregistern, ADCH und ADCL, abzurufen ist. In der Regel werden die zehn Bit im insgesamt 16-Bit langen Doppelregister rechtsbündig angeordnet, aber es kann auch durch das Setzen des Bit ADLAR im Register ADMUX linksbündig dargestellt werden. Wenn das Ergebnis linksbündig angeordnet und eine Auflösung von mehr als 8 Bit nicht

erforderlich ist, reicht es aus, das Register ADCH auszulesen. Andernfalls muss zunächst das Register ADCL und anschließend das Register ADCH ausgelesen werden. Nur dann ist sichergestellt, dass die Inhalte der Datenregister zu derselben Umsetzung gehören. Wenn das Register ADCL einmal ausgelesen wurde, bleibt der Zugriff zu den Datenregistern für den Konverter blockiert. Das bedeutet, dass, wenn das Register ADCL ausgelesen wurde und eine Umsetzung abgeschlossen ist, bevor das Register ADCH ausgelesen wird, keines der beiden Register aktualisiert wird und das Ergebnis der Umsetzung verloren geht. Erst wenn das Register ADCH ausgelesen wurde, wird der ADC-Zugriff auf die Register ADCH und ADCL wieder freigegeben.

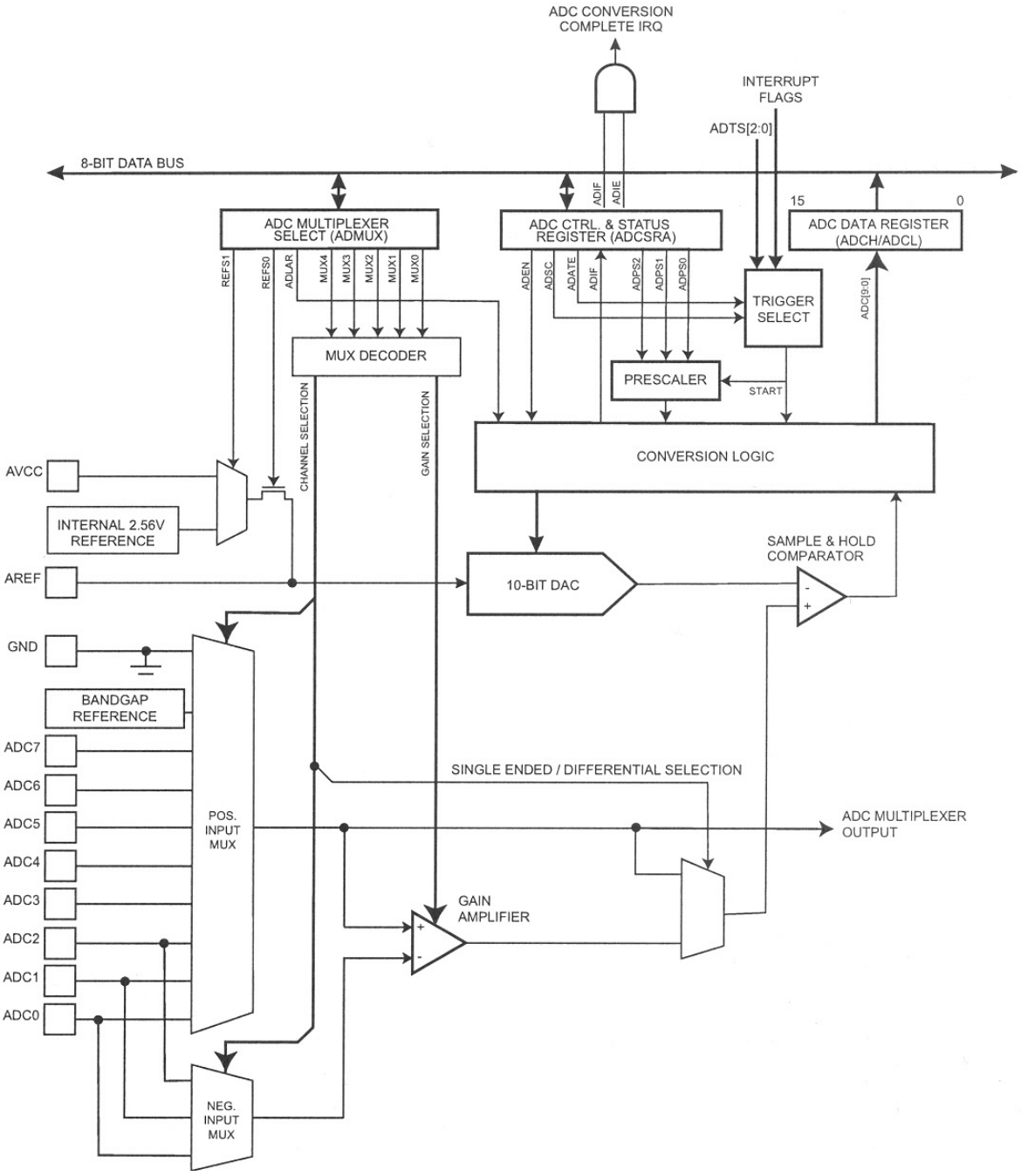

Abbildung 1: Blockschaltbild des AD-Umsetzers

Der ADC hat einen eigenen Interrupt, der nach einer abgeschlossenen Umsetzung ausgelöst wird. Wenn der ADC-Zugriff auf die Datenregister zwischen dem Auslesen von ADCH und ADCL blockiert ist, wird der Interrupt auch ausgelöst, wenn das Ergebnis verloren geht.

2.2 Start der Umsetzung

Die Umsetzung kann auf verschiedene Weise gestartet werden. Mit dem Bit ADATE aus dem Register ADCSRA (Kap. 2.7.2) wird eingestellt, ob nur eine einmalige Umsetzung oder aber eine durch bestimmte Ereignisse getriggerte Umsetzung erfolgen soll.

Eine einzelne Umsetzung wird durch das Setzen des Bit ADSC (ADC Start Conversion) gestartet. Dieses Bit bleibt genau so lange Eins bis die Umsetzung abgeschlossen ist, dann wird es durch die Hardware wieder zurückgesetzt. Wenn der Eingangskanal während einer Umsetzung gewechselt wird, wird der ADC die aktuelle Umsetzung zunächst bis zum Abschluss durchführen, bevor der Kanalwechsel ausgeführt wird.

Für die getriggerte Umsetzung stehen mehrere Triggerquellen zur Auswahl. Die Triggerquellen werden durch die Bits ADT2-0 im Register SFIOR (Kap. 2.7.4) eingestellt. Die erste Umsetzung muss durch Setzen des Bit ADSC im Register ADCSR gestartet werden.

2.3 Frequenzuntersetzer und Umsetzzeit

Der Schaltkreis der sukzessiven Approximation erfordert eine Eingangstaktfrequenz zwischen 50 kHz und 200 kHz für die höchste Auflösung. Falls eine niedrigere Auflösung als 10 Bit gebraucht wird, kann die Eingangstaktfrequenz höher als 200 kHz sein, um eine höhere Abtastrate zu erzielen.

Der ADC beinhaltet einen Frequenzuntersetzer, der eine für den ADC geeignete Taktrate aus der CPU-Taktrate erzeugt (siehe Kap. 2.7.2). Die Untersetzung wird durch die Bits ADPS2-0 im Register ADCSR eingestellt. Der Untersetzer startet in dem Einschaltaugenblick des ADC, wenn das Bit ADEN im Register ADCSR gesetzt wird. Der Untersetzer läuft ohne Unterbrechung, solange das Bit ADEN gesetzt ist und wird fortwährend zurückgesetzt, solange ADEN Null ist.

Wenn eine Umsetzung durch das Setzen des Bit ADSC im Register ADCSR eingeleitet wird, startet die Umsetzung auf der folgenden steigenden Flanke des ADC Taktes. Eine normale Umsetzung beansprucht 13 Takte (Abbildung 3). Die erste Umsetzung nach dem Einschalten des ADC (Setzen von ADEN im Register ADCSR) dauert 25 Takte, da die analoge Schaltung erst initialisiert werden muss (Abbildung 2).

Der aktuelle Sample-And-Hold passiert 1,5 Takte nach dem Start einer normalen Umsetzung und 13,5 Takte nach dem Start der ersten Umsetzung. Wenn eine Umsetzung komplett ist, wird das Ergebnis in das ADC Datenregister geschrieben und ADIF wird gesetzt. Bei einer einzelnen Umsetzung wird das Bit ADSC gleichzeitig gelöscht. Per Software kann das Bit ADSC dann wieder gesetzt

werden und eine neue Umsetzung auf der ersten steigenden Flanke des ADC-Taktes eingeleitet werden.

