Institut für Industrielle Informationstechnik (IIIT)
Universität (TH) Karlsruhe

Hertzstr. 16 / Geb. 06.35
76187 Karlsruhe
Tel.: 0721 608 4521
Fax: 0721 608 4500

Prof. Dr.-Ing. Uwe Kiencke
Prof. Dr.-Ing. habil. K. Dostert

Praktikum: **„Mikrocontroller und digitale Signalprozessoren“**

Versuch 1

Digitale Drehzahlmessung mit dem µC 80C517A

Inhaltsverzeichnis

1 Einleitung

In diesem Praktikumsversuch soll die Drehzahl eines Elektromotors mit Hilfe des Mikrocontrollers 80C517A erfaßt werden. Für die Erfassung von Drehzahlsignalen gibt es prinzipiell mehrere Möglichkeiten. Die Winkelgeschwindigkeit einer Welle kann z.B. mit einem Tachogenerator in ein amplitudenanaloges Signal umgeformt werden. Weniger störanfällig und um ein Vielfaches genauer ist demgegenüber das Verfahren, welches das eigentliche Drehzahlsignal mittels Inkrementalgeber in ein periodisches Signal umformt. Die zu messende Winkelgeschwindigkeit ω liegt dann als frequenzanaloges Signal vor. Die Auswertung dieses Signals beruht auf einer Zeitmessung, die durch die Verwendung von Quarzoszillatoren wesentlich genauer realisiert werden kann als dies bei einer Spannungsauswertung mit vertretbarem Aufwand der Fall ist. Die dazu notwendigen Zähler sind in der Peripherie von handelsüblichen Mikrocontrollersystemen mehrfach vorhanden.

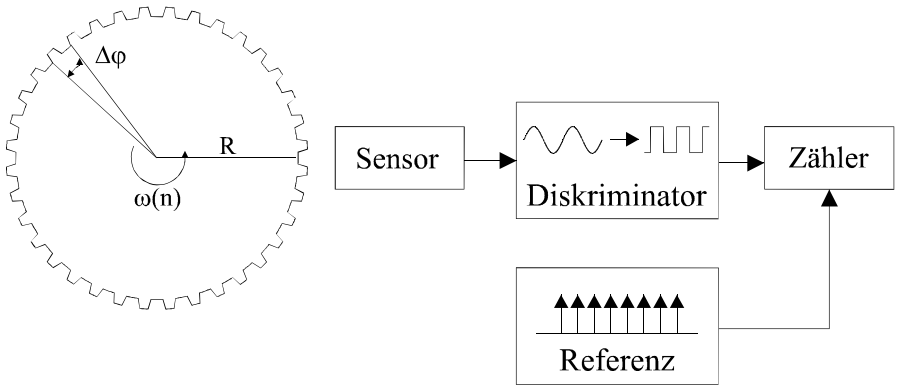


Abbildung 1 Meßsystem zur frequenzanalogen Drehzahlmessung

In Abbildung 1 ist das Prinzip der Meßanordnung dargestellt. Ein auf der Welle rotierendes Zahnrad bildet den Modulator zur Umformung der amplitudenanalogen Eingangsgröße Winkelgeschwindigkeit ω in ein frequenzanaloges Signal. Die Zahnhöhe wird durch einen Sensor erfaßt und in ein periodisches Signal umgewandelt.

In der Praxis werden bevorzugt magnetische Verfahren zur Drehzahlmessung eingesetzt, da sie berührungsfrei arbeiten und somit kein mechanischer Verschleiß auftritt. Grundsätzlich gibt es zwei physikalische Prinzipien, nach denen magnetische Sensoren arbeiten: die induktive Methode und die Hall-Messung.

Ein Hallsensor besteht aus zwei leitenden Sensorflächen, die von einem konstanten Strom durchflossen werden. Werden die Sensorflächen senkrecht hierzu von einem Magnetfeld durchflutet, kann man senkrecht zur Stromrichtung eine dem Magnetfeld proportionale Spannung, die sogenannte Hall-Spannung, abgreifen. Die beiden Sensorflächen sind im Zahnradsensor in Differenz geschaltet, um äußere Störeinflüsse, die durch Temperatur,

Fertigungstoleranzen und mechanische Verspannungen verursacht werden, zu reduzieren. Steht der einen Sensorfläche gerade ein Zahn, der anderen eine Lücke des Zahnrades gegenüber, so verstärkt sich die magnetische Induktion und es entsteht ein Differenzsignal. Dreht sich das Zahnrad, so ändert die Differenz ihre Polarität mit der Frequenz des Wechsels von Zahn auf Lücke und umgekehrt. Es ergibt sich ein sinusförmiger Verlauf der Differenzspannung. Das periodische Signal wird mit Hilfe eines Schmitt-Triggers in ein Rechtecksignal umgeformt.

Im Praktikumsversuch wird ein induktiver Sensor verwendet (siehe Abbildung 2). Dieser mißt das Magnetfeld nicht direkt, sondern den aus ihm resultierenden magnetischen Fluß. Ändert sich dieser, so wird in einer Spule gemäß dem Induktionsgesetz eine Spannung induziert, welche der Flußänderung proportional ist. Sensoren dieser Art werden als passive induktive Sensoren bezeichnet, da sie keine eigene Stromversorgung benötigen. Die induktive Messung hat den Vorteil, daß sie auch bei eingeschränkten Platzverhältnissen eingesetzt werden kann, während bei der Hall-Messung die Zähne für eine einwandfreie Messung sehr breit sein müssen und somit das Zahnrad eine gewisse Baugröße nicht unterschreiten darf. Zudem sind induktive Sensoren robuster und preiswerter als Hall-Sensoren. Andererseits eignen sich letztere eher für die Messung tiefer Frequenzen, da die Amplitude des Ausgangssignals nur durch die magnetische Flußdichte und die Stromstärke durch das Hall-Plättchen beeinflußt wird.

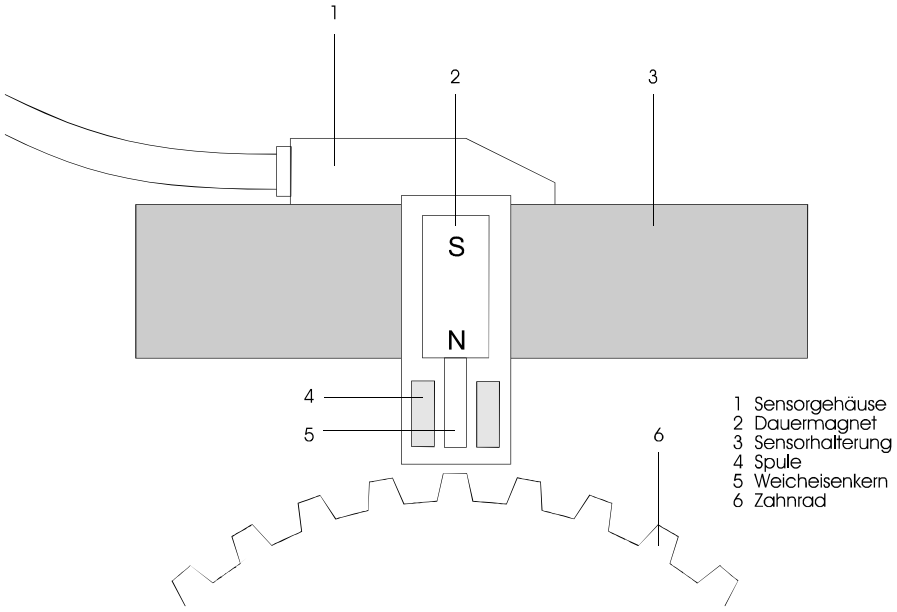


Abbildung 2 Aufbau des induktiven Zahnradsensors

Bei der induktiven Meßmethode ist die Amplitude des Signals von der Flußänderung bzw. von der Drehzahl abhängig. Deshalb ist eine Signalaufbereitung zur Erzielung einer konstanten Ausgangsamplitude erforderlich. Ein Schmitt-Trigger erzeugt ein Rechtecksignal, dessen Frequenz proportional zur Drehzahl ist. Frequenzanaloge Signale lassen sich mit sehr hoher Genauigkeit messen und bieten darüber hinaus folgende Vorteile:

- Sie sind unempfindlich gegen Änderungen der Leitungsparameter,
- werden weniger als analoge Signale durch elektromagnetische Einstreuungen gestört,
- können ohne Verlust an Genauigkeit verstärkt werden und
- lassen sich mit einfachen Hilfsmitteln verarbeiten.

Die Information über die Drehzahl liegt mit der Frequenz des Rechtecksignals analog vor. Die Signalform ist jedoch binär und eignet sich daher sehr gut zur Auswertung mit Hilfe eines Mikrocontrollers. Es sollen in diesem Versuch zwei Verfahren zur digitalen Drehzahlmessung vorgestellt werden, die Frequenzzählung und die Periodendauermessung. Diese beiden Verfahren werden im folgenden nun beschrieben.

