Karlsruher Institut für Technologie

IIIT

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

Praktikum: „Mikrocontroller und digitale Signalprozessoren“

Versuch 1

Digitale Drehzahlmessung mit dem µC ADuC7026

Inhaltsverzeichnis

1.1 Einleitung

In diesem Praktikumsversuch soll die Drehzahl eines Elektromotors mit Hilfe des Mikrocontrollers *ADuC7026* 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.

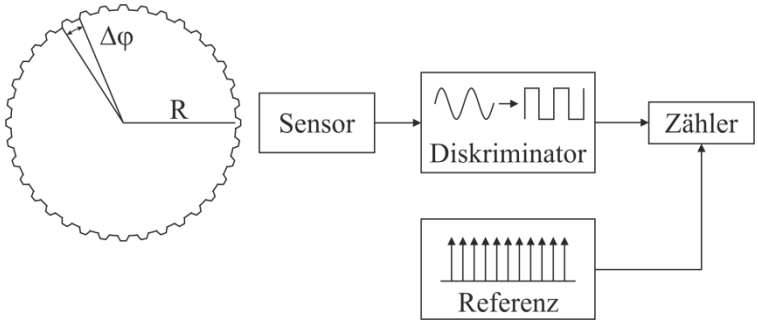


Abbildung 1:Messsystem zur frequenzanalogen Drehzahlmessung

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, wie in Abbildung 2 dargestellt. 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 zur 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.

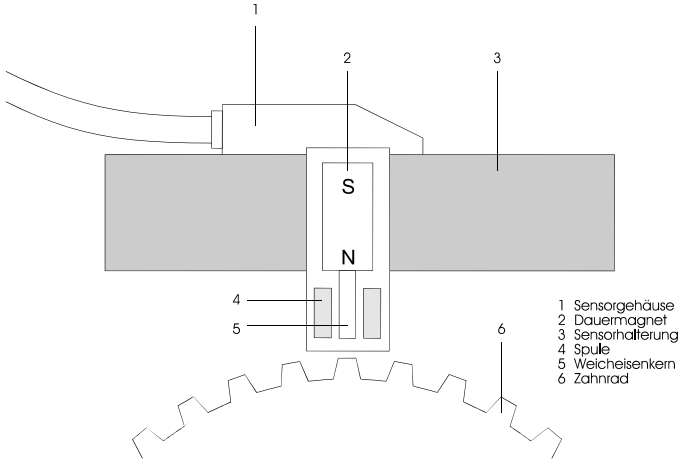


Abbildung 2: Aufbau des induktiven Zahnradsensors

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. Da die Signalform binär ist, eignet sich daher ein Mikrocontrollers sehr gut zur Auswertung.

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.

1.2 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

$$N = f \cdot t \ .$$

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

$$\omega = \frac{\varphi_2 - \varphi_1}{t_2 - t_1} \ .$$

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

1.2.1 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 $\varphi_m(i) = \varphi_{m2} - \varphi_{m1}$, 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.