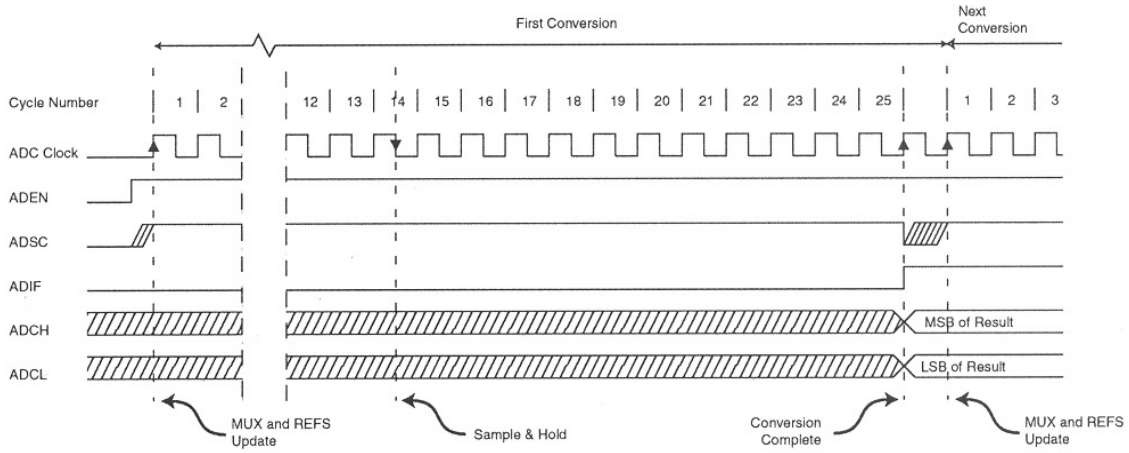

Abbildung 2: ADC Zeitdiagramm für die erste Umsetzung

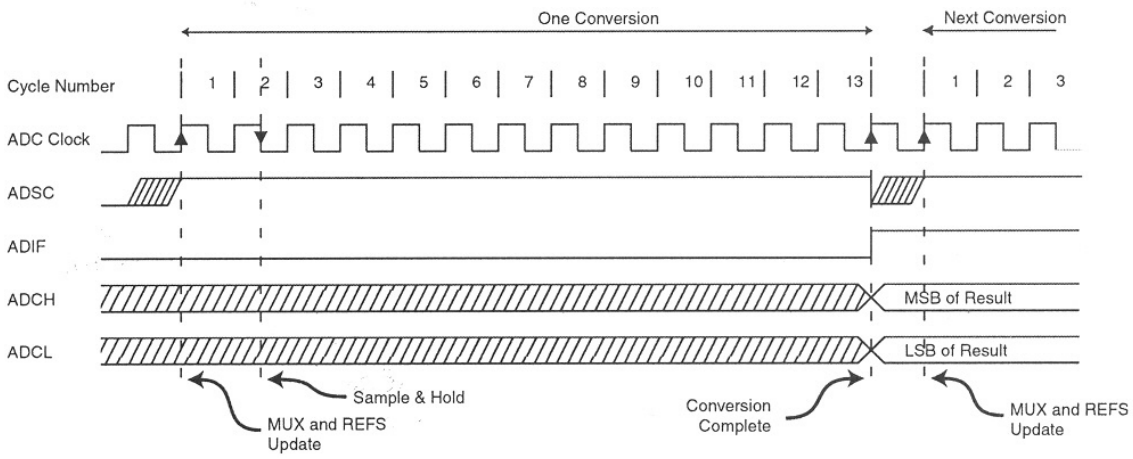

Abbildung 3: ADC Zeitdiagramm für eine einzelne Umsetzung

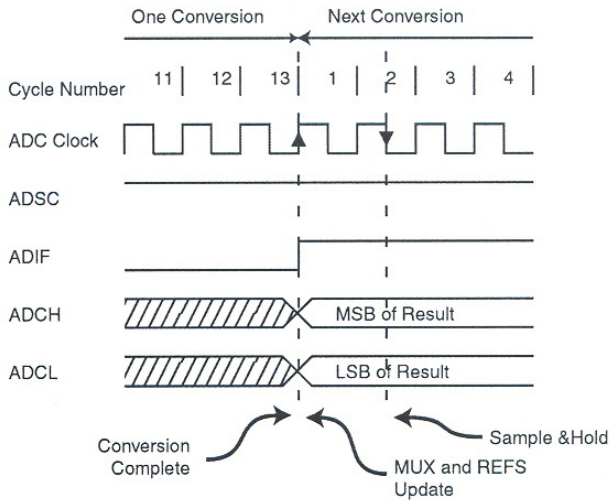

Abbildung 4: ADC Zeitdiagramm bei fortlaufender Umsetzung

2.4 Kanalwechsel und Referenzauswahl

Die Bits MUXn und REFSn aus dem Register ADMUX legen die Kanalauswahl und die Referenzspannung fest (Details werden im Kapitel 2.7.1 gezeigt). Die Werte werden in einem temporären Register zwischengespeichert auf das die CPU freien Zugriff hat. Das garantiert, dass der Kanalwechsel und die Referenzauswahl nur zu sicheren Zeitpunkten während der Umsetzung statt findet. Die Kanal- und Referenzauswahl wird fortwährend aktualisiert bis eine Umsetzung gestartet ist.

Wenn einmal eine Umsetzung gestartet wurde, ist die Kanal- und Referenzauswahl gesperrt, um für den ADC eine ausreichende Abtastzeit sicherzustellen. Das fortwährende Aktualisieren wird im letzten Taktzyklus vor dem Abschluss der Umsetzung fortgesetzt. Man beachte, dass die Umsetzung auf die folgende steigende Flanke des ADC-Taktes startet, nachdem das Bit ADSC gesetzt wurde.

2.4.5 ADC-Eingangskanäle

Wenn die Kanalauswahl geändert wird, sollte der Anwender folgende Richtlinien beachten, um sicherzustellen, dass der korrekte Kanal ausgewählt ist.

Für einmalige Umsetzungen sollte man die Kanalauswahl immer vor dem Start der Umsetzung auswählen.

Bei getriggerten Umsetzungen sollte man den Kanal auswählen, bevor die erste Umsetzung startet. Wird die Kanalauswahl geändert, wenn die Umsetzung schon gestartet wurde, wird das nächste Ergebnis die vorherige Kanalauswahl wiedergeben. Nachfolgende Umsetzungen werden die neue Kanalauswahl wiedergeben.

2.4.6 ADC-Referenzspannung

Die Referenzspannung V_{ref} des AD-Umsetzers legt den Eingangsbereich fest. Eingangskanäle, die die Referenzspannung übersteigen, werden als 0x3FF eingelesen. V_{ref} kann ausgewählt zwischen AV_{CC}, der internen Referenzspannung oder einer Spannung, die an den externen Anschluss AREF gelegt wird.

Wenn der Anwender eine externe Referenzspannung angeschlossen hat, sollte er keine anderen Referenzspannungsoptionen nutzen, da die Spannungen praktisch widerstandslos verbunden werden und somit hohe Ausgleichsströme fließen. Falls keine externe Referenzspannung benutzt wird, kann der Anwender zwischen den beiden übrigen Optionen beliebig umschalten. Die erste Umsetzung nach dem Umschalten der Referenzspannung kann jedoch fehlerbehaftet sein und der Anwender sollte dieses Ergebnis verwerfen.

2.5 Analoger Eingangskreis

Der analoge Eingangskreis ist in Abbildung 5 illustriert. Eine an den Anschluss ADCn angeschlossene analoge Quelle wird durch den Ladekondensator $C_{S/H}$, den Serienwiderstand $R_{S/H}$ und den Ableitungswiderständen R1 und R2 belastet.

Der AD-Converter ist ausgelegt für analoge Eingangsquellen mit einem Widerstand von ungefähr 10 kΩ oder weniger. Wenn eine derartige Quelle benutzt wird, ist die Aufladezeit des Kondensators vernachlässigbar. Wird jedoch eine Quelle mit höherem Widerstand angeschlossen, so wird die Ladezeit des Kondensators in Abhängigkeit von dem Widerstand erhöht. Der Anwender sollte nur niederohmige Quellen verwenden, so dass die Ladezeit des Kondensators klein bleibt.

Bei dem Einlesen von zeitabhängigen Signalen sollte der Anwender darauf achten, dass das Abtasttheorem nach Nyquist ($f_{ADC/2}$) erfüllt ist. Höherfrequente Signale sollten einen Eingangstiefpass durchlaufen, um Aliasing zu vermeiden.

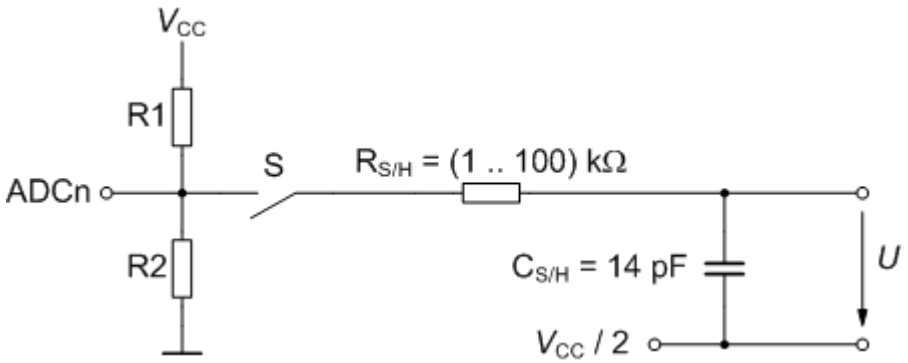

Abbildung 5: Analoger Eingangskreis

2.6 Umsetzungsergebnis

Nach einer kompletten Umsetzung (ADIF ist auf Eins gesetzt) kann das Ergebnis aus den Datenregistern (ADCL und ADCH) ausgelesen werden. Bei einer Umsetzung mit einer Auflösung von 10 Bit kann das Ergebnis wie folgt bestimmt werden:

$$ADC = \frac{V_{IN} \cdot 2^{10}}{V_{ref}},$$

wobei V_{IN} die Eingangsspannung und V_{ref} die gemäß Kap. 2.4.6 gewählte Referenzspannung ist. 0x000 repräsentiert GND und 2^{10} (1024_D) die Referenzspannung minus 1 LSB.

$$LSB = \frac{V_{ref}}{1024}$$

2.7 Beteiligte Register

Für die Steuerung des AD-Umsetzers, sowie zum Abruf des Ergebnisses werden die folgenden Register verwendet.