2 Grundlagen

Die im Sensor induzierte Spannung wird zunächst zur Signalaufbereitung geführt. Dort wird das periodische Signal zur Erzielung einer konstanten Ausgangsamplitude über einen Schmitt-Trigger in ein Rechtecksignal gleicher Periodendauer umgewandelt, das anschließend im Mikrocontroller mit Hilfe eines Referenzzählers ausgezählt wird. Die Auswertung des Rechtecksignals durch den Mikrocontroller kann prinzipiell auf zwei unterschiedliche Arten erfolgen, entweder mittels Frequenzzählung oder durch Periodendauermessung. Bei beiden Verfahren läuft eine Impulsfolge der Frequenz f während eines Zeitintervalls t in einen Zähler ein und führt dort zu dem Zählerstand

$$N = f * t$$

Das pro Zahn überstrichene Winkelinkrement beträgt

$$\varphi_0 = \frac{2\pi}{Z} ,$$

wobei Z die Anzahl der Zähne auf dem Zahnrad ist. Die Winkelgeschwindigkeit ω berechnet sich aus der Differenz des überstrichenen Winkelintervalls $\varphi_2 - \varphi_1$ und der dafür benötigten Durchlaufzeit $t_2 - t_1$ zu

$$\omega_m = \frac{\varphi_2 - \varphi_1}{t_2 - t_1}$$

Auf die Unterschiede der beiden Verfahren soll nun näher eingegangen werden.

2.1 Digitale Frequenzmessung

Abbildung 3 zeigt das Strukturbild der Frequenzzählung. Bei der Frequenzzählung ist das Zeitintervall $T_{ref} = t_2 - t_1$, auch Torzeit genannt, fest vorgegeben. Gemessen wird der in dieser Zeit überstrichene Winkel $\varphi_m(i) = \varphi_2 - \varphi_1$, womit sich die Drehzahl berechnet zu

$$\omega_m(i) = \frac{\varphi_m(i)}{T_{ref}}$$

Zur Bestimmung des Winkels $\varphi_m(i)$ werden die steigenden (oder fallenden) Flanken des frequenzanalogen Signals während der bekannten Torzeit T_{ref} ausgezählt und führen zu dem Zählerstand N_F. Die Abtastung erfolgt bei der Frequenzzählung *zeitsynchron*, da nach jedem Zeitintervall T_{ref} ein digitaler Meßwert aufgenommen wird.

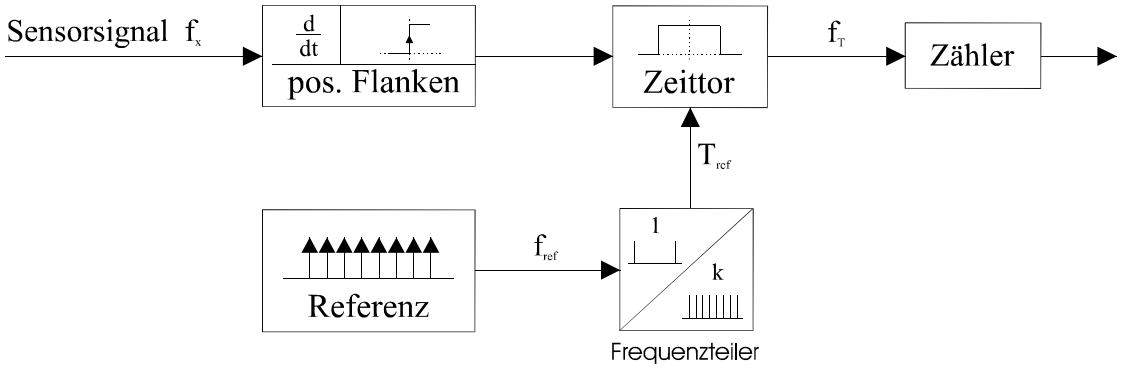


Abbildung 3 Strukturbild zur Frequenzzählung

Die Torzeit wird mittels digitaler Teilung der Referenzfrequenz f_{ref} durch einen Faktor k erzeugt. Die Verknüpfung des Torzeitsignals mit der zu messenden Frequenz f_x generiert das Signal f_T, das mit Hilfe eines Zählers ausgewertet wird. Mit $T_{ref} = k / f_{ref}$ ergibt sich die unbekannte Frequenz aus dem Zählerstand N_F zu

$$f_x = \frac{N_F}{T_{ref}} = N_F * \frac{f_{ref}}{k}$$

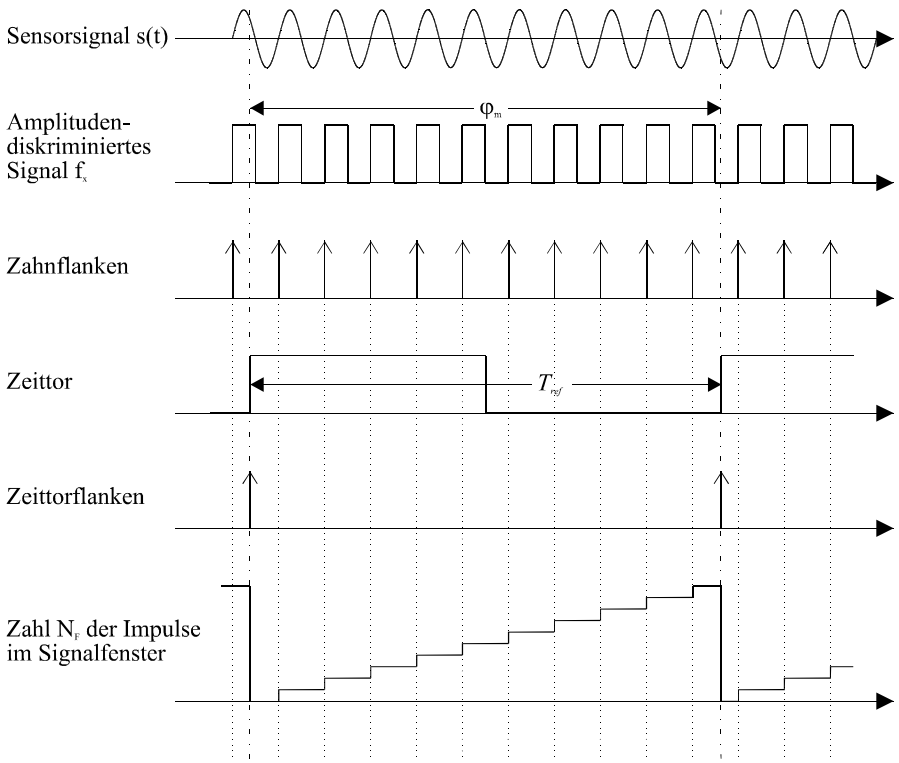


Abbildung 4 Digitalisierung frequenzanaloger Signale durch Frequenzzählung

Die erreichbare Frequenzauflösung, darunter versteht man die Frequenzänderung bei Abweichung des Zählerstandes um N = 1 Bit, hängt von der Torzeit (=Meßzeit) T_{ref} ab. Sie berechnet sich zu:

$$|\Delta f_x| = \frac{1}{T_{ref}}$$

Geht man von einer festen Torzeit aus, dann stellt der Betrag der Frequenzauflösung einen konstanten Wert dar. Aus dynamischen Gründen ist die Torzeit oft auf 1s beschränkt. Daraus folgt eine Auflösung von 1 Hz.

2.2 Digitale Periodendauermessung

Abbildung 5 zeigt das Strukturbild der Periodendauermessung. Bei der Periodendauermessung wird die kontinuierliche Zeit $T_m(i) = t_2 - t_1$ gemessen, die für einen diskreten Winkelschritt

$$\varphi_0 = \varphi_2 - \varphi_1 = \frac{2\pi}{Z}$$ Z: Anzahl der Zähne

benötigt wird. Dieser Winkel ist durch die Winkelteilung des Zahnrades fest vorgegeben.

$$\omega_m(i) = \frac{2\pi}{T_m(i) * Z} = \frac{\varphi_0}{T_m(i)}$$

Die Winkelgeschwindigkeit ω ist somit umgekehrt proportional zu der Periode T_m zwischen dem Durchlauf zweier aufeinanderfolgender Zähne am Sensor vorbei.

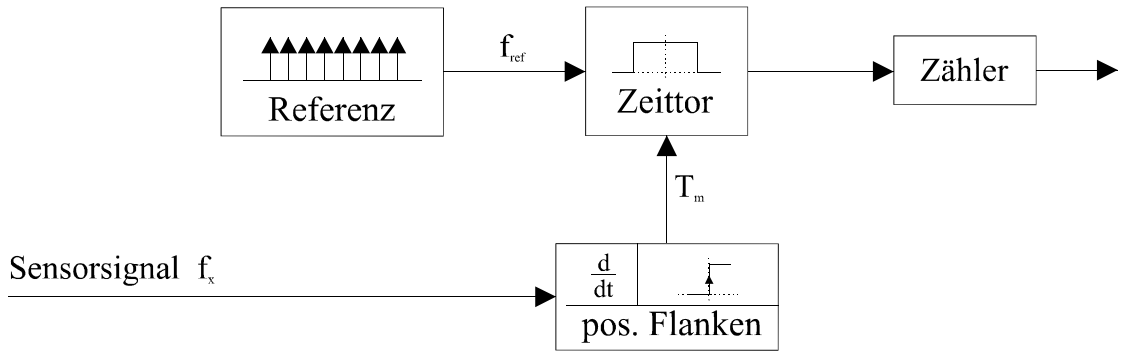


Abbildung 5 Strukturbild zur Periodendauermessung

Die Periodendauer T_m wird bestimmt, indem die von einer bekannten Referenzfrequenz f_{ref} während dieser Zeit in einen Zähler einlaufenden Impulse N_T gezählt werden. Aus

dem so gewonnenen Zählerstand N_T berechnet sich die Frequenz $f_x = 1/T_m$ nach folgender Formel:

$$f_x = \frac{f_{ref}}{N_T}$$

Die Abtastung erfolgt hierbei *winkelsynchron*, da mit jeder neuen Zahnflanke φ_0 ein digitaler Meßwert aufgenommen wird.