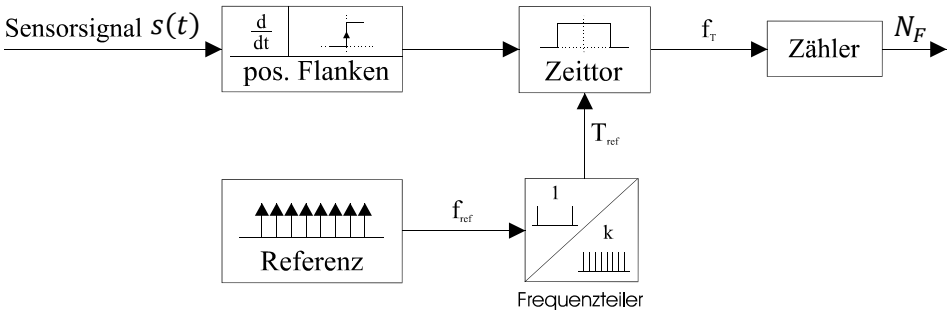


Abbildung 3: Strukturbild zur Frequenzzählung

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

$$f_x = \frac{N_F}{T_{ref}} = N_F \cdot \frac{f_{ref}}{k} \quad .$$

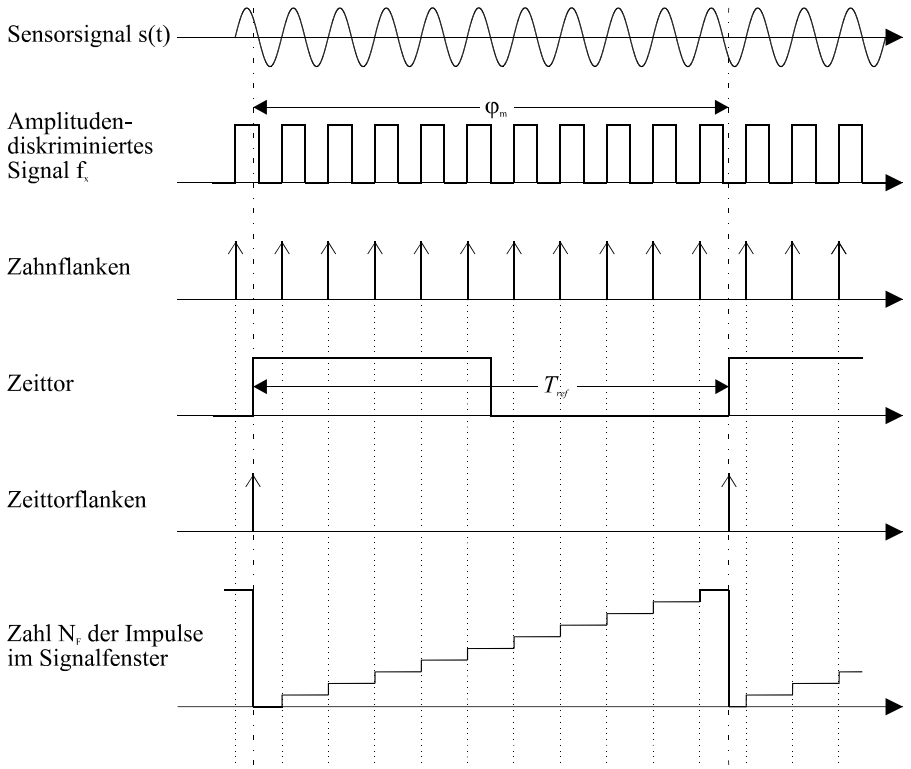


Abbildung 4: Digitalisierung frequenzanaloger Signale durch Frequenzzählung

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:

$$\left|\Delta f_x\right| = \frac{1}{T_{ref}} \quad .$$

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.

1.2.2 Digitale Periodendauermessung

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

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

benötigt wird. Dieser Winkel ist durch die Winkelteilung des Zahnrades fest vorgegeben.
Die Winkelgeschwindigkeit ergibt sich zu

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

und umgekehrt proportional zu der Periode T_m zwischen dem Durchlauf zweier aufeinanderfolgender Zähne am Sensor vorbei.

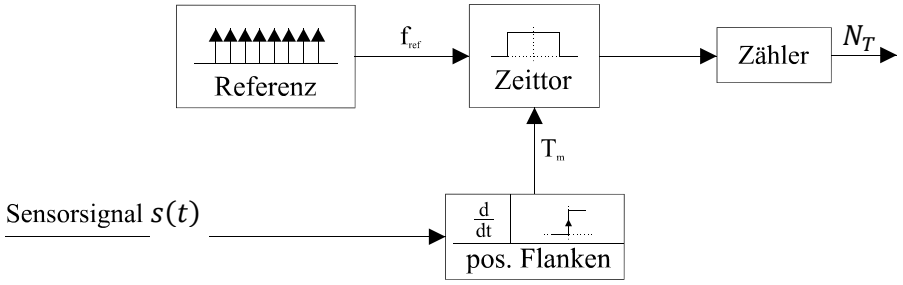


Abbildung 5: Strukturbild zur Periodendauermessung

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.