2.7.1 ADC Multiplexer Auswahlregister - ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	**REFS0**	**ADLAR**	**MUX4**	**MUX3**	**MUX2**	**MUX1**	**MUX0**	**ADMUX**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

REFS1 und REFS0
Diese Bits wählen die Referenzspannung für den AD-Converter, wie in Tabelle 1 gezeigt wird.

Tabelle 1: Referenzspannungsauswahl

REFS1	REFS0	Referenzspannung
0	0	Externe Spannung an AREF, interne Referenz ist ausgeschaltet
0	1	AV_{CC}
1	0	Reserviert
1	1	Interne 2,56 V Referenz

ADLAR (ADC Left Adjust Result)
Das Bit ADLAR wirkt sich auf die Darstellung des Umsetzungsergebnisses in den Datenregistern aus. Um das Ergebnis linksbündig darzustellen, wählt man ADLAR = 1. Andernfalls wird das Ergebnis rechtsbündig dargestellt. Die Änderung des Bit wirkt sich sofort auf die Datenregister aus.

MUX 4 bis MUX0
Diese Bits wählen die Kombination der Eingangskanäle für den AD-Converter entsprechend der Tabelle 2. Für die differentiellen Eingänge werden durch die Bits auch die Verstärkung eingestellt.

Tabelle 2: Eingangskanalsauswahl

MUX4:0	Single Ended Input	Pos. Differential Input	Neg. Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			

00111	ADC7			
01000	0	ADC0	ADC0	-
01001	0	ADC1	ADC0	-
01010	0	ADC0	ADC0	-
01011	0	ADC1	ADC0	-
01100	1	ADC2	ADC2	-
01101	1	ADC3	ADC2	-
01110	1	ADC2	ADC2	-
01111	1	ADC3	ADC2	1,30 V
10000		ADC0	ADC1	0 V
10001		ADC1	ADC1	
10010		ADC2	ADC1	
10011		ADC3	ADC1	
10100		ADC4	ADC1	
10101		ADC5	ADC1	
10110		ADC6	ADC1	
10111		ADC7	ADC1	
11000		ADC0	ADC2	
11001		ADC1	ADC2	
11010		ADC2	ADC2	
11011		ADC3	ADC2	
11100		ADC4	ADC2	
11101		ADC5	ADC2	
11110	1.2 V (V_{BG})			
11111	0 V (GND)			

2.7.2 ADC Control and Status Register A - ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	**ADSC**	**ADATE**	**ADIF**	**ADIE**	**ADPS2**	**ADPS1**	**ADPS0**	**ADCSRA**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADEN: ADC Enable
Durch das Setzen dieses Bit wird der AD-Umsetzer aktiviert. Beim Schreiben einer Null wird der AD-Umsetzer ausgeschaltet. Wird der ADC während einer Umsetzung ausgeschaltet, so wird die Umsetzung abgebrochen.

ADSC: ADC Start Conversion
Bei einer einzelnen Umsetzung wird jede Umsetzung durch das Schreiben einer Eins in dieses Bit gestartet. Bei freilaufender Umsetzung wird durch Schreiben einer Eins in dieses Bit die erste Umsetzung gestartet.

Das Bit ADSC wird solange als Eins gelesen, wie die Umsetzung läuft. Wenn die Umsetzung abgeschlossen ist, wird ADSC als Null gelesen. Das Schreiben einer Null in das Bit hat keine Auswirkungen.

ADATE: ADC Auto Trigger Enable
Wenn dieses Bit auf Eins gesetzt ist, wird die Umwandlung automatisch gestartet. Die Umwandlung wird bei einer positiven Flanke des ausgewählten Triggersignals gestartet. Die Triggerquelle wird im Register SFIOR ausgewählt.

ADIF: ADC Interrupt Flag
Dieses Bit wird gesetzt, wenn eine Umsetzung abgeschlossen ist und die Datenregister aktualisiert sind. Der Interrupt ADC Conversion Complete wird aufgerufen, wenn das Bit ADIE und das I-Bit im Statusregister gesetzt sind. ADIF wird von der Hardware automatisch zurückgesetzt, wenn der entsprechende Interrupt-Vektor aufgerufen wird. Alternativ kann ADIF durch das Überschreiben mit Eins zurückgesetzt werden.

ADIE: ADC Interrupt Enable
Wenn dieses Bit auf Eins gesetzt wird und das I-Bit im Statusregister gesetzt ist, ist der Interrupt ADC Conversion Complete aktiviert.

ADPSn: ADC Prescaler Select Bits
Diese Bits bestimmen das Untersetzungsverhältnis zwischen dem Systemtakt und dem Eingangstakt des ADC.

Tabelle 3: ADC Untersetzungsauswahl

ADPS2	ADPS1	ADPS0	Teilungsverhältnis
0	0	0	2
0	0	0	2
0	0	1	4
0	0	1	8
0	1	0	16
0	1	0	32
0	1	1	64
0	1	1	128

2.7.3 ADC Datenregister – ADCL und ADCH

Wenn eine AD-Umsetzung abgeschlossen ist, ist das Ergebnis in den Registern ADCH und ADCL zu finden. Wenn ADCL ausgelesen wurde, bleiben die Datenregister unverändert bis auch ADCH ausgelesen wurde. Wenn eine Auflösung von acht Bit für die Anwendung ausreicht und das Ergebnis linksbündig angeordnet ist, reicht es aus, das Register ADCH auszulesen. Andernfalls muss zuerst ADCL und anschließend ADCH ausgelesen werden.

Das Bit ADLAR aus dem Register ADMUX beeinflusst die Darstellung des Ergebnisses. Ist ADLAR = 1, wird das Ergebnis rechtsbündig angeordnet.

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Ist ADLAR = 0, wird das Ergebnis linksbündig angeordnet.

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	-	-	-	-	-	-	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

2.7.4 Special Function IO Register - SFIOR

Nicht sämtliche Bits aus dem Register SFIOR werden vom ADC genutzt. Die genutzten Bits sind in der folgenden Darstellung grau markiert. Die Funktion der übrigen Bits wird an dieser Stelle nicht erklärt.

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADTS 2:0: ADC Auto Trigger Source
Durch das Setzen dieses Bit wird der AD-Umsetzer aktiviert. Beim Schreiben einer Null wird der AD-Umsetzer ausgeschaltet. Wird der ADC während einer Umsetzung ausgeschaltet, so wird die Umsetzung abgebrochen.

Tabelle 4: ADC Triggerquelle

ADTS2	ADTS1	ADTS0	Triggerquelle
0	0	0	Automatische startende Umsetzung (ohne Triggerquelle)
0	0	0	Analogkomparator
0	0	1	Externer Interrupt 0
0	0	1	Übereinstimmung mit Timer / Counter 0 Vergleichsregister
0	1	0	Timer / Counter 0 Überlauf
0	1	0	Übereinstimmung mit T/C1 Vergleichsregister B
0	1	1	Timer / Counter 1 Überlauf

0	1	1	Timer / Counter 1 Capture Ereignis

2.8 Beispielcode für das Einlesen von Analogwerten

Der folgende Beispielcode zeigt wie man den AD-Converter in der initialisiert und im Programmverlauf Werte einliest.

```
// Funktion zum Einlesen eines Analogwertes
// Rückgabewert ist der zu 8-Bit gewandelte Analogwert
//Übergabeparameter ist der AD-Kanal
char AD_READ (int CHANNEL)
// Initialisierung des ADC
{
ADMUX = (1 << ADLAR) | (0 << REFS1) | (1 << REFS0);
ADMUX|=CHANNEL;
ADCSRA= (1 << ADEN);
// Start der Umwandlung
ADCSRA|= (1 << ADSC);
// Abwarten der Umwandlung
while(ADSC==1);
// Schreiben des Rückgabewertes
return
```

3 Analoge Ausgangsspannung mittels PWM

Die Pulsweitenmodulation (PWM) ist ein Verfahren, um aus einem Digitalwert einen analogen Spannungswert (DA-Umsetzung) zu erzeugen. Mit Hilfe der Timer/Counter des Mikrocontrollers kann ein pulsweitenmoduliertes Ausgangssignal erzeugt werden. In diesem Versuch wird der Timer/Counter2 benutzt. Das Ausgangssignal ist am Anschluss OC2 abzugreifen.

Die folgende Abbildung 6 zeigt das Prinzip, wie aus der zwischen LOW und HIGH umschaltenden Ausgangsspannung am Anschluss OC2 durch Taktung eine in ihrem Mittelwert veränderbare Ausgangsspannung $U_{D/A}$ erzeugt wird.
Der konstante Zeitraum T (Periodendauer) wird in einen Zeitabschnitt b unterteilt, der dann Pulsweite genannt wird, und in einen Zeitabschnitt (T - b).
Während des Zeitabschnittes b liefert der Ausgang eine Eins, während des Zeitabschnittes (T - b) ein LOW-Signal.
Durch Veränderung der Pulsweite b wird der Mittelwert der Ausgangsspannung verändert.