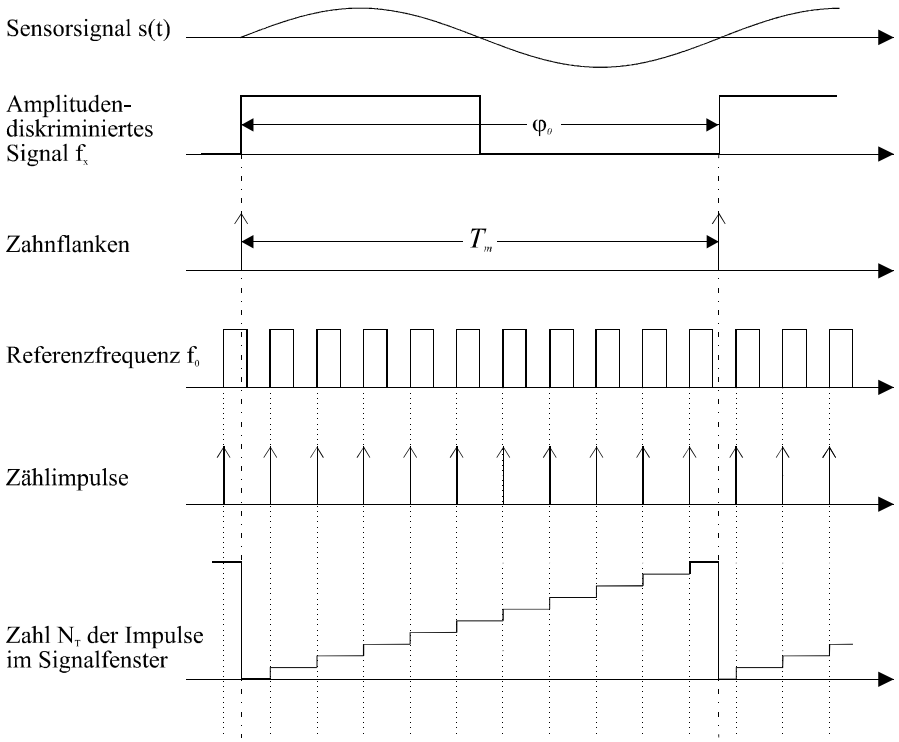


Abbildung 6 Digitalisierung frequenzanaloger Signale durch Periodendauermessung

Die Frequenzauflösung ist bei der Periodendauermessung nicht konstant, da hier die Frequenz umgekehrt proportional zum Zählerstand ist. Es ergibt sich:

$$|\Delta f_x| = \frac{f_{ref}}{N_T (N_T + 1)}$$

Für die häufig benutzte Referenzfrequenz $f_{ref} = 1$ MHz ergibt sich somit bei Messung eines 10 Hz Signals eine Frequenzauflösung von 10^{-4} Hz. Mißt man eine Frequenz von 1 kHz, so verschlechtert sich die Auflösung auf 1 Hz.

Die Periodendauermessung von Sensorzahn zu Sensorzahn mit einer festen Referenzfrequenz f_{ref} ist das Verfahren mit dem kürzesten Zeitverzug, um zu einer digitalen Zahl für die Drehzahl zu gelangen. Bei der Impulszählung zählt man mit den von der Meßeinrichtung kommenden Impulsen eine feste Referenzperiode T_{ref} aus. Der

Zeitverzug hierbei ist wesentlich größer, weshalb die Impulszählung nur bei frequenzanalogen Signalen hoher Frequenz angewendet wird.

Da die Frequenzauflösung bei der digitalen Frequenz- und Periodendauermessung im allgemeinen unterschiedlich ist, stellt sie ein wichtiges Kriterium zur Auswahl des entsprechenden Meßverfahrens dar. Ein weiterer Gesichtspunkt ist die Meßgenauigkeit der Verfahren, die im folgenden näher untersucht wird.

2.3 Absoluter Quantisierungsfehler

Die Abbildung einer analogen auf eine digitale Größe, in diesem speziellen Fall die Abbildung der Zeit bzw. der Frequenz auf einen Zählerstand, ist im allgemeinen mit einem Informationsverlust verbunden. Dieser macht sich im Quantisierungsfehler bemerkbar.

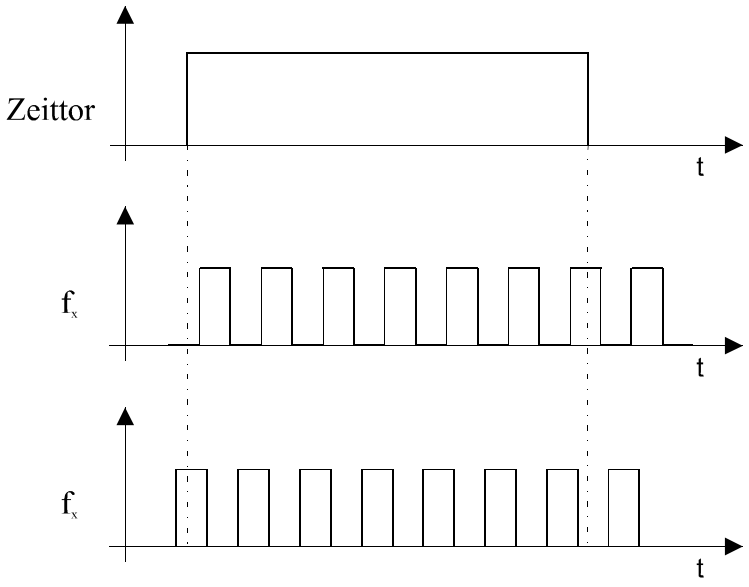


Abbildung 7 Quantisierungsfehler: je nach Lage von Zeit und Frequenzsignal zueinander können 6 oder 7 Vorderflanken gezählt werden

Aufgrund der Synchronisation der Sensorimpulse mit der Referenzfrequenz f_{ref} können je nach Phasenlage von Sensorsignal und Referenzimpulsen unterschiedlich viele Meßimpulse in das Meßintervall t fallen. Eine quantitative Betrachtung ergibt sich aus folgendem Bild:

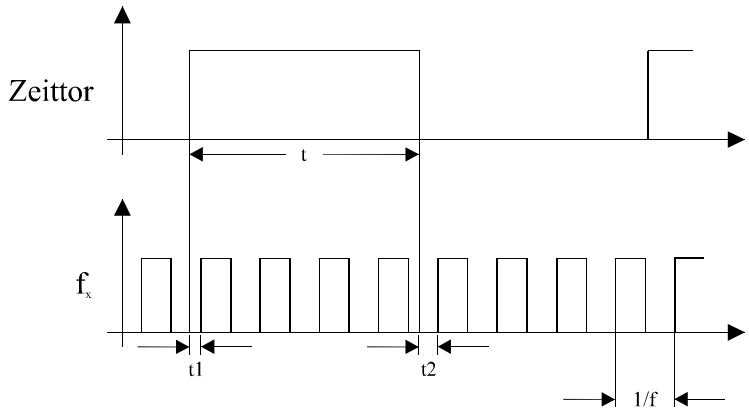


Abbildung 8 Definition beim Quantisierungsfehler

Das Zeitsignal eilt dem Frequenzsignal um die Zeit t_1 voraus. Am Ende des Meßintervalls ergibt sich eine Zeitverschiebung um t_2. Die Länge t des Zeitintervalls ergibt sich damit zu

$$t = N * \frac{1}{f} + (t_1 - t_2) = N_{soll} * \frac{1}{f}$$

N bedeutet die ganzzahlige Impulszahl, die vom Zähler erfaßt wird. N_{soll} ist eine gebrochene Zahl, die angibt, wie oft die reziproke Periodendauer $1/f$ in der Meßzeit t enthalten ist. Nach Multiplikation mit der Frequenz f erhält man die Beziehung

$$f * t = N + f * (t_1 - t_2) = N_{soll}$$

Der absolute Quantisierungsfehler berechnet sich aus der Differenz der ganzzahligen Impulszahl N und des Sollwertes N_{soll} zu

$$F_{Qabs} = N - N_{soll} = f * (t_2 - t_1)$$

Als Maximalwerte ergeben sich

$F_{Qabs} = +1$ für $t_1 = 0$ und $t_2 = 1/f$

$F_{Qabs} = -1$ für $t_1 = 1/f$ und $t_2 = 0$

Ist die Zeitdauer t gerade zufällig ein ganzzahliges Vielfaches der reziproken Frequenz $1/f$, so sind die Restzeiten t_1 und t_2 gleich groß. Der Quantisierungsfehler verschwindet dann, unabhängig vom Startzeitpunkt der Messung.

Durch eine Synchronisation von Zeit- und Frequenzsignal kann die Restzeit t_1 zu Null verfügt werden. Der absolute Quantisierungsfehler schwankt dann zwischen 0 und +1.

2.4 Relativer Quantisierungsfehler

Die beiden Verfahren beruhen darauf, daß die Bestimmung der Drehzahl ω entweder über das Zählen der Impulse eines Referenztaktes $f_0 = 1/T$ (Periodendauermesung) zwischen zwei Flanken des Sensorsignals geschieht, oder über die Anzahl der in einer Referenzperiode T_{ref} (Frequenzzählverfahren) einlaufenden Flankenimpulse des Sensorsignals. In beiden Fällen erfolgt eine Quantisierung mit entsprechenden Quantisierungsfehlern.