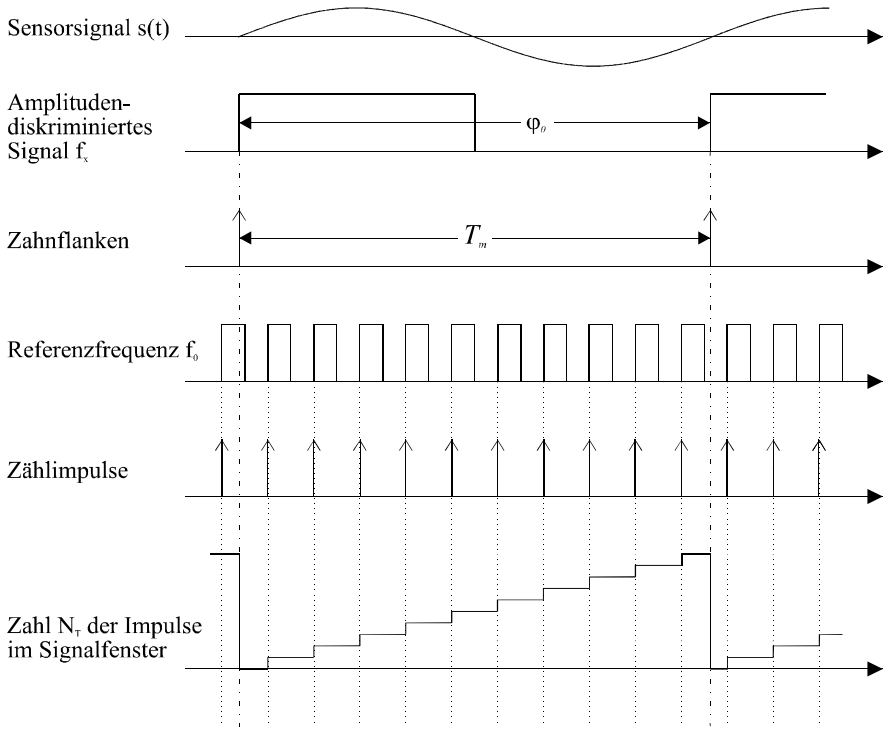


Abbildung 6: Digitalisierung frequenzanaloger Signale durch Periodendauermessung

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

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

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.

1.2.3 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.

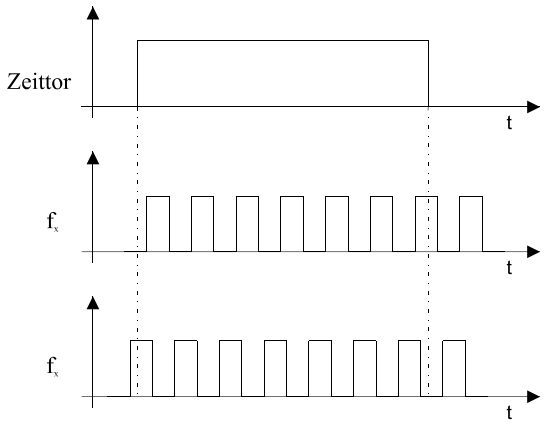


Abbildung 7: Quantisierungsfehler: je nach Lage von Zeit und Frequenzsignal zueinander können 6 oder 7 Vorderflanken gezählt werden

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:

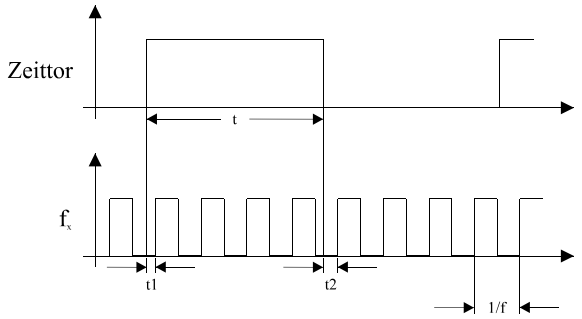


Abbildung 8: Definition beim Quantisierungsfehler

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

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

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

$$f \cdot t = N + f \cdot (t_1 - t_2) = N_{soll} \ .$$

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

$$F_{Qabs} = N - N_{soll} = f \cdot (t_2 - t_1) \ .$$

Als Maximalwerte ergeben sich

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

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

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.

1.2.4 Relativer Quantisierungsfehler

Die beiden Verfahren beruhen darauf, dass die Bestimmung der Drehzahl ω entweder über das Zählen der Impulse eines Referenztaktes $f_0 = 1/T$ (Periodendauermessung) 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 $N \cdot T_0$ 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

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

Bei maximaler Abweichung um ein Zählintervall T_0

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

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 Sensorflanken-Impulse. 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

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

Bei maximaler Abweichung um ein Winkelinkrement φ_0

$$\left|\varphi_m - N \cdot \varphi_0\right| \leq \varphi_0$$

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.