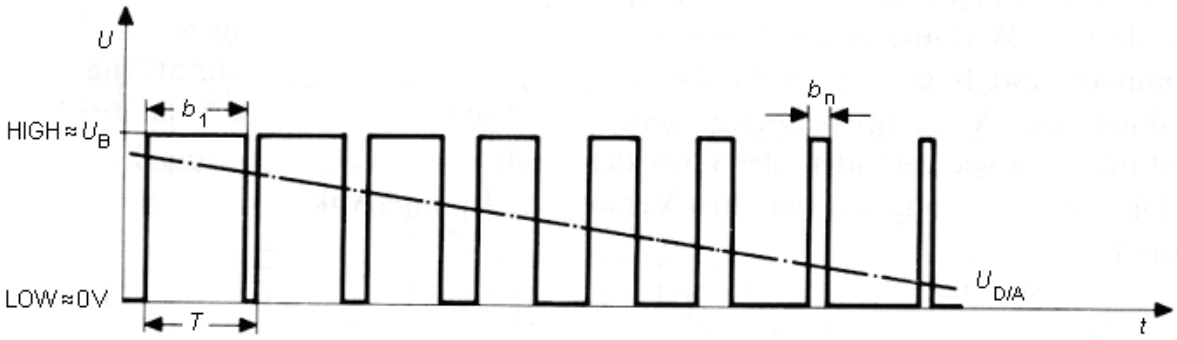

Abbildung 6: Prinzip der PWM

Durch einen Ausgangstiefpass kann die Ausgangsspannung noch geglättet werden (Abbildung 7) und man erhält $U_{D/A}$.

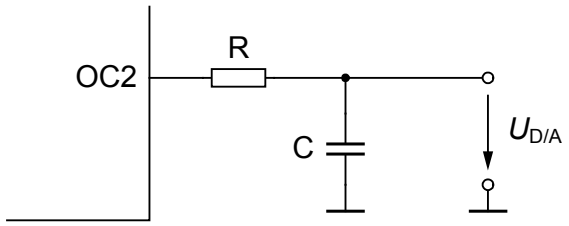

Abbildung 7: Ausgangstiefpass des PWM-Signals

Die Ausgangsspannung $U_{D/A}$ lässt sich wie folgt berechnen:

$$U_{D/A} = \int_0^b \frac{1}{T} U_B \cdot dt \,,$$

$$U_{D/A} = \frac{b}{T} U_B .$$

Der Timer/Counter 2 ist ein vielseitig verwendbares, einkanaliges, 8-Bit Timer/Counter-Modul. Zunächst wird die Funktionsweise anhand des Blockschaltbildes erklärt. Bevor auf einige Details näher eingegangen wird und die bauteilspezifischen I/O-Register und Bits erklärt werden.

3.1 Blockschaltbild des Timer/Counter2

Ein vereinfachtes Blockschaltbild des 8-Bit Timer/Counter2 zeigt die Abbildung 8. I/O-Register und Anschlüsse, auf die die CPU direkten Zugriff hat, sind fett gedruckt.

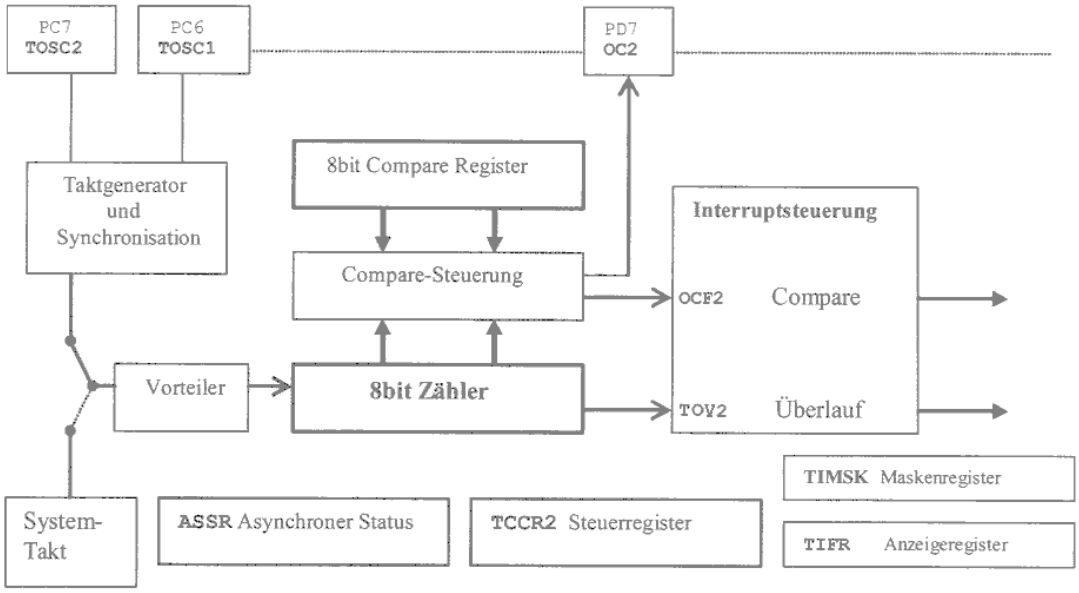

Abbildung 8: 8-Bit Timer/Counter Block-Diagramm

Der 8-Bit Zähler (TCNT2) und das 8-Bit Output Compare Register (OCR2) werden ständig verglichen. Das Ergebnis des Vergleichs kann von der Compare-Steuerung dazu benutzt werden, um eine Pulsweitenmodulation am Anschluss OC2 auszugeben. Das Vergleichsergebnis setzt gleichzeitig das Vergleichsflag (OCF2), welches benutzt werden kann, um eine Output Compare Interrupt Anforderung zu erzeugen. Der Timer/Counter kann von dem Systemtakt oder asynchron durch einen an die Anschlüsse TOSC 1 und TOSC 2 gelegten Takt gesteuert werden. Beide Frequenzen können mit Hilfe des Vorteilers (Frequenzuntersetzer) herabgesetzt werden (siehe Kapitel 3.2). Der Timer/Counter steht still, wenn keine Taktquelle ausgewählt ist. Der asynchrone Betrieb wird durch das Register ASSR (Asynchronous Status Register) eingestellt.

3.2 Taktgenerator

Der Timer/Counter2 kann von dem Systemtakt ($clk_{I/O}$) oder einer externen Taktquelle gesteuert werden. Laut Voreinstellung wird der Takt vom Systemtakt abgeleitet. Wenn jedoch das Bit AS2 im Register ASSR jedoch auf Eins gesetzt ist, wird der Takt des an die Anschlüsse TOSC1 und TOSC2 anzuschließenden Oszillators übernommen.

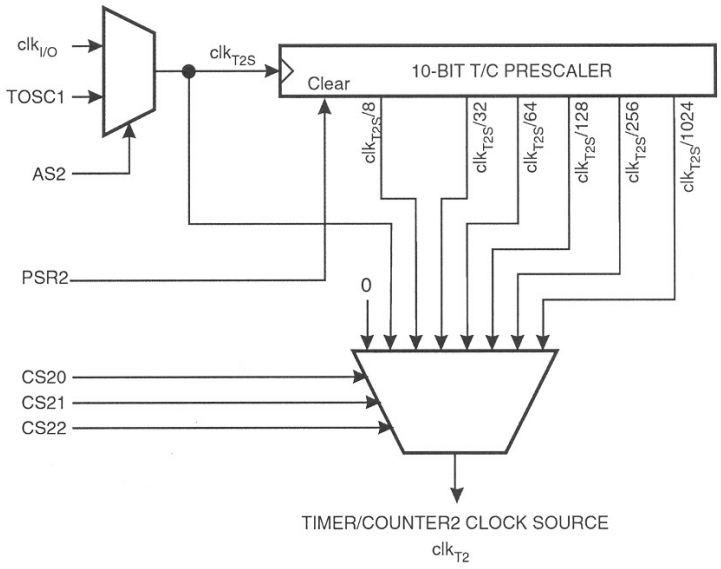

Abbildung 9: Taktgenerator

Mit Hilfe der Steuerbits CS20 - CS22 aus dem Register TCCR2 kann eine Frequenzuntersetzung des Taktes durchgeführt werden. Der genaue Zusammenhang zwischen der Frequenzuntersetzung und den Steuerbits wird im Kapitel 3.6.1 angegeben.

3.3 Zähleinheit

Der Hauptbestandteil des Timer/Counters ist die programmierbare bidirektionale Zähler-Einheit.

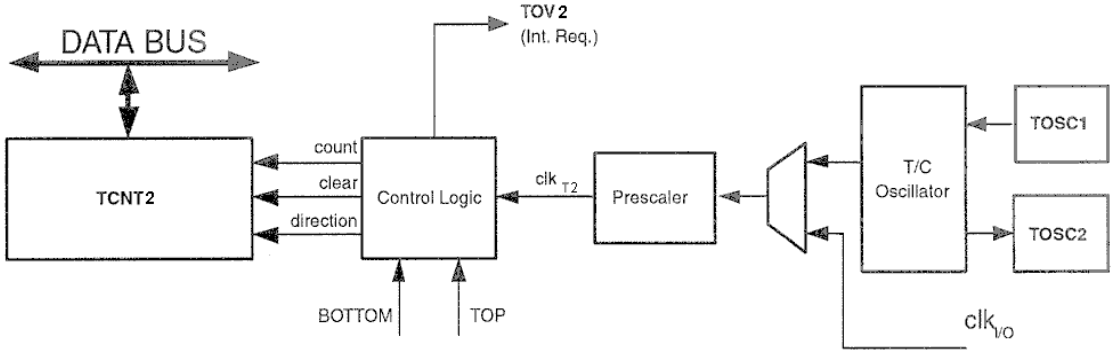

Abbildung 10: Blockschaltbild der Zähler-Einheit

Die internen Signale der Einheit (Abbildung 10) sind:

- count: Aktiviert den Zähler (bei 1),
- direction: Gibt die Zählrichtung vor,
- clear: Setzt alle Bits des Registers TCNT2 auf Null,
- clk_{T2}: Taktsignal,
- TOP: Signalisiert ,dass TCNT2 seinen Höchstwert erreicht hat,
- BOTTOM: Signalisiert ,dass TCNT2 bei Null angekommen ist.