Im schlimmsten Fall kann die quantisierte Periodendauer $N * T_0$ durch die endliche Auflösung des Zählers um ein Quantisierungsintervall des Referenztaktes $T_0 = 1/f_0$ von der kontinuierlichen Periodendauer T_m zwischen zwei Flanken des Sensorsignals abweichen. Der relative Drehzahlfehler ergibt sich damit bei der Periodendauermessung zu

$$F_r = \frac{|\omega_q - \omega_m|}{\omega_m} = \left| \frac{\frac{\varphi_0}{N \cdot T_0} - \frac{\varphi_0}{T_m}}{\frac{\varphi_0}{T_m}} \right| = \left| \frac{T_m}{N \cdot T_0} - 1 \right|$$

Bei maximaler Abweichung um ein Zählintervall T_0

$$|T_m - N \cdot T_0| \le T_0$$

erhält man den relativen Quantisierungsfehler zu

$$F_r \le \frac{1}{N} = \frac{\omega_q}{\varphi_0 \cdot f_0} \ .$$

Beim Frequenzzählverfahren erhält man in der Referenzperiode T_{ref} N Sensorflankenimpulse. Jeder Flankenimpuls entspricht dem Durchlauf des Sensorrades durch das Winkelinkrement $\varphi_0 = 2\pi / Z$. Der wirklich in der Meßzeit T_{ref} überstrichene Winkel kann bis zu einem Winkelinkrement vom gemessenen Wert abweichen. Der Fehler wird dann

$$F_r = \frac{|\omega_q - \omega_m|}{\omega_m} = \left| \frac{\varphi_0 \cdot \frac{N}{T_{ref}} - \frac{\varphi_m}{T_{ref}}}{\frac{\varphi_m}{T_{ref}}} \right| = \left| \frac{\varphi_0 \cdot N - \varphi_m}{\varphi_m} \right| \approx \left| \frac{\varphi_0 \cdot N - \varphi_m}{\varphi_0 \cdot N} \right|$$

Bei maximaler Abweichung um ein Winkelinkrement φ_0

$$|\varphi_m - N \cdot \varphi_0| \le \varphi_0$$

erhält man den relativen Quantisierungsfehler zu

$$F_r \leq \frac{1}{N} = \frac{\varphi_0}{\omega_0 \cdot T_{ref}}$$

In Abbildung 9 ist der relative Quantisierungsfehler $1/N$ für verschiedene Referenzperiodendauern T_{ref} und Referenzzählfrequenzen f_0 im doppeltlogarithmischen Maßstab dargestellt.

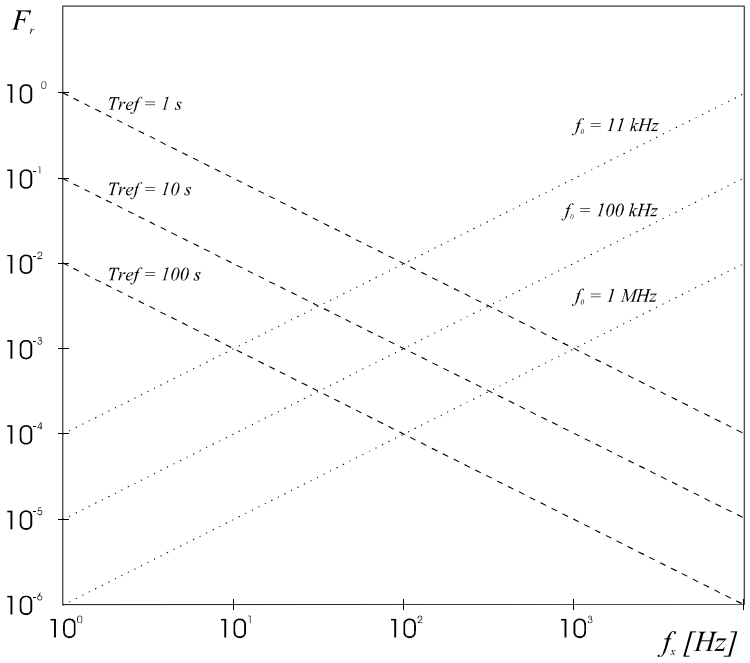


Abbildung 9 Relativer Quantisierungsfehler (--- Frequenzzählung, ⋯ Periodendauermessung)

Bei der Periodendauermessung nimmt der relative Quantisierungsfehler mit zunehmender Meßfrequenz zu, bei der Frequenzzählung dagegen nimmt er ab. Dieses Verhalten ist sehr anschaulich, da mit steigender Frequenz der Zählerstand N bei der Frequenzzählung größer wird und der Quantisierungsfehler somit immer geringer ins Gewicht fällt. Bei der Periodendauermessung nimmt die Zahl der zu zählenden Impulse mit kleiner werdender Frequenz zu. Für die Messung niedriger Frequenzen ist deshalb die Periodendauermessung günstiger.

Die Frequenzmessung bietet zusätzlich die Möglichkeit, durch Vergrößern des Zeitfensters den Fehler zu senken. Ein um den Faktor 10 größeres Zeittor ergibt dabei einen Fehler, der um den Faktor 10 kleiner ist.

2.5 Multiperiodendauermessung

Ähnlich wie bei der Frequenzmessung die Möglichkeit besteht, durch Variieren der Länge des Zeitfensters den relativen Quantisierungsfehler zu beeinflussen, erlaubt die Periodendauermessung durch Messen über mehrere Perioden eine Verringerung dieses Fehlers. Eine Messung über *n* Perioden senkt den Fehler um den Faktor *n*.

Dabei wird über eine ganzzahlige Anzahl $k = T_k \omega_q / \varphi_0$ von Perioden gemessen. Die Periode T_k für den Durchlauf des Sensorrades durch den Winkelschritt $k \cdot \varphi_0$ wird gemessen. Es ergibt sich der gleiche Fehler wie bei der Periodendauermessung, wenn man dort den festen, einfachen Winkelschritt φ_0 durch den *k*-fachen Winkelschritt $k \cdot \varphi_0$ ersetzt:

$$F_r = \frac{\left|\omega_q - \omega_m\right|}{\omega_m} \leq \frac{\omega_q}{k \cdot \varphi_0 f_0} = \frac{\omega_q}{\frac{T_k \omega_q}{\varphi_0} \varphi_0 f_0} = \frac{1}{T_k \cdot f_0}$$

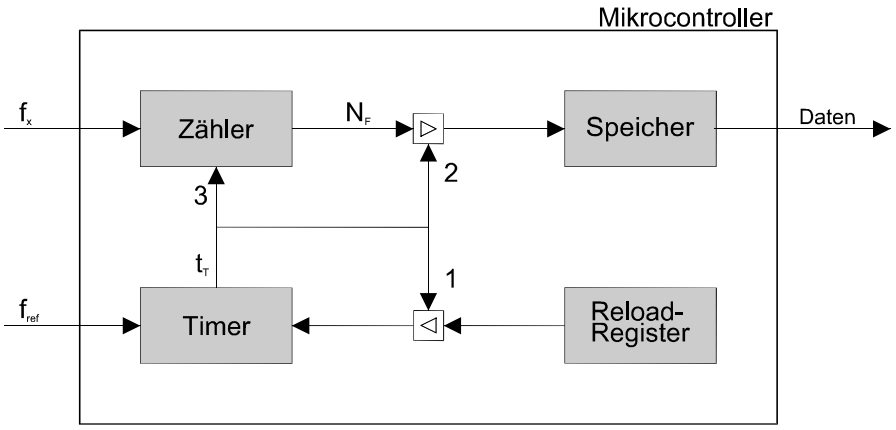
Die Multiperiodendauermessung hat ihre Stärken im mittleren Frequenzbereich. Dort ist der Fehler bei Umschalten zwischen Frequenzmessung und Einzelperiodendauermessung am größten. Bei niederen Frequenzen läuft der Impulszähler bei der Multiperiodendauer sehr schnell über, bei sehr hohen Frequenzen ist die digitale Frequenzmessung oft genauer. Umfaßt die zu messende Frequenz ein breites Spektrum, so wäre es am günstigsten, für die Bereiche niederer, mittlerer und hoher Frequenz zwischen den Methoden der Periodendauermessung, der Multiperiodendauermessung und der direkten Frequenzzählung umzuschalten. Allerdings ist zu berücksichtigen, daß das Meßinstrument hierbei eventuell eine Uminitialisierung durchzuführen hat, die eine gewisse Zeit benötigt. Während dieser Zeit können wertvolle Meßdaten verloren gehen. Das angesprochene Problem macht sich vor allem bei softwaregesteuerten Meßaufnehmern, wie z.B. Mikrorechnern, unangenehm bemerkbar, da hier die Uminitialisierung relativ lange dauert.

3 Drehzahlmessung mit dem Mikrocontroller

Im vorliegenden Praktikumsversuch soll die Auswertung des Sensorsignals mit Hilfe eines Mikrocontrollers durchgeführt werden. Dabei soll sowohl die digitale Frequenz- als auch die digitale Periodendauermessung, die beide, wie gesehen, äquivalent zu einer Drehzahlmessung sind, auf dem Mikrocontroller implementiert werden. Der im Praktikum verfügbare Mikrocontroller 80C517A stellt hierzu insgesamt vier Timer zur Verfügung, die auch als Zähler programmiert werden können. Welcher dieser Bausteine verwendet wird, hängt von der Art der zu lösenden Aufgabe ab, da jeder von ihnen spezielle Zusatzfunktionen besitzt.