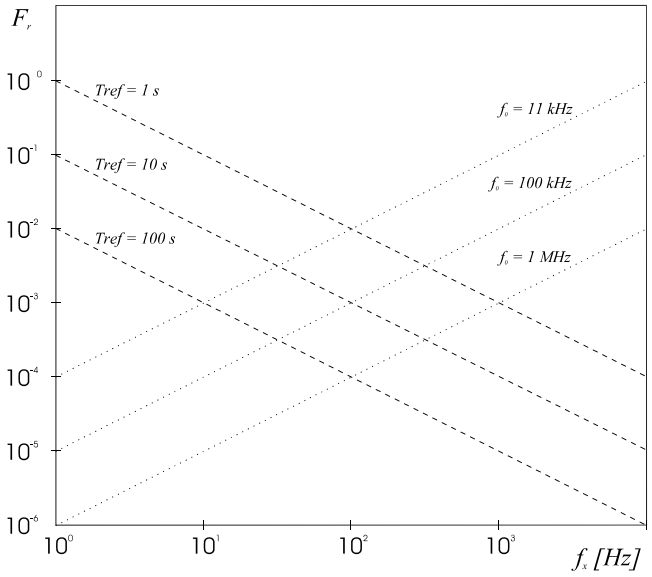


Abbildung 9: Relativer Quantisierungsfehler (--- Frequenzzählung, . . . Periodendauermessung)

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.

1.2.5 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:

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

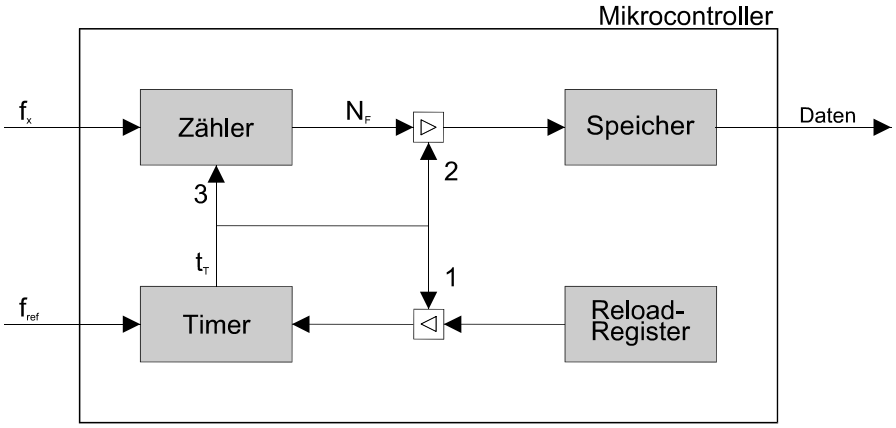
Die Multiperiodendauermessung hat ihre Stärken im mittleren Frequenzbereich. Dort ist der Fehler bei Umschalten zwischen Frequenzmessung und Einzelperiodendauermessung am größten. Bei niederen Frequenzen läuft der Impulszähler bei der Multiperiodendauer sehr schnell über, bei sehr hohen Frequenzen ist 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 Frequenz-Zählung 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.

1.3 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 auf dem Mikrocontroller implementiert werden. Der im Praktikum verfügbare Mikrocontroller ADuC7026 stellt hierzu insgesamt vier Timer zur Verfügung, die auch als Zähler eingestellt werden können. Welcher dieser Bausteine verwendet wird, hängt von der Art der zu lösenden Aufgabe ab, da jeder von ihnen spezielle Zusatzfunktionen besitzt.

1.3.1 Implementierung der digitalen Frequenzmessung

Für die digitale Frequenzmessung mit dem ADuC7026 wird ein Register als Impulszähler verwendet. Er hat die Aufgabe, die 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 (als Aufwärts-Zähler), d.h. das Umkippen des Zählerstandes von FFFFFFFFh nach 00000000h löst den Autoreload und zusätzlich einen Interrupt aus.[1] 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).