In Abhängigkeit von der Betriebsart wird der Zähler bei jedem Takt zurückgesetzt, inkrementiert oder dekrementiert. Das Signal clk_{T2} kann in Abhängigkeit von den Clock Select Bits des Timer Counter Control Registers (TCCR2) frequenzuntersetzt werden. Falls keine Taktquelle ausgewählt wird, bleibt der Zähler stehen.
Das Timer/Counter Overflow Flag (TOV2) wird entsprechend der Betriebsart gesetzt. TOV2 kann benutzt werden, um einen Interrupt auszulösen. Der Betriebsmodus wird durch die Bits WGM21 und WGM20 des Timer/Counter Control Registers (TCCR2) vorgegeben.

3.4 Ausgangsvergleichseinheit

Der 8-Bit-Komparator vergleicht fortwährend den Zählerstand im Register TCNT2 mit dem Inhalt des Output Compare Registers (OCR2). Bei jeder Übereinstimmung der Registerinhalte zeigt der Komparator diese an. Bei Übereinstimmung wird das Output Compare Bit (OCF2) mit dem nächsten Takt auf Eins gesetzt. OCF2 kann benutzt werden, um einen Interrupt auszulösen.
Der Waveform Generator benutzt das Signal OCF2, um je nach der ausgewählten Betriebsart ein Ausgangssignal am Ausgangspin OC2 zu erzeugen. Die Signale TOP und BOTTOM werden von dem Waveform Generator benutzt, um auf diese Extremwerte des Zählerstandes entsprechend der Betriebsart zu reagieren.
Das Register OCR2 wird bei der Pulsweitenmodulation in einem temporären Register zwischengespeichert. Durch die Zwischenspeicherung wird die Aktualisierung des Registers OCR2 mit dem Erreichen des Zählerhöchststandes bzw. Null synchronisiert. Die Synchronisation verhindert das Auftreten von Glitches am Ausgangspin OC2.

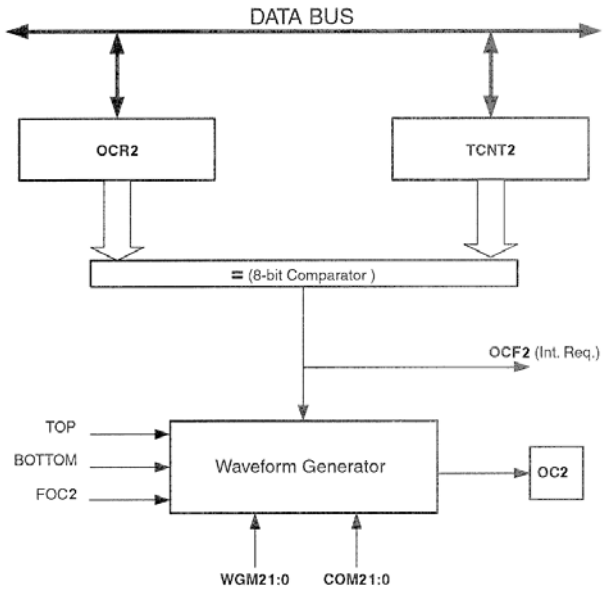

Abbildung 11: Output Compare Unit

3.5 Schneller PWM-Betrieb

Die Betriebsart des Timer/Counters wird im Register TCCR2 eingestellt. Details folgen im Kapitel 3.6.1. In diesem Versuch wird ausschließlich die Betriebsart schnelle PWM benutzt. Deshalb wird an dieser Stelle auf die anderen Betriebsarten nicht weiter eingegangen.

Die schnelle Pulsweitenmodulation ermöglicht die Erzeugung eines hochfrequenten pulsweitenmodulierten Ausgangssignals. Die schnelle Pulsweitenmodulation unterscheidet sich von den anderen PWM-Arten dadurch, dass der Zähler fortwährend nur in eine Richtung zählt. Der Zähler zählt von Null nach 0xFF und springt dann wieder auf Null zurück. Im nichtinvertierenden Betrieb wird das Ausgangssignal beim Zählerstand Null gesetzt und beim Erreichen des Vergleichswertes zurückgesetzt. Im invertierenden Betrieb wird das Ausgangssignal bei Null gesetzt und beim Erreichen des Vergleichswertes auf Eins gesetzt. In der Abbildung 12 ist der zeitliche Verlauf der Signale dargestellt.

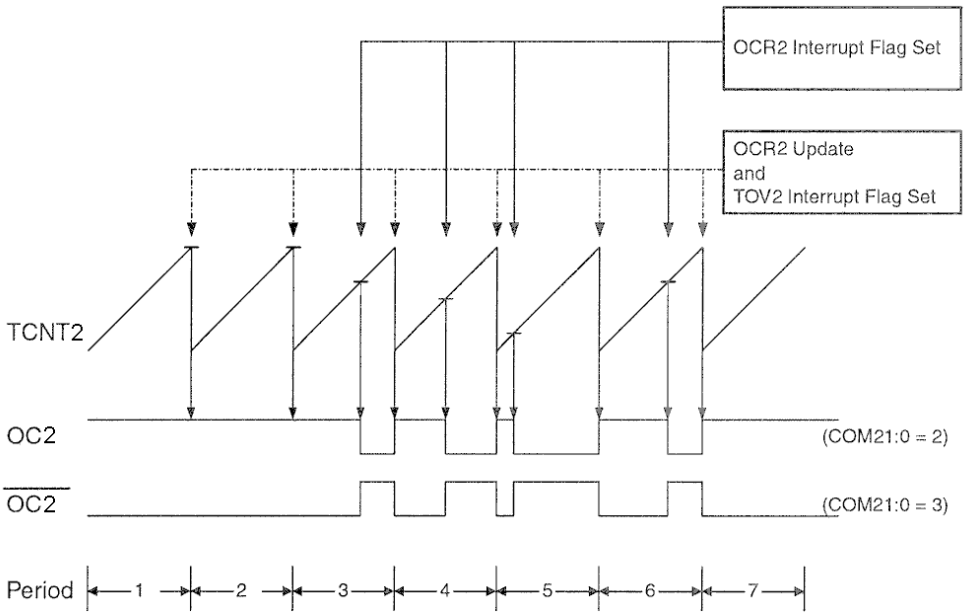

Abbildung 12: Zeitdiagramm bei schneller Pulsweitenmodulation

Das Flag Timer/Counter Overflow (TOV2) wird jedes Mal gesetzt, wenn der Zähler seinen Höchststand erreicht. Falls der Interrupt freigegeben ist, kann die Interrupt Service Routine benutzt werden, um den Vergleichswert zu aktualisieren.
Das pulsweitenmodulierte Signal wird am Anschlusspin OC2 ausgegeben. Der Wert von OC2 kann am Ausgang nur abgegriffen werden, wenn das Datenrichtungsregister des Pins auf Ausgang gestellt ist.
Die PWM-Frequenz des Ausgangssignals wird wie folgt berechnet:

$$f_{\mathrm{OC2PWM}} = \frac{f_{\mathrm{clk\_I/O}}}{N \cdot 256},$$

wobei die Variable *N* den Teilungsfaktor (1, 8, 32, 64, 128, 256 oder 1024) bezeichnet.

Die Werte Null und FF_H des Vergleichsregisters repräsentieren Sonderfälle am Ausgang der schnellen Pulsweitenmodulation. Falls im Vergleichsregister Null steht, tritt am Ausgang ein Peak auf. Falls im Vergleichsregister der Höchstwert steht, liegt am Ausgang konstant High-Pegel (bei invertierendem Betrieb) oder Low-Pegel (bei nichtinvertierendem Betrieb) Spannung an.

3.6 Beteiligte Register

Für die Steuerung von Timer/Counter2 werden die folgenden Register verwendet.

3.6.1 Timer/Counter Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
	FOC2	**WGM20**	**COM21**	**COM20**	**WGM21**	**CS22**	**CS21**	**CS20**	**TCCR2**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

FOC2: Force Output Compare
Das Bit FOC2 ist nur aktiv, wenn die Bits WGM20 und WGM21 einen PWM-Betrieb verhindern. Um die Kompatibilität mit den Nachfolgemodellen des ATmega16 sicherzustellen, muss jedoch das Bit auf Null gesetzt werden, wenn das Register TCCR2 beschrieben wird und der PWM-Mode vorgesehen ist.
Das Bit FOC2 wird immer als Null gelesen.

WGM 21 und WGM 20: Einstellung der Betriebsart
Diese Bits kontrollieren die Zählfolge des Zählers, die Quelle des Höchstwertes und den Ausgangskurvenverlauf. Die folgende Tabelle zeigt die verschiedenen Betriebsmodi in Abhängigkeit von den Bits WGM 21 und WGM 21, wobei im vorliegenden Versuch nur die Betriebsart „Schnelle PWM“ relevant ist. Auf die anderen Betriebsarten wird deshalb an dieser Stelle nicht eingegangen.

Tabelle 5: Betriebsarten von Timer/Counter2

Betriebs-art	WGM21	WGM20	Betriebs-art	Zähler-höchst-stand	Aktualisierung von OCR2	Setzen des TOV2 Flag bei
0	0	0	Normal	0xFF	sofort	bei Zählerhöchststand
1	0	1	PWM, phasenrichtig	0xFF	bei Zählerhöchststand	bei Zählerstand 0x00
2	1	0	CTC	OCR2	sofort	bei Zählerhöchststand
3	1	1	Schnelle PWM	0xFF	bei Zählerstand 0x00	bei Zählerhöchststand

COM21 und COM20: Einstellung am Ausgangsanschluss
Diese Bits steuern das Verhalten des Ausgangspins OC2. Wenn eines dieser Bits oder beide gesetzt sind, dann bestimmt das Bit OC2 die Funktion des zugeordneten Ausgangspins. Trotzdem muss das entsprechende Bit des Ausgangspins im entsprechenden Data-Direction-Register gesetzt sein, um den Ausgangstreiber zu aktivieren.