3.1 Implementierung der digitalen Frequenzmessung

Für die digitale Frequenzmessung mit dem 80C517A wird ein Timer als Impulszähler verwendet. Er hat die Aufgabe, die am entsprechenden Mikrocontrollerpin auftretenden Vorderflanken des Meßsignals zu zählen. Der Zeitraum, in dem diese Zählung stattfindet, wird durch einen weiteren Timer bestimmt. Benutzt man dafür den Timer 2, so hat man den Vorteil, daß dieser im Autoreload-Modus betrieben werden kann.



1 Autoreload (Hardware)
2 Retten des Zählerstandes (ISR)
3 Rücksetzen des Zählers (ISR)

Abbildung 10 Hardwareaufwand bei der Frequenzmessung

Ein Überlauf von Timer 2, d.h. das Umkippen des Zählerstandes von FFFFh nach 0000h, löst den Autoreload und zusätzlich einen Interrupt aus. Der Überlauf entspricht dem Ende des gewünschten Zeitfensters. Der Autoreload, darunter versteht man das Laden des

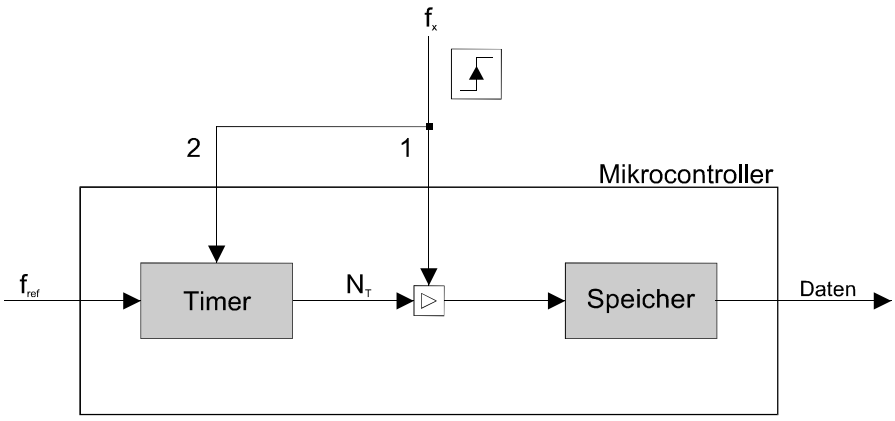
Timers mit einem Startwert aus dem zugehörigen Reload-Register, geschieht auf reiner Hardwarebasis und ist daher sehr schnell. Mit einer kleinen Zeitverzögerung auf den Autoreload erfolgt dann die Abarbeitung der Interrupt-Service-Routine (ISR).

3.2 Implementierung der digitalen Periodendauermessung

Je nach Art und Weise, wie die Kopplung zwischen Zeitsignal und Zeitmesser (Timer) erfolgt, unterscheidet man drei verschiedene Arten, auf die man beim 80C517A eine Periodendauermessung durchführen kann.

3.2.1 Rein interrupt-gesteuertes Verfahren

Bei diesem Verfahren löst jede Vorderflanke des zu messenden Frequenzsignals über einen Pin des Mikrocontrollers einen Interrupt aus. Die zugehörige Interrupt-Service-Routine (ISR) speichert den Zählerstand eines mitlaufenden Timers, der von einer Referenzquelle getaktet wird. Der erhaltene Zählerstand ist proportional zur Periodendauer.



1 Timerstand auslesen (ISR)
2 Rücksetzen des Timers (ISR)

Abbildung 11 Periodendauermessung mit Interrupts

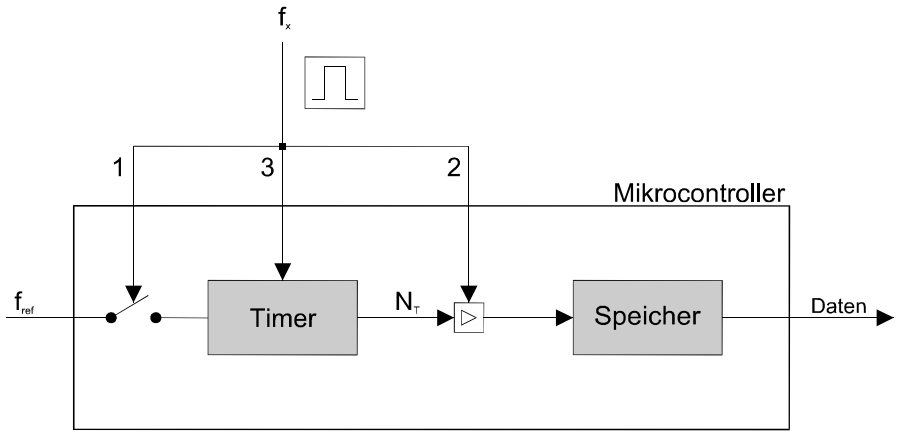
Der Vorteil dieser Meßmethode liegt in ihrem geringen Hardwareaufwand. Dadurch, daß nur ein einfacher Timer und ein externer Interrupt notwendig ist, ist das Verfahren sehr portabel, d.h. es kann auf vielen verschiedenen Mikrocontrollersystemen implementiert werden.

Als Nachteil ist die relativ hohe Ungenauigkeit anzusehen, da die Kopplung zwischen Meßsignal und Zählerstand softwaremäßig erfolgt. Die Zeitspanne zwischen dem Auslösen des Interrupts durch eine Vorderflanke und dem Aufruf bzw. der Abarbeitung der ISR schwankt zwischen 3 und 9 Maschinenzyklen. Dies hängt zum einen damit zusammen, daß der Interrupt asynchron auftritt und somit zuerst der aktuelle Befehlszyklus, der unterschiedlich lang sein kann, abgeschlossen werden muß. Zum anderen benötigt auch

das anschließende Laden des Programmzählers mit der Startadresse der ISR etwas Zeit. Da der Timer während dieser Zeit weiterläuft, entsteht ein Fehler, der einige Bits betragen kann.

3.2.2 Die gate-gesteuerte Methode

Das zu messende Signal steuert über einen (Transistor-) Schalter im Mikrocontroller die Verbindung zwischen der Referenzfrequenz und dem Timer. Bei einem High-Pegel ist der Schalter geschlossen, so daß der Timer läuft. Ein Übergang zum Low-Pegel öffnet den Schalter, der Timer stoppt. Gleichzeitig mit der fallenden Flanke wird ein Interrupt ausgelöst, der dafür sorgt, daß der Timer ausgelesen und anschließend zurückgesetzt wird.



1 Schaltersteuerung (Hardware)
2 Timerstand auslesen (ISR)
3 Rücksetzen des Timers (ISR)

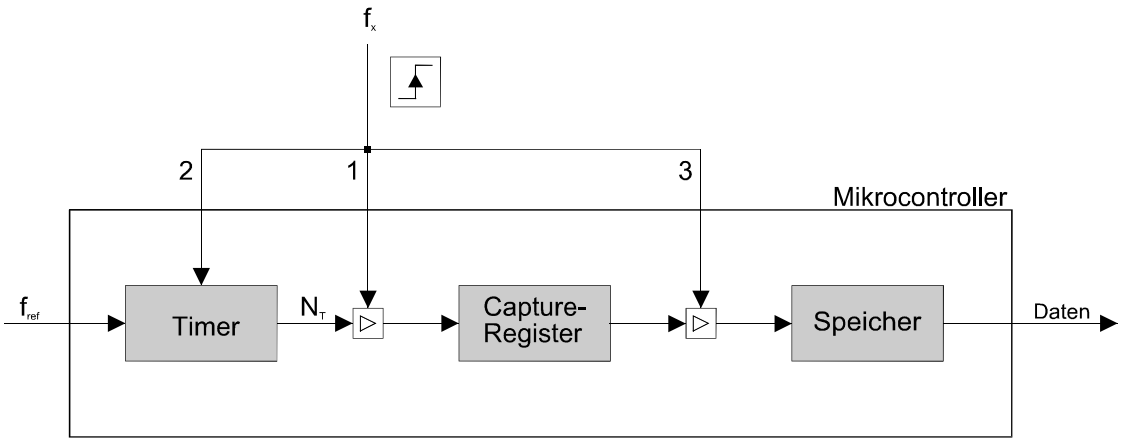
Abbildung 12 Gate-gesteuerte Periodendauermessung

Durch die Hardwarekopplung von Meßsignal und Zähler ist die Abbildung der Periodendauer auf den Zählerstand sehr genau. Ein Fehler in der Größenordnung wie bei der interrupt-gesteuerten Periodendauermessung tritt hier nicht auf. Dieser Vorteil wird durch die etwas aufwendigere Hardware erkauft. Gate-gesteuerte Timer sind jedoch schon in den einfachen und preiswerten Mikrocontrollern vorhanden. Bei gleichzeitiger Messung mehrerer Frequenzen nimmt der Umfang der benötigten Mittel allerdings stark zu, da jede zu messende Frequenz ihr eigenes Gate mit dem dazugehörigen Timer benötigt.

Das Verfahren kann nur dann direkt angewandt werden, wenn das Tastverhältnis des zu messenden Rechtecksignals bekannt ist, da nur ein Teil der Periodendauer (High-Pegel) ausgemessen wird. Ist dies nicht der Fall, so ist eine Frequenzteilung um den Faktor 2 mit Hilfe eines externen Flip-Flops durchzuführen. Die Länge des High-Pegels entspricht dann der zu messenden Periodendauer.