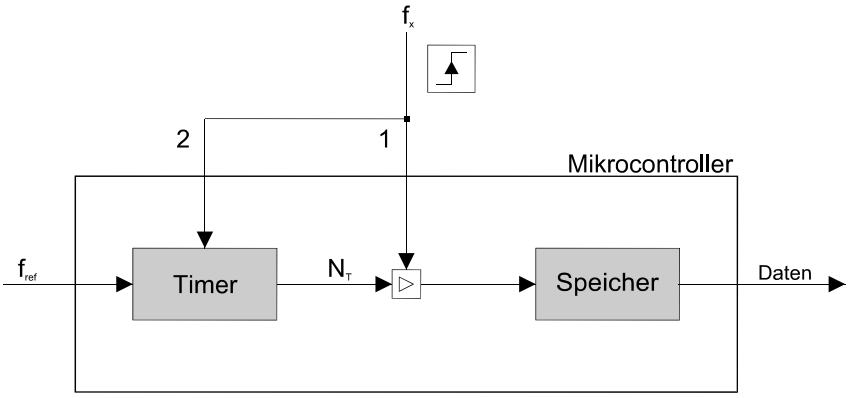
1.3.2 Implementierung der digitalen Periodendauermessung

Je nach Art und Weise, wie die Kopplung zwischen Zeitsignal und Zeitmesser (Timer) erfolgt, unterscheidet man drei verschiedene Arten, auf die man beim ADuC7026 eine Periodendauermessung durchführen kann.

1.3.2.1 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] Das nachstehende h kennzeichnet Hexadezimalzahlen



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

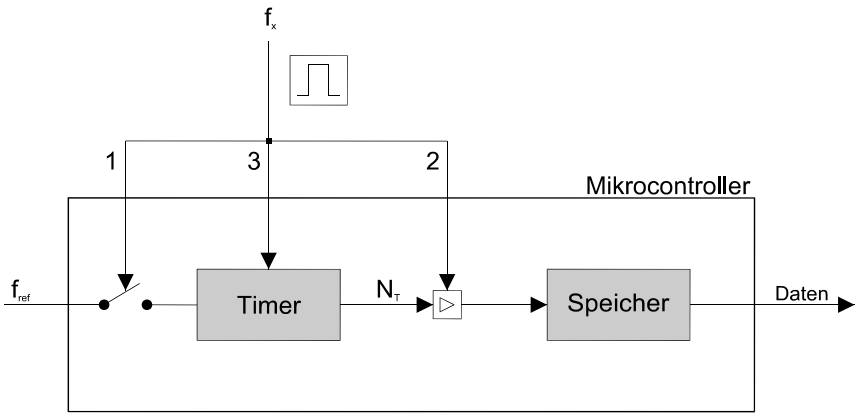
Abbildung 11: Periodendauermessung mit Interrupts

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 dauert mehrere 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.

1.3.2.2 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

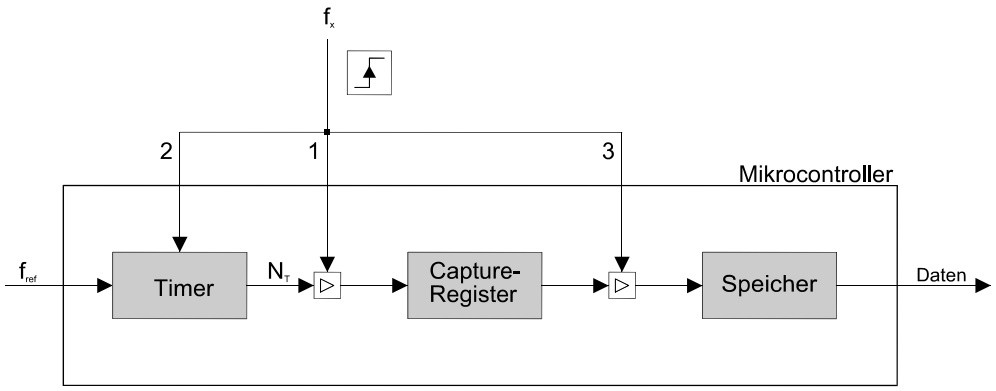
Durch die Hardwarekopplung von Messsignal und Zähler ist die Abbildung der Periodendauer auf den Zählerstand sehr genau. Ein Fehler in der Größenordnung wie bei der Interrupt-gesteuerten Periodendauermessung tritt hier nicht auf. Dieser Vorteil wird durch die etwas aufwendigere Hardware erkauft.