Die folgende Tabelle zeigt die möglichen Einstellungen von COM21 und COM20 und die Auswirkung für den Ausgang in der Betriebsart „Schnelle PWM“:

Tabelle 6: Einstellungen des Ausgangsanschlusses

COM21	COM20	Beschreibung
0	0	OC2 ist nicht mit dem Port verbunden.
0	1	Reserviert
1	0	OC2 wird beim Zählerstand 0x00 gesetzt und beim Erreichen des Vergleichswertes gelöscht (nichtinvertierender Betrieb).
1	1	OC2 wird beim Zählerstand 0x00 gelöscht und beim Erreichen des Vergleichswertes gesetzt (invertierender Betrieb).

CS22, CS21 und CS20: Taktauswahl
Mit den drei Taktauswahlbits wird der Frequenzuntersetzer zur Erzeugung des Taktsignals des Timer/Counter eingestellt:

Tabelle 7: Taktauswahl

CS22	CS21	CS20	Beschreibung
0	0	0	Timer/Counter ist angehalten
0	0	1	$clk_{T2} = clk_{T2S}$ keine Frequenzuntersetzung
0	1	0	$clk_{T2} = clk_{T2S} / 8$
0	1	1	$clk_{T2} = clk_{T2S} / 32$
1	0	0	$clk_{T2} = clk_{T2S} / 64$
1	0	1	$clk_{T2} = clk_{T2S} / 128$
1	1	0	$clk_{T2} = clk_{T2S} / 256$
1	1	1	$clk_{T2} = clk_{T2S} / 1024$

Die Taktquelle von Timer/Counter2 wird mit clk_{T2S} bezeichnet. Das Signal clk_{T2S} ist nach einem Reset mit dem Systemtakt verbunden. Durch das Setzen des Bit AS2 im Register ASSR kann der Timer/Counter2 auch von einem externen Takt gesteuert werden (asynchroner Betrieb).

3.6.2 Timer/Counter Register – TCNT2

Bit	7	6	5	4	3	2	1	0	
	TCNT27	**TCNT26**	**TCNT25**	**TCNT24**	**TCNT23**	**TCNT22**	**TCNT21**	**TCNT20**	**TCNT2**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Das Timer/Counter Register erlaubt direkten Zugriff auf den aktuellen Zählerstand, sowohl in Lese- als auch in Schreibrichtung. Das Schreiben in das Register TCNT2 blockiert den Vergleich mit dem Register OCR2.

3.6.3 Vergleichsregister – OCR2

Bit	7	6	5	4	3	2	1	0	
	OCR27	**OCR26**	**OCR25**	**OCR24**	**OCR23**	**OCR22**	**OCR21**	**OCR20**	**OCR2**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Das Vergleichsregister beinhaltet einen 8-Bit-Wert der fortwährend mit dem Inhalt des Registers TCNT2 verglichen wird. Die Übereinstimmung kann genutzt werden, um einen Interrupt oder eine Reaktion am Ausgangspin OC2 zu erzeugen.

3.6.4 Statusregister für asynchronen Betrieb – ASSR

Bit	7	6	5	4	3	2	1	0	
	-	**-**	**-**	**-**	**AS2**	**TCN2UB**	**OCR2UB**	**TCR2UB**	**ASSR**
Read/Write	R	R	R	R	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

AS2: Asynchroner Timer/Counter2
Wenn das Bit AS2 auf Null gesetzt ist, wird Timer/Counter2 von dem Prozessortakt bzw. einem von ihm abgeleiteten Takt versorgt. Falls das Bit AS2 auf Eins gesetzt ist, wird der Timer/Counter2 von dem an den Anschlusspin TOSC1 angeschlossenen Takt gesteuert. Wenn das Bit AS2 geändert wird, können die Inhalte der Register TCNT2, OCR2 und TCCR2 verloren gehen.

TCN2UB: Timer/Counter2 Update Busy
Überschreibt man im asynchronen Betrieb das Register TCNT2, wird dieses Bit gesetzt. Wenn das Register TCNT2 von dem temporären Speicherregister aktualisiert wurde, wird das Bit von der Hardware zurückgesetzt. Die logische Null in diesem Speicherbit zeigt an, dass TCNT2 bereit ist, um mit einem neuen Wert beschrieben zu werden.

OCR2UB: Output Compare Update Busy
Überschreibt man im asynchronen Betrieb das Register OCR2, wird dieses Bit gesetzt. Wenn das Register OCR2 von dem temporären Speicherregister aktualisiert wurde, wird das Bit von der Hardware zurückgesetzt. Die logische Null in diesem Speicherbit zeigt an, dass OCR2 bereit ist, um mit einem neuen Wert beschrieben zu werden.

TCR2UB: Timer/Counter Control Register 2 Update Busy
Überschreibt man im asynchronen Betrieb das Register TCCR2, wird dieses Bit gesetzt. Wenn das Register TCCR2 von dem temporären Speicherregister aktualisiert wurde, wird das Bit von der Hardware zurückgesetzt. Die logische Null in diesem Speicherbit zeigt an, dass TCCR 2 bereit ist, um mit einem neuen Wert beschrieben zu werden.

3.6.5 Timer/Counter Interrupt-Mask Register – TIMSK

Das Register TIMSK enthält Bits, die das Verhalten aller drei Timer/Counter des ATmega16 steuern. Hier wird nur auf die für Timer/Counter2 relevanten Bits eingegangen. Die relevanten Bits sind grau unterlegt.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	**TOIE2**	**TICIE1**	**OCIE1A**	**OCIE1B**	**TOIE1**	**-**	**TOIE0**	**TIMSK**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCIE2: Timer/Counter Output Compare Match Interrupt Enable
Wenn das Bit OCIE2 auf Eins gesetzt wird und das I-Bit im Status-Register gesetzt ist, ist der Interrupt „Timer/Counter2 Compare Match“ freigegeben.

TOIE2: Timer/Counter Overflow Enable
Wenn das Bit TOIE2 auf Eins gesetzt wird und das I-Bit im Status-Register gesetzt ist, ist der Interrupt „Timer/Counter2 Overflow“ freigegeben.

3.6.6 Timer/Counter Interrupt-Flag Register – TIFR

Das Register TIFR enthält Bits, die das Verhalten aller drei Timer/Counter des ATmega16 steuern. Hier wird nur auf die für Timer/Counter2 relevanten Bits eingegangen. Die relevanten Bits sind grau unterlegt.

Bit	7	6	5	4	3	2	1	0	
	OCF2	**TOV2**	**ICF1**	**OCF1A**	**OCF1B**	**TOV1**	**-**	**TOV0**	**TIFR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCF2: Timer/Counter Output Compare Flag 2
Das Bit OCF2 wird gesetzt, wenn die Inhalte der Register TCNT2 und OCR2 übereinstimmen. Das Bit OCF2 wird von der Hardware zurückgesetzt, wenn der entsprechende Interrupt ausgeführt wird. Alternativ kann das Flag auch durch das Überschreiben mit einer logischen Eins gelöscht werden. Wenn die Bits I, OCIE2 und OCF2 gesetzt sind, wird der Interrupt Timer/Counter2 Compare Match ausgeführt.

TOV2: Timer/Counter Output Overflow Enable
Das Bit TOV2 wird gesetzt, wenn ein Überlauf des Timer/Counter2 erfolgt. Das Bit TOV2 wird von der Hardware zurückgesetzt, wenn der entsprechende Interrupt ausgeführt wird. Alternativ kann das Flag auch durch das Überschreiben mit einer logischen Eins gelöscht werden. Wenn die Bits I, TOIE2 und TOV2 gesetzt sind, wird der Interrupt Timer/Counter2 Overflow ausgeführt.

3.7 Beispielcode für eine PWM

Der folgende Beispielcode zeigt wie man welche Register initialisiert, um ein pulsweitenmoduliertes Ausgangssignal am Pin OC2 zu erzeugen.

```
// Funktion zum Starten des PWM-Betriebs
// FAST PWM
//Übergabeparameter ist der Vergleichswert
void FAST_PWM (int vergleichswert)
{
// Setzen des Vergleichswertes
OCR2 = Vergleichswert;
// Einrichten des Steuerregisters
TCCR2 = (1 << COM21) | (1 << WGM21) | (1 << WGM20) | (1 << CS20);
}
;
```

4 EEPROM Data Memory

Der ATmega16 hat 512 Bytes EEPROM-Datenspeicher. Dieser Bereich ist als getrennter Datenspeicher organisiert, in den einzelne Bytes geschrieben und aus dem einzelne Bytes gelesen werden können. Das EEPROM hat eine Lebensdauer von wenigstens 100000 Schreib-/Lesezyklen. Der Zugriff auf das EEPROM wird im Folgenden beschrieben, die EEPROM Adress-Register, das Datenregister und das EEPROM- Kontrollregister werden erklärt.

4.1 EEPROM Schreib- und Lesezugriff

Der EEPROM-Bereich dient zum nichtflüchtigen Aufbewahren von Daten. Diese gehen im Gegensatz zum SRAM nach dem Abschalten der Versorgungsspannung nicht verloren und stehen nach dem Einschalten wieder zur Verfügung. Die Programmierung erfolgt entweder über eine spezielle Datei, die der Anwender erzeugt oder durch Schreibzugriff innerhalb des Anwendungsprogramms.