3.2.3 Verwenden eines Capture-Registers

Im Praktikumsversuch soll die Periodendauermessung mit einem speziellen Feature des 80C517A durchgeführt werden, dem Capture-Register. Eine steigende (oder fallende) Flanke an einem Pin des Mikrocontrollers, dem ein Capture-Register zugeordnet ist, bewirkt, daß der aktuelle Timerstand in dieses Register übernommen wird. Dieser Vorgang erfolgt hardwaregesteuert und ist somit sehr schnell. Der Mikrocontroller kann derart programmiert werden, daß die Flanke am Pin zusätzlich einen Interrupt auslöst.



1 Timerstand auslesen (Hardware)
2 Rücksetzen des Timers (ISR)
3 Inhalt des Capture-Registers retten (ISR)

Abbildung 13 Periodendauermessung mit dem Capture-Register

Diese Meßmethode der Periodendauermessung stellt eine Mischung aus den beiden bisher vorgestellten Verfahren dar, wobei deren Nachteile fast vollständig vermieden werden können. So wird durch die schnelle Übernahme des Meßwertes in das Capture-Register eine Meßunsicherheit wie beim Interruptverfahren umgangen. Da immer über eine komplette Periodendauer gemessen wird, entfällt hier auch das Problem der Frequenzteilung, wie es vom Gate-Verfahren her bekannt ist. Noch genauer ist diese Methode, wenn man auf ein softwaremäßiges Rücksetzen des Zählerstandes verzichtet und statt dessen den vorher gespeicherten Stand des Zählers aus der letzten Periode vom aktuellen Zählerstand abzieht, den Zähler also als sogenannten Ringzähler verwendet.

Der 80C517A enthält einen Timer (Timer 2), an dem bis zu fünf Capture-Register betrieben werden können. Dies bedeutet, daß es möglich ist, gleichzeitig bis zu fünf verschiedene Drehzahlen mit diesem Baustein zu messen.

4 Praktikumsversuch

Am Versuchstag steht Ihnen ein kompletter Aufbau zur Drehzahlmessung zur Verfügung. Desweiteren ist ein PC eingerichtet, auf dem Sie die notwendige Software für die Drehzahlmessung entwickeln und testen können. Die Entwicklungsumgebung und der Versuchsaufbau werden im folgenden beschrieben.

4.1 Versuchsaufbau

Der Arbeitsplatz zur Durchführung der Drehzahlmessung besteht aus einem PC, dem Mikrocontrollerboard mit der Anzeigeeinheit sowie einem Gleichstrommotor mit ferromagnetischem Zahnrad.

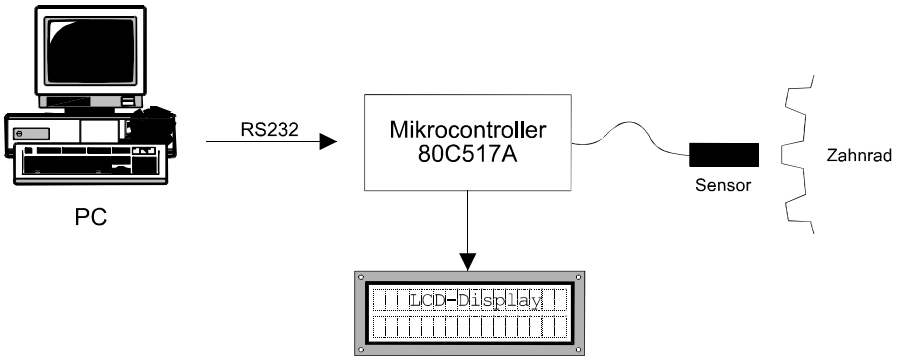


Abbildung 14 Versuchsaufbau zur Drehzahlmessung

Der Gleichstrommotor, der an einer regelbaren Spannungsquelle betrieben wird, liefert als Meßgröße die Drehzahl. Ein Zahnradsensor ist mit Hilfe des Zahnrades, das am Motor angebracht ist, in der Lage, daraus ein rechteckförmiges Frequenzsignal zu erzeugen. Dieses wird zur Anpassung an die zulässige Eingangsspannung des Mikrocontrollers zur Benutzerplatine geführt. Von dort aus gelangt es an die Portpins P1.1 und P3.4 des Mikrocontrollers. Der Eingangspin P3.4 kann als Impulszählereingang von Timer0 programmiert werden, womit sich eine Frequenzmessung realisieren läßt. Eine Periodendauermessung ist mit dem Signaleingang P1.1 möglich, da hierüber das Capture/Compare-Register 1 von Timer2 aktiviert wird. Ebenso ist es an diesem Pin möglich, einen externen Interrupt auszulösen. Die Ausgabe des Meßergebnisses erfolgt auf einem LCD, das über den Adreß-/Datenbus des Mikrocontrollers angesprochen werden kann.

4.2 Darstellung der Drehzahl auf dem Display

Die Drehzahl liegt zunächst als Binärzahl vor. Diese wird in eine vierstellige BCD-Zahl umgewandelt. Diese Umwandlung erledigt die Unterroutine *bin2bcd.a51* (siehe Anhang). Die vier Dezimalstellen können anschließend auf dem Display dargestellt werden.

Um ein Zeichen auf dem Display darzustellen, muß zuerst die gewünschte Adresse im Anzeigen-RAM gewählt werden. Der Befehl zum Setzen einer Adresse hat das Format 1xxx xxxx. Das erste Zeichen auf dem Display besitzt z.B. die Adresse 00h, der zugehörige Befehl lautet deshalb 80h. Anschließend kann der Code für das darzustellende Zeichen in das Anzeigen-RAM geschrieben werden. Jeder Schreib-/Lesevorgang hat ein Autoinkrement des Adreßzählers zur Folge, so daß alle nachfolgenden Daten in das Anzeigen-RAM fließen.

Die Auswahl zwischen Befehls- und Datenbytes erfolgt mit dem Signal RS. Ist es LOW, können Befehle an den Grafikcontroller gesendet oder der Adreßzähler und das Busy-Flag gelesen werden. Ist es HIGH, werden Datenbytes in den Anzeigenspeicher geschrieben oder aus ihm gelesen.

RS	RW	Vorgang
0	0	Schreiben von Befehlen
0	1	Lesen des Adreßzählers und des Busy-Flags
1	0	Schreiben von Daten
1	1	Lesen von Daten

Tabelle 1 Die Steuersignale RS und RW

Bevor der Mikrocontroller Daten an den Grafikcontroller schreiben kann, muß er prüfen, ob dieser zum Empfang bereit ist. Dies geschieht mit dem Busy-Flag. Ist dieses logisch 0, so können Daten geschrieben werden. Der Mikrocontroller wartet also auf die Bereitschaft der Anzeige und übergibt dann die Daten. Dies bringt einen entscheidenden Nachteil mit sich. Da der Grafikcontroller wesentlich langsamer arbeitet als der Mikrocontroller, wirkt die Anzeige wie ein Bremsklotz. Der Mikrocontroller verliert dadurch, daß er auf die Bereitschaft der Anzeige warten muß, wertvolle Rechenzeit. Diese beträgt pro gesendetem Byte ca. 40 µs und bei Aktualisierung der ganzen Anzeige bereits über 1 ms. In dieser Zeit kann der Mikrocontroller keine anderen Aufgaben wahrnehmen, da er in einer Warteschleife „hängt“. Da die Hauptaufgabe des Mikrocontrollers die digitale Signalverarbeitung und nicht die Ansteuerung des Displays ist, wird ein eleganterer Weg beschritten:

Im internen Datenspeicher des Mikrocontrollers wird ein Bereich reserviert, der ein genaues Abbild des Anzeigen-RAMs ist, so daß jedem Anzeigenplatz auf dem Display eine Speicheradresse im internen RAM zugeordnet ist. Es wird der Bereich 60h bis 7Fh in den unteren 128 Byte des Datenspeichers verwendet.

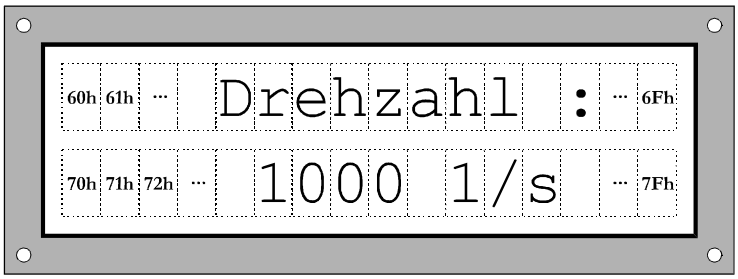


Abbildung 15 Abbildung der Adressen des internen Datenspeichers auf das Display

Mit Hilfe des Timer1-Interrupts werden die Zeichencodes vom internen Datenspeicher in das Anzeigen-RAM geschrieben. Mit jedem weiteren Interruptaufruf wird somit das nächste Zeichen in die Anzeige geschrieben. Dazu wird ein Zeiger verwendet, der jeweils auf das aktuelle Zeichen im internen Datenspeicher zeigt und mit jedem Interrupt erhöht wird. Sind alle Zeichen auf der Anzeige dargestellt, muß der Zeiger wieder auf die Adresse des ersten Zeichens zurückgesetzt werden. Die Zeitdauer zwischen den Timer1-Interrupts wird so gewählt, daß sich die gesamte Anzeige mit 100 Hz aufbaut. Da die Anzeige 32 Zeichen darstellen kann und für den Sprung von der ersten zur zweiten Zeile und umgekehrt zusätzlich zwei Befehlsbytes benötigt werden, muß die Interruptroutine ca. 3400 mal in der Sekunde aufgerufen werden. Dies bedeutet, daß alle 300 µs ein Interrupt ausgelöst wird. Da die Anzeige eine Verarbeitungszeit von etwa 40 µs besitzt, kann sie beim nächsten Interrupt wieder Daten aufnehmen.