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.

1.3.2.3 Verwenden eines Capture-Registers

Im Praktikumsversuch soll die Periodendauermessung mit einem speziellen Feature des ADuC7026 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 TxVAL in dieses Register TxCAP übernommen wird. Dieser Vorgang erfolgt hardwaregesteuert und ist somit sehr schnell.



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 stattdessen den vorher gespeicherten Stand des Zählers aus der letzten Periode vom aktuellen Zählerstand abzieht, den Zähler also als sogenannten Ringzähler verwendet.

Der ADuC7026 enthält den Timer1, der als einziger mit der Capture-Funktion betrieben werden kann. Entsprechend muss dieser im T1CON Register konfiguriert werden.

1.4 Darstellung der Drehzahl auf dem Display

Zur Anzeige von Messergebnissen des Mikrocontrollers ist der Versuchsaufbau mit einem Anzeigenmodul ausgestattet. Es besteht aus einer LCD-Punkt-Matrix, einem Grafikcontroller, einem Anzeigenspeicher und einer integrierten Ansteuerlogik. Es ist in der Lage, zwei Zeilen mit jeweils 16 Zeichen darzustellen. Der Adressbereich erstreckt sich für die 16 Zeichen der ersten Zeile von 00h bis 0Fh, für die 16 Zeichen der zweiten Zeile von 40h bis 4Fh.

Der Grafikcontroller besitzt ein Charakter-ROM, in dem die Zeichencodes der darstellbaren Zeichen gespeichert sind. Ein Zeichen setzt sich aus einer 5x7-Punktmatrix zusammen. Im Anzeigenspeicher stehen die Codes der Zeichen, die auf dem Display dargestellt werden sollen (siehe Abbildung 17 im Anhang). Er besitzt eine Größe von 80 Bytes. Jedes Byte repräsentiert einen Anzeigenplatz auf dem Display, es bildet also einen Vektor in das Charakter-ROM, von wo sich der Grafikcontroller das entsprechende Muster für das Zeichen holt. Der Zeichensatz umfasst mit kleinen Ausnahmen den ASCII-Code (Zeichencodes 20h bis 7Dh).

Der Grafikcontroller besitzt einige Befehle, mit denen er die Anzeige steuern kann, z.B. um das Display ein- und auszuschalten oder den Anzeigenspeicher zu löschen (siehe Tabelle 3 im Anhang). Neben den Befehlsbytes empfängt er Datenbytes. Zur Unterscheidung werden die Leitungen *RS* (*Register Select*) und *R/W* (*Read/Write*) verwendet. Die Beschreibung des LCD Leitungen sind in Tabelle 1 zu sehen.

Tabelle 1: Pinbeschreibung des LCD-Moduls

Pin	Symbol	Logik	Richtung	Beschreibung
1	Vss	-	-	Masse
2	Vcc	-	-	Spannungsversorgung (+5V)
3	Vee	-	-	Kontrast anpassen
4	RS	0/1	IN	0: Befehl; 1: Daten
5	R/W	0/1	IN	0: Schreiben; 1: Lesen
6	E	0/1	IN	Enable-Signal
7	DB0	0/1	IN/OUT	Datenbus Bit 0 (LSB)
8	DB1	0/1	IN/OUT	Datenbus Bit 1
9	DB2	0/1	IN/OUT	Datenbus Bit 2
10	DB3	0/1	IN/OUT	Datenbus Bit 3
11	DB4	0/1	IN/OUT	Datenbus Bit 4
12	DB5	0/1	IN/OUT	Datenbus Bit 5
13	DB6	0/1	IN/OUT	Datenbus Bit 6
14	DB7	0/1	IN/OUT	Datenbus Bit 7 (MSB)

Für die Ansteuerung der Anzeigeneinheit sind *GPIOs* (*General Purpose Input/Output*) des Mikrocontrollers reserviert. Die Ankopplung des Displays an den GPIO zeigt Abbildung 14.