Für den Zugriff auf das EEPROM stehen spezielle Register zur Verfügung. Die EEPROM-Zugriffsregister gehören zum I/O-Bereich des Prozessors. Der Schreibzugriff auf das EEPROM dauert 8,5 ms. Eine Zeitfunktion zeigt der Benutzersoftware an, wenn das nächste Byte geschrieben werden kann. Falls aus dem EEPROM gelesen wird, wird die CPU für vier Takte angehalten, bevor die nächste Instruktion ausgeführt wird. Falls das EEPROM beschrieben wird, wird die CPU für zwei Takte angehalten, bevor die nächste Instruktion ausgeführt wird.

4.2 Beteiligte Register

Für den Zugriff auf den EEPROM-Speicherbereich werden die folgenden Register verwendet.

4.2.1 Die EEPROM-Adress-Register – EERH und EEARL

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	-	**EEAR8**	**EEARH**
	EEAR7	**EEAR6**	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**	**EEARL**
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

Bits 9 – 15: Reservierte Bits
Diese Bits sind reserviert und werden immer als Null gelesen.

Bits 0 – 8: EEPROM-Adresse
Die EEPROM-Adress-Register EEARH und EEARL spezifizieren die EEPROM-Adresse im 512 Byte großen EEPROM-Bereich. Der EEPROM-Adressbereich liegt zwischen Null und 511.

4.2.2 Das EEPROM-Datenregister – EEDR

Bit	7	6	5	4	3	2	1	0	
	EEDR7	**EEDR6**	**EEDR5**	**EEDR4**	**EEDR3**	**EEDR2**	**EEDR1**	**EEDR0**	**EEDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

EEDR0..7: EEPROM-Daten
Bei einem Schreibbefehl beinhaltet das Register EEDR die Daten, die in das angesprochene Register geschrieben werden. Nach einem Lesebefehl beinhaltet das Register EEDR die Daten, die aus dem im EEPROM-Adress-Register angegebenen Register gelesen wurden.

4.2.3 Das EEPROM-Kontrollregister – EECR

Bit	7	6	5	4	3	2	1	0	
	-	**-**	**-**	**-**	**EERIE**	**EEMWE**	**EEWE**	**EERE**	**EECR**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

EEDR4..7: Reservierte Bits
Diese Bits sind reserviert und werden immer als Null gelesen.

EERIE: EEPROM Ready Interrupt Enable
Das Setzen des Bit EERIE auf Eins gibt den Interrupt EEPROM-Ready frei, falls auch das Bit I im SREG gesetzt ist. Das Rücksetzen des Bit EERIE sperrt den Interrupt EEPROM-Ready.

EEMWE: EEPROM Master Write Enable
Das Bit EEMWE bestimmt, ob das Setzen des Bit EEWE auf Eins auch ein Schreiben ins EEPROM zur Folge hat. Nur falls EEMWE gesetzt ist und das Bit EEWE innerhalb von vier Taktzyklen später gesetzt wird, werden tatsächlich Daten ins EEPROM geschrieben. Falls EEMWE Null ist, hat das Setzen von EEWE keine Auswirkung. Falls das Bit EEMWE durch die Anwendersoftware gesetzt wurde, setzt die Hardware dieses Bit automatisch nach vier Taktzyklen zurück.

EEWE: EEPROM Write Enable
Das Bit EEWE löst das Schreiben in das EEPROM aus. Wenn die Adresse und die Daten korrekt eingerichtet sind, muss das Bit EEWE auf Eins gesetzt werden, um Daten ins EEPROM zu schreiben. Bevor das Bit EEWE auf Eins geschrieben wird, muss das Bit EEMWE auf Eins gesetzt werden, ansonsten werden keine Daten ins EEPROM geschrieben.

Nachdem die Zugriffszeit für das Beschreiben des EEPROM überschritten ist, wird EEWE durch die Hardware zurückgesetzt. Die Anwendersoftware kann dieses Bit abfragen und nach dem das Bit als Null eingelesen wird, mit dem Beschreiben des EEPROM fortfahren.

EERE: EEPROM Read Enable
Das Bit EEWE löst das Auslesen des EEPROM aus. Wenn die Adresse im Register EEAR eingerichtet ist, muss das Bit EERE auf Eins gesetzt werden, um Daten aus dem EEPROM zu lesen. Das Auslesen benötigt nur einen Befehl, und die Daten sind sofort abrufbar. Wenn aus dem EEPROM ausgelesen wird, wird die CPU für vier Takte angehalten, bevor der nächste Befehl abgearbeitet wird. Bevor der Anwender ein Lesezugriff startet, sollte er das Bit EEWE abfragen, da es während eines Schreibvorganges in das EEPROM weder möglich ist, das EEPROM auszulesen noch das Adressregister EEAR zu ändern.

4.3 Beispielcode für das Beschreiben des EEPROM

Der folgende Programmcode zeigt eine Beispielroutine für das Schreiben in das EEPROM:

- Zugriffsroutine für das Anwenderprogramm:

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
/* Abwarten bis der vorherige Schreibvorgang abgeschlossen ist */
while(EECR & (1<<EEWE))
;
/* Einrichten der Adresse und des Datenregisters */
EEAR = uiAddress;
EEDR = ucData;
/* Setzen des Bit EEMWE auf logisch Eins */
EECR |= (1<<EEMWE);
/* Start des Schreibvorganges durch Setzen von EEWE */
EECR |= (1<<EEWE);
}
```

- Möchte man direkt einen Bereich des EEPROM beschreiben, so kann man z. B ein Array erzeugen und es im EEPROM ablegen. Dazu wird der Variablen das Attribut „EEPROM“ zugewiesen. Diese Art der Zuweisung muss außerhalb der Funktion „main“ vereinbart werden.

```
unsigned char __attribute__((section(".eeprom"))) sinus[4] = {10,30,5,40};
```

Beim Starten des Debuggers bekommt man die in Abbildung 13 gezeigte Meldung, die mit OK zu beantworten ist.

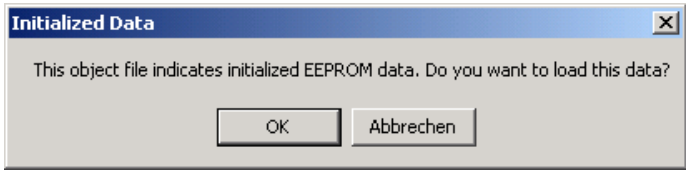

Abbildung 13: Bestätigen der EEPROM-Programmierung

4.4 Beispielcode für das Auslesen des EEPROM

Der folgende Programmcode zeigt eine Beispielroutine für das Auslesen des EEPROM:

- Beispielfunktion:

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
/* Abwarten bis der vorherige Schreibvorgang abgeschlossen ist */
while(EECR & (1<<EEWE))
;
/* Einrichten des Adressregisters */
EEAR = uiAddress;
/* Start des Lesevorganges*/
EECR |= (1<<EERE);
/* Rückgabewert ist der Inhalt des Datenregisters */
return EEDR;
}
```

5 Versuchsvorbereitung

Für die Ausgabe der verschiedenen Funktionsverläufe sollen für eine Rechteck-, eine Sinus- und eine Rampenfunktion für eine Periode jeweils 16 Stützpunkte berechnet werden. Dazu ist folgende Kennlinie gegeben, die den Zusammenhang zwischen der Ausgangsspannung und dem Wert im Register OCR2 zeigt.

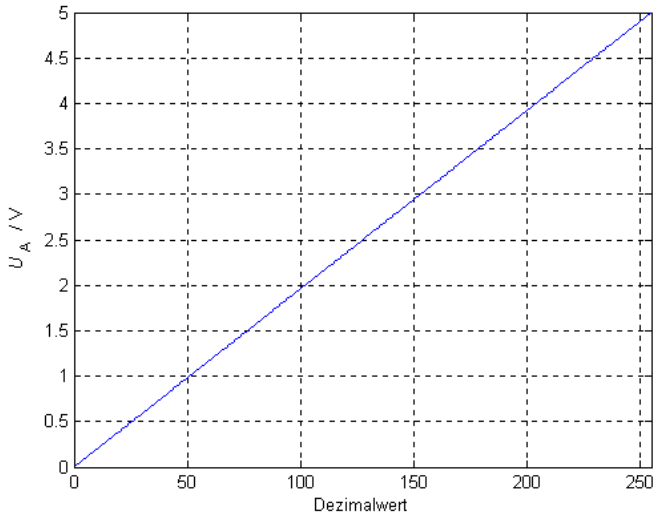

Abbildung 14: Ausgangsspannung in Abhängigkeit von OCR2

5.1 Sinusfunktion

Die folgende Abbildung zeigt die Werte der Ausgangsfunktion in Abhängigkeit von der Nummer des Stützpunktes.

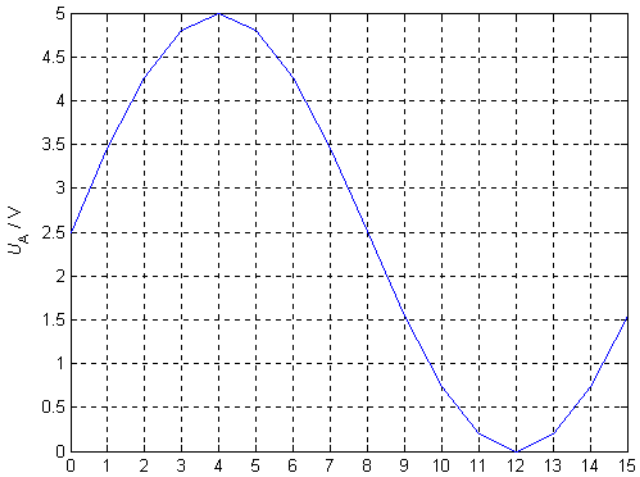

Abbildung 15: Sinusfunktion

Lesen Sie zunächst aus der Abbildung 15 die Ausgangsspannungen ab. Rechnen Sie sie dann in den entsprechenden Dezimal- bzw. Hexadezimalwert um.