Das Hauptprogramm sendet nun die Daten nicht an die Anzeige, sondern beschreibt ohne Wartezeiten das reservierte interne RAM des Mikrocontrollers. Den Rest erledigt die Interruptroutine, und der Mikrocontroller kann sich der digitalen Signalverarbeitung widmen. Die Display-Refresh-Routine *lcd.a51* finden Sie im Anhang abgedruckt.

Zu Beginn muß das Display initialisiert werden. Dies geschieht, indem nacheinander die folgenden Befehle mit den entsprechenden Wartezeiten an das Display geschrieben werden. Die Adresse für Befehle lautet FE00h:

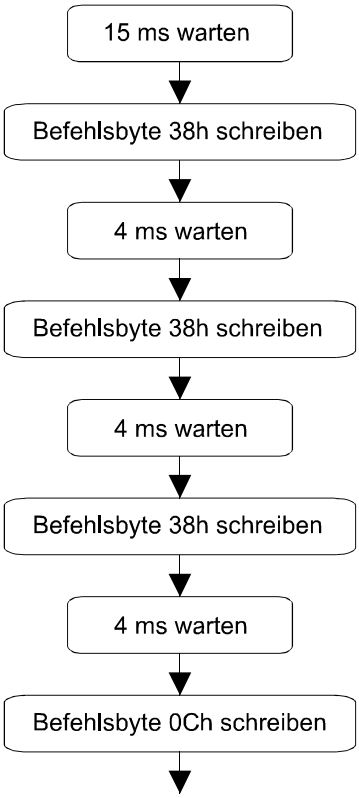


Abbildung 16 Initialisierung des Displays

5 Aufgaben

Im Rahmen dieses Praktikumsversuchs sind die folgenden Aufgaben zu bearbeiten. Die Aufgaben 1-3 dienen der Vorbereitung, während die Aufgaben 4-6 am Versuchstag selbst erledigt werden. Um die Ihnen zur Verfügung stehende Zeit am Versuchstag effektiv nutzen zu können, sollten Sie bereits zu Hause die Lösung der folgenden Programmieraufgaben vorbereiten. Die Programme sollen in der Programmiersprache Assembler erstellt werden. Für jeden Aufgabenteil ist ein Projekt vorhanden, in dem die notwendigen Einstellungen zur Codegenerierung durchgeführt wurden. Bei der Bearbeitung der Aufgaben öffnen Sie diese Projekte und fügen ihnen Ihre eigenen Module hinzu. Die zu messenden Frequenzen bewegen sich im Bereich von ca. 500Hz bis 2kHz. Das Zahnrad besitzt 50 Zähne.

Aufgabe 1

a) Wie lange dauert ein Maschinenzyklus?

b) Welche maximale Torzeit kann mit dem Timer2 realisiert werden? Wie kann man größere Torzeiten auf dem Mikrocontroller realisieren?

c) Schauen Sie sich die Routine *lcd.a51* zur Aktualisierung des Displays an. Welche Adressierungsarten kommen hier zum Einsatz und welche Aufgaben haben sie?

Aufgabe 2

a) Welche Vorteile besitzt die Periodendauermessung gegenüber der Frequenzmessung und wodurch ergeben sich Nachteile in bezug auf die Meßgenauigkeit?

b) Mit welcher Methode wird die Periodendauermessung auf dem Mikrocontroller implementiert und welche Besonderheit des Mikrocontrollers wird dabei ausgenutzt?

Aufgabe 3

a) Was ist der Grund für die Quantisierung des Drehzahlsignals bei der digitalen Drehzahlmessung?

b) Wie beeinflußt die Anzahl der Zähne auf dem Zahnrad die Meßgenauigkeit bei beiden Verfahren?

Aufgabe 4

Für die Darstellung der Meßergebnisse auf der Anzeigeneinheit muß die 16bit-Binärzahl zunächst in eine vierstellige BCD-Zahl umgewandelt werden. Dies übernimmt das Unterprogramm *bin2bcd.a51*. Die vier Dezimalstellen der BCD-Zahl müssen anschließend in das gespiegelte Anzeigen-RAM geschrieben werden. Die Display-Refresh-Routine *lcd.a51* hat die Aufgabe, die Anzeigeneinheit ständig zu aktualisieren. Die beiden Unterprogramme *bin2bcd.a51* und *lcd.a51* befinden sich im Verzeichnis *c:\versuch1\aufgabe4* und sind im Anhang abgedruckt. Das LCD muß zu Beginn, wie in Kapitel 4.2 beschrieben, initialisiert werden.

a) Schreiben Sie ein Unterprogramm *lcdinit.a51*, das die Anzeigeneinheit initialisiert. Für die benötigten Wartezeiten steht Ihnen die Unterroutine *delay.a51* zur Verfügung. Die Wartezeit in Millisekunden wird ihr in den Registern R6 (höherwertiges Byte) und R7 (niederwertiges Byte) übergeben.

b) Erstellen Sie ein Unterprogramm *display.a51*, das eine 4-stellige BCD-Zahl auf dem Display ausgibt. Beachten Sie, daß das gegebene Unterprogramm *bin2bcd.a51* die vier Digits der darzustellenden BCD-Zahl in den Registern R4 und R5 zurückgibt. Im unteren Halbbyte von R4 steht Digit0, im oberen Halbbyte Digit1. Im unteren Halbbyte von R5 steht Digit2 und im oberen Halbbyte Digit3. Die Digits in den oberen Halbbytes können mit dem Befehl swap A in das untere Halbbyte kopiert werden. Das obere Halbbyte kann danach ausgeblendet werden. Die BCD-Zahl muß anschließend noch durch Addition von 30h in den entsprechenden ASCII-Code umgewandelt werden.

c) **Zusatzaufgabe:** Überlegen Sie sich, wie Sie die führenden Nullen bei der Ausgabe des Meßergebnisses unterdrücken können.

Aufgabe 5

Zur Realisierung der digitalen Frequenzmessung mit Hilfe des Mikrocontrollers 80C517A soll Timer0 als Impulszähler verwendet werden, für die Festlegung des Zeittors von einer Sekunde wird Timer2 im Autoreload-Modus verwendet. Timer1 ist für den Display-Refresh zuständig und muß so programmiert werden, daß der zugehörige Interrupt mit einer Frequenz von 3400 Hz aufgerufen wird.

- Stellen Sie ein Flußdiagramm zur Durchführung der digitalen Frequenzmessung auf. Das Hauptprogramm soll zunächst alle notwendigen Initialisierungen durchführen und danach das Meßergebnis in einer Endlosschleife ausgeben. Die Aufgabe der Timer2-ISR besteht in der Meßwertaufnahme und der Binär-BCD-Wandlung.

- Setzen Sie das Flußdiagramm in ein lauffähiges Programm um. Öffnen Sie hierzu das Projekt *c:\versuch1\aufgabe5\freqmess.prj*. Fügen Sie dem Projekt die erstellten

Unterprogramme aus Aufgabe 4 hinzu und binden Sie die externen Labels ´bin2bcd´, ´timer1_isr´ und ´delay´ sowie die Variablen ´anzeigen_ram_ptr´ und ´lock´ in das Hauptprogramm ein.

Damit die Display-Refresh-Routine korrekt funktioniert, muß die Variable *anzeigen_ram_ptr* mit dem Wert 60h initialisiert werden und das Bit *lock* muß zu Beginn gelöscht sein.

Aufgabe 6

Für die digitale Periodendauermessung soll das Capture-Register CC1 von Timer2 verwendet werden. Gleichzeitig mit dem Capture-Aufruf wird der externe Interrupt EX4 aktiviert. Timer1 liefert den Takt für die Aktualisierung des Displays.

a.) Stellen Sie ein Flußdiagramm zur Durchführung der digitalen Periodendauermessung auf. Die Ausgabe des Meßergebnisses soll nicht vom Hauptprogramm, sondern von der Timer0-ISR durchgeführt werden, die bewirkt, daß die Ausgabe eines neuen Meßwertes unabhängig von der Drehzahl mit 2 Hz erfolgt. Die Aufgabe der EX4-ISR besteht in der Meßwertaufnahme und der Binär-BCD-Wandlung.

- Schreiben Sie ein Unterprogramm *per2freq.a51*, das die Periodendauer in eine Frequenz (Zähne pro Sekunde) umwandelt. Benutzen Sie dazu die Multiplikations-/Divisions-Einheit des Mikrocontrollers. Die Periodendauer soll in den Registern R6 (niederwertiges Byte) und R7 (höherwertiges Byte) übergeben werden und das Ergebnis muß ebenfalls in diesen Registern (R6, niederwertiges Byte und R7, höherwertiges Byte) zurückgegeben werden.

- Setzen Sie das Flußdiagramm in ein lauffähiges Programm um. Öffnen Sie hierzu das Projekt *c:\versuch1\aufgabe6\permess.prj*. Fügen Sie dem Projekt analog zu Aufgabe 5 die benötigten Unterprogramme hinzu.

b.) Die Drehzahl soll nun in der Einheit Umdrehungen/min auf dem Display dargestellt werden. Führen Sie die notwendigen Änderungen durch.

6 Programmvorlagen

```
;*******************************************************************************
;*         Praktikum    Mikrocontroller und digitale Signalprozessoren          *
;*                                                                              *
;*                                 Modul    bin2bcd                            *
;*                                                                              *
;*                                                                              *
;*        Umwandlung einer 16-Bit Binärzahl in eine 4-stellige BCD-Zahl         *
;*                                                                              *
;*        EINGABE:     R6: niederwertiges Byte                                  *
;*                     R7: höherwertiges Byte                                   *
;*                                                                              *
;*        AUSGABE:     R4 unteres Halbbyte:   Digit0                            *
;*                     R4 oberes  Halbbyte:   Digit1                            *
;*                     R5 unteres Halbbyte:   Digit2                            *
;*                     R5 oberes  Halbbyte:   Digit3                            *
;*                                                                              *
;*******************************************************************************
$NOMOD51                        ;Registerbelegung des 8051 abschalten
$include(REG517A.INC)           ;SFR Symbole von 80C517A benutzen

NAME bin2bcd


;*******************************************************************************
;*                             Public-Definitionen                             *
;*******************************************************************************
PUBLIC  bin2bcd


;*******************************************************************************
;*                             Segment-Definitionen                            *
;*******************************************************************************
PROG    SEGMENT  CODE           ;Codesegment definieren

using   0                       ;Registerbank 0 auswählen


;*******************************************************************************
;*                             Code-Segment                                    *
;*******************************************************************************
        RSEG   PROG             ;Hier beginnt der Programmcode der Unterroutine


bin2bcd:
   mov   R4,#00h                ;Ergebnis-Register zurücksetzen. Die Bits werden
   mov   R5,#00h                ;von rechts nach links in R5:R4 geschoben

   mov   R2,#0Fh                ;Start mit Bit 15
nextbit:
   mov   A,R2
   jz    LSB                    ;Bit 0 erreicht?
   anl   A,#08h                 ;falls R2<8, dann
   jz    rotateR6               ;rotiere R6
   mov   A,R7                   ;sonst
   rlc   A                      ;rotiere R7 links durch Carry
   mov   R7,A
   jmp   testbit
```

```
rotateR6:
   mov   A,R6                    ;rotiere R6 links durch Carry
   rlc   A
   mov   R6,A
testbit:
   jnc   mal2                    ;teste Carry-Flag auf 0 oder 1
   inc   R4                      ;C=1: erhöhe R4 und multipliziere R4 und R5 mit 2
                                 ;C=0: multipliziere R4 und R5 mit 2
mal2:
   mov   A,R4
   add   A,R4
   da    A                       ;Pseudo-Tetraden korrigieren
   mov   R4,A
   mov   A,R5
   addc  A,R5
   da    A                       ;Pseudo-Tetraden korrigieren
   mov   R5,A
   dec   R2
   jmp   nextbit

lsb:
   mov   A,R6                    ;Bit 0 bearbeiten, das am Ende der Schiebe-
   jnb   ACC.7,return            ;operation in Bit7 von R6 steht
   inc   R4                      ;falls Bit gesetzt, R4 um eins erhöhen

return:
   ret

END
```

Listing 1 Modul bin2bcd.a51

```
;*******************************************************************************
;*         Praktikum    Mikrocontroller und digitale Signalprozessoren         *
;*                                                                             *
;*                                  Modul    lcd                               *
;*                                                                             *
;*                                                                             *
;*        Die Aktualisierung des Displays geschieht mit Hilfe des Timer1-      *
;*        Overflow-Interrupts. Die ISR liest bei jedem Aufruf ein Zeichen      *
;*        aus dem internen Datenspeicher des Mikrocontrollers (60h-7Fh) und    *
;*        schreibt es in das Anzeigen-RAM des Displays.                        *
;*                                                                             *
;*        Anzeigen-RAM-Adressen des Displays :                                 *
;*        Zeile 1:  0x00 - 0x0F   (16 Zeichen)                                 *
;*        Zeile 2:  0x40 - 0x4F   (16 Zeichen)                                 *
;*                                                                             *
;*******************************************************************************
$NOMOD51                          ;Registerbelegung des 8051 abschalten
$include(REG517A.INC)             ;SFR Symbole von 80C517A benutzen

NAME lcd


;*******************************************************************************
;*                               Public-Definitionen                           *
;*******************************************************************************
PUBLIC  timer1_isr
PUBLIC  anzeigen_ram_ptr, lock


;*******************************************************************************
;*                               Konstanten                                    *
;*******************************************************************************
WR_Control     EQU   0FE00h       ;Write Control Register
WR_Data        EQU   0FE01h       ;Write Data Register
LCD_Zeilen     EQU   2            ;Anzahl der Zeilen des Displays
LCD_Spalten    EQU   16           ;Anzahl der Spalten des Displays
anzeigen_ram   EQU   60h          ;Anfangsadresse des Speicherbereichs im internen
                                  ;RAM, in das das Anzeigen-RAM gespiegelt wird
                                  ;Speicherbereich (60h-7Fh)

;*******************************************************************************
;*                               Segment-Definitionen                          *
;*******************************************************************************
PROG    SEGMENT  CODE             ;Codesegment definieren
VAR     SEGMENT  DATA             ;Datensegment definieren
BITS    SEGMENT  BIT              ;Bitsegment definieren

using   0                         ;Registerbank 0 auswählen


;*******************************************************************************
;*                               Daten-Segment                                 *
;*******************************************************************************
        RSEG   VAR
anzeigen_ram_ptr:   DS   1        ;Zeiger auf Puffer für Anzeige


;*******************************************************************************
;*                                Bit-Segment                                  *
;*******************************************************************************
        RSEG   BITS
lock:   DBIT   1                  ;Schalter zur Verriegelung des Zeilenumbruchs
```

```
;*******************************************************************************
;*                              Code-Segment                                   *
;*******************************************************************************
        RSEG   PROG               ;Hier beginnt der Programmcode der Unterroutine


timer1_isr:

   push  ACC                              ;Register sichern
   push  AR0
   push  DPL
   push  DPH
   push  PSW

zeilenumbruch:
   mov   A,anzeigen_ram_ptr
   cjne  A,#anzeigen_ram+LCD_Spalten,home  ;Ende von Zeile 1 erreicht?
   jb    lock,home                        ;Zeilenumbruch verriegelt?
   mov   A,#40h+80h                       ;Akku mit Anzeigen-RAM-Adresse laden
   mov   dptr,#WR_Control                 ;Adreßzähler des Displays
   movx  @dptr,A                          ;auf 2.Zeile, 1.Spalte setzen
   setb  lock                             ;Zeilenumbruch verriegeln
   ljmp  reload

home:
   mov   A,anzeigen_ram_ptr               ;Ende von Zeile 2 erreicht?
   cjne  A,#anzeigen_ram+(LCD_Spalten*LCD_Zeilen),writedata
   mov   A,#00h+80h                       ;Akku mit Anzeigen-RAM-Adresse laden
   mov   dptr,#WR_Control                 ;Adreßzähler des Displays
   movx  @dptr,A                          ;auf 1.Zeile, 1.Spalte setzen
   mov   anzeigen_ram_ptr,#anzeigen_ram   ;Zeiger zurücksetzen
   ljmp  reload

writedata:
   mov   R0,anzeigen_ram_ptr
   mov   A,@R0                            ;Byte aus internem Datenspeicher holen
   mov   dptr,#WR_Data                    ;Datenbyte in Anzeigen-RAM
   movx  @dptr,A                          ;des Displays schreiben
   inc   anzeigen_ram_ptr                 ;Zeiger auf nächste Speicherstelle
   clr   lock                             ;Zeilenumbruch freigeben

reload:
   clr   TR1                              ;Timer1 stoppen
   mov   TL1,#47h                         ;Timer1 neu laden,
   mov   TH1,#0FEh                        ;da nicht im Reload-Modus
   setb  TR1                              ;Timer1 starten

   pop   PSW                              ;Register wiederherstellen
   pop   DPH
   pop   DPL
   pop   AR0
   pop   ACC

   reti

END
```

Listing 2 Modul lcd.a51

```
;*******************************************************************************
;*         Praktikum    Mikrocontroller und digitale Signalprozessoren          *
;*                                                                              *
;*                                                                              *
;*                                  Modul    delay                              *
;*                                                                              *
;*              Zeitschleife für n * 1ms                                        *
;*                                                                              *
;*              EINGABE:    R6:  Highbyte der Zahl n                            *
;*                          R7:  Lowbyte  der Zahl n                            *
;*              AUSGABE:    keine                                               *
;*              verändert Register R5                                           *
;*                                                                              *
;*******************************************************************************
NAME delay


;*******************************************************************************
;*                              Public-Definitionen                             *
;*******************************************************************************
PUBLIC  delay


;*******************************************************************************
;*                              Segment-Definitionen                            *
;*******************************************************************************
PROG    SEGMENT  CODE             ;Codesegment definieren

using   0                         ;Registerbank 0 auswählen


;*******************************************************************************
;*                              Code-Segment                                    *
;*******************************************************************************
        RSEG   PROG               ;Hier beginnt der Programmcode der Unterroutine

delay:
   inc   R6
de:
   mov   R5,#250
wait1ms:                          ;Taktfrequenz 18 MHz  -->  Zyklus 0,667 µs
   nop                            ; 1
   nop                            ; 1
   nop                            ; 1
   nop                            ; 1
   djnz  R5,wait1ms               ; 2 = 6 Zyklen * 0,667 µs * 250  =  1 ms
   djnz  R7,de
   djnz  R6,de

   ret

END
```

Listing 3 Modul delay.a51

7 Literaturverzeichnis

Autor	**Titel**	**Verlag**
Roth	Das Mikrocontroller-Kochbuch	IWT-Verlag ´95
Kiencke	Meßtechnik	Skriptum zur Vorlesung