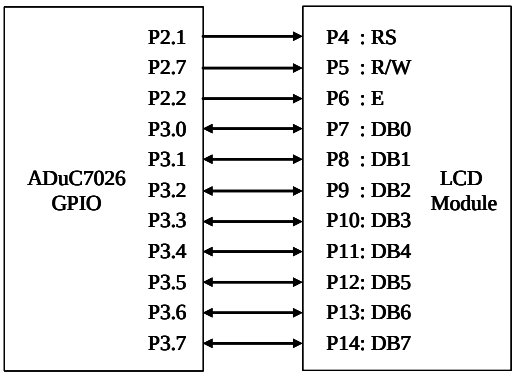


Abbildung 14: Anbindung des LCD-Displays an den ADuC7026 GPIO

Die Befehle und Daten werden über den Datenbus (DB0 bis DB7) transportiert. Die Auswahl zwischen Befehls- und Datenbytes erfolgt mit dem Signal RS. Ist es Logik 0, können Befehle an den Grafikcontroller gesendet oder der Adresszähler und das *Busy-Flag* gelesen werden. Ist es Logik 1, werden Datenbytes in den Anzeigenspeicher geschrieben oder aus ihm gelesen.

Die Kommunikation zwischen LCD Modul und µC ist bidirektional. Zur Unterscheidung der Richtung wird das Signal Read/Write (R/W) verwendet. Ist es Logik 0 so werden Befehle und Daten aus µC zum LCD gesendet, sonst ist die Richtung umgekehrt.

Erst bei fallender Flanke des Enable-Signals (*E*) werden die Befehle oder Daten von LCD Modul aufgenommen bzw. ausgegeben (Timingdiagramm in Abbildung 15). Nach einer Verarbeitungszeit von etwa 40µs kann der Grafikcontroller wieder das nächste Byte aufnehmen.

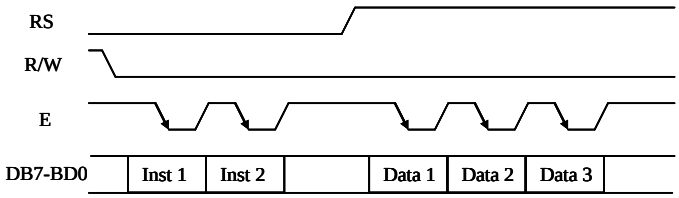


Abbildung 15: Zeitdiagram

Die Drehzahl liegt zunächst als Integer vor. Diese wird in eine vierstellige BCD[2]-Zahl umgewandelt. 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 geschrieben werden. Jeder Schreib-/Lesevorgang hat ein Autoinkrement des Adresszählers zur Folge, so dass alle nachfolgenden Daten in das Anzeigen-RAM fließen.

[2] Binary Coded Decimal.

1.5 Praktikumsversuch

Am Versuchstag steht Ihnen ein kompletter Aufbau zur Drehzahlmessung zur Verfügung. Des Weiteren 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.

1.5.1 Versuchsaufbau

Der Arbeitsplatz zur Durchführung der Drehzahlmessung besteht aus einem PC, dem Mikrocontrollerboard mit der Anzeigeeinheit sowie einem Gleichstrommotor mit ferromagnetischem Zahnrad (50 Zähne).

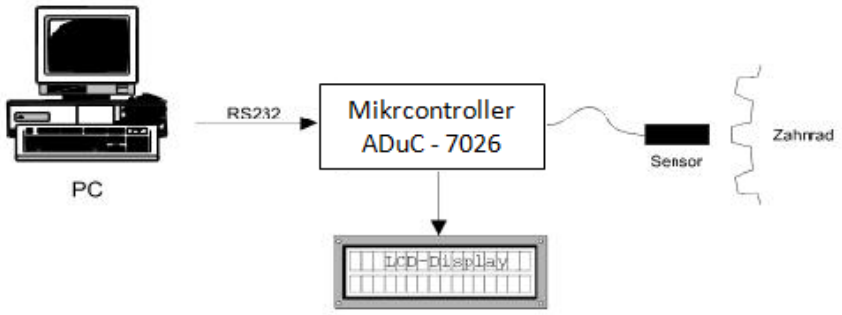


Abbildung 16: Versuchsaufbau zur Drehzahlmessung

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. Das Ausgangssignal des Komparators, welcher ein Interrupt (IRQ) bei jeder Flanke auslöst, soll als Impulszähler verwendet werden, womit sich eine Frequenzmessung realisieren lässt. Eine Periodendauermessung erfolgt ebenfalls über den Komparator der als Capture-Event von Timer1 dienen soll. Die Ausgabe des Messergebnisses