Stützpunkt	Analoge Ausgangs-spannung / V	Dez-Wert	Hex-Wert
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

5.2 Rechteckfunktion

Die folgende Abbildung zeigt die Werte der Ausgangsfunktion in Abhängigkeit von der Nummer des Stützpunktes.

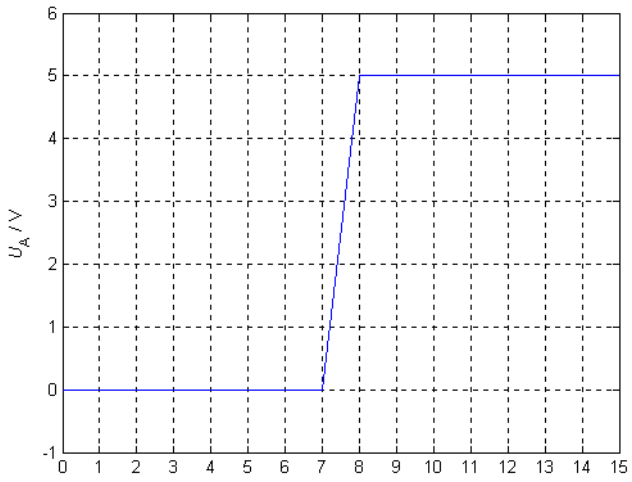

Abbildung 16: Rechteckfunktion

Lesen Sie zunächst aus der Abbildung 15 die Ausgangsspannungen ab. Rechnen Sie sie dann in den entsprechenden Dezimal- bzw. Hexadezimalwert um.

Stützpunkt	Analoge Ausgangs-spannung / V	Dez-Wert	Hex-Wert
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

5.3 Sägezahnfunktion

Die folgende Abbildung zeigt die Werte der Ausgangsfunktion in Abhängigkeit von der Nummer des Stützpunktes.

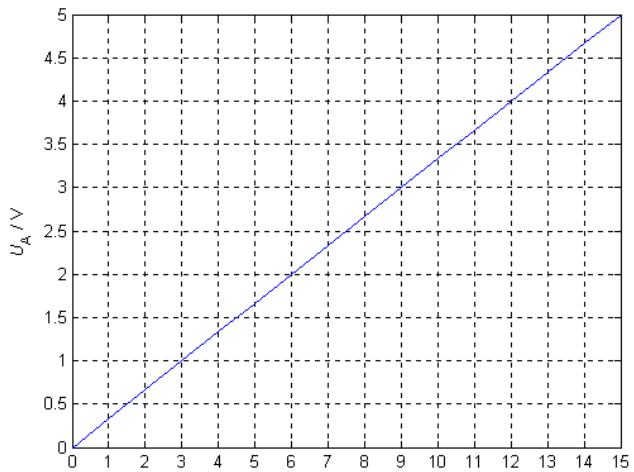

Abbildung 17: Sägezahnfunktion

Lesen Sie zunächst aus der Abbildung 15 die Ausgangsspannungen ab. Rechnen Sie sie dann in den entsprechenden Dezimal- bzw. Hexadezimalwert um.

Stützpunkt	Analoge Ausgangs-spannung / V	Dez-Wert	Hex-Wert
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

6 Versuchsdurchführung

Schrittweise soll ein Funktionsgenerator aufgebaut werden. Der Funktionsgenerator soll:

- Am Ausgang OC2 die drei verschiedenen Funktionsverläufe (Sägezahn, Rechteck, Sinus) liefern.
- Die Funktionsauswahl findet durch Tastendruck der Taster SW0 bis SW2 statt.
- Die Frequenz der Funktion ist über den Analogeingang AD0 zu variieren.

6.1 AD-Umsetzung

6.1.1

Schreiben Sie ein C-Programm, um Analogwerte einzulesen. Der entsprechende Digitalwert soll am Port D angezeigt werden.

Stellen Sie dazu im Register ADMUX folgende Eigenschaften ein:

- Das Ergebnis soll eine Auflösung von acht Bit haben und linksbündig im Register ADCH erscheinen.
- Die Referenzspannung soll die Spannung am Eingang AVCC (5V) sein
- Wählen Sie den Eingangskanal AD0 aus.

Nehmen Sie dazu im Register ADCSRA folgende Einstellungen vor:

- Geben Sie den AD-Converter frei (Enable).
- Starten Sie eine Umsetzung.
- Wählen Sie die Betriebsart: Single Conversion.
- Sperren Sie den Interrupt.
- Die Frequenzuntersetzung des Eingangstaktes zum ADC-Takt soll acht betragen.

6.1.2

Verbinden Sie Port D mit den LED. Notieren Sie die Digitalwerte durch Ablesen der LED für unterschiedliche Eingangsspannungen:

Eingangsspannung *U* / V	0	1	2	3	4	5
Digitalwert						

6.1.3

Speichern Sie ihr Projekt.

6.2 DA-Umsetzung mit schneller PWM

6.2.1

Stellen Sie mit Timer/Counter2 eine Pulsweitenmodulation im Fast PWM-Modus ein. Das Vergleichsregister soll während des Programmablaufs mit drei unterschiedlichen Werten geladen werden. Die zu ladenden Werte sollen sich im EEPROM-Bereich auf den Adressen 00 bis 02 befinden und 0x10, 0x4F und 0xF0 betragen.
Jeder Wert soll für die Dauer des Ablaufs der Funktion „Zeitschleife“ (ca. 1sec.) ausgegeben werden. Zum Programmieren der Zeitschleife benutzen Sie die Funktion „delay“(siehe Versuch 1).

Stellen Sie dazu im Register TCCR2 folgende Eigenschaften ein:

- schnelle PWM,
- nichtinvertierender Betrieb,
- die Frequenzuntersetzung des Eingangstaktes zum Timer/Counter2-Takt soll 1 betragen.

6.2.2

Notieren Sie die für die unterschiedlichen Digitalwerte die Periodendauer *T*, die Pulsweite *b* berechen Sie $U_{D/A}$ für unterschiedliche Digitalwerte:

Dig. Ausgabewert	0x10	0x4F	0xF0
Periodendauer *T* / ms			
Pulsweite *b* / ms			
Ausgangsspannung $U_{D/A}$ / V			

6.2.3

Speichern Sie ihr Projekt.

6.3 Kurvenausgabe (Werte aus Speicherbereich ausgeben)

6.3.1

Vergleichbar mit dem vorherigen Versuchsteil sollen wieder Analogspannungen ausgegeben werden. Jetzt sollen aber keine drei verschiedenen Werte, sondern komplette Kurvenverläufe ausgegeben werden.
Es soll periodisch eine Sinus-, eine Rechteck- oder eine Sägezahnfunktion ausgegeben werden. Die Kurvenfunktion ist über die Taster SW0 bis SW2, welche mit den IO-Ports PD0 bis PD2 verbunden werden, auszuwählen. Wird keine Taste gedrückt, soll die zuletzt angewählte Funktion weiterhin ausgegeben werden.
Für die Kurvenverläufe geben Sie die in der Versuchsvorbereitung berechneten Stützpunkte ein. Die Stützpunkte sollen im EEPROM-Bereich auf 16 hintereinanderliegenden Speicherplätzen abgelegt sein. Jeder Wert soll für die Dauer des

Ablaufs der Funktion „Zeitschleife“ ausgegeben werden. Programmieren Sie das EEPROM mit einem Array (siehe Kap. 4.3).

6.3.2

Messen Sie mit dem Oszilloskop:

Kurvenverlauf	Sägezahn	Sinus	Rechteck
Periodendauer T / ms			
Frequenz f / Hz			
$U_{A\,max}$			
$U_{A\,min}$			

6.3.3

Speichern Sie ihr Projekt.

6.4 Kurvenausgabe mit variabler Geschwindigkeit

6.4.1

Die Frequenz der Ausgabefunktion soll über den Analogeingang AD0 einstellbar sein. Dazu soll die Funktion „Zeitschleife“ wie folgt modifiziert werden:
Innerhalb der Funktion soll der Analogeingang AD0 wie in 6.1 eingelesen werden. Das Ergebnis der AD-Umsetzung soll benutzt werden, um die Zeitschleife zu verlängern bzw. zu verkürzen.

6.4.2

Ermitteln Sie mit dem Oszilloskop die:

	Periodendauer T / ms		
U_{AD0} / V	Sägezahn	Sinus	Rechteck
0,0			
1,0			
2,0			
3,0			
4,0			
5,0			

6.4.3

Speichern Sie ihr Projekt.

7 Literatur

1) AVR-RISC Mikrocontroller
Wolfgang Trampert
Franzis` Verlag, 2000

2) AVR-Mikrocontroller Praxis
Safinaz Volpe
Elektor Verlag, 2001, 2. Auflage

3) Mikrocomputer-Technik, Vorlesungsskript
Prof. Dr. Lothar Howah
Physikalische Technik, FH Gelsenkirchen, 2004

4) Programmieren in C
Brian W. Kernighan, Dennis M. Ritchie
Hanser Verlag München Wien, 1990

5) Mikrocomputertechnik mit Controllern der Atmel AVR-RISC-Familie
Günter Schmitt
Oldenbourg Wissenschaftsverlag, 2006, 2. Auflage

Außerdem gibt es zahlreiche Links im Internet mit Datenblättern, Anwendungen usw.:

1) http://www.avr-forum.com

2) http://www.avrfreaks.net

3) http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf

4) http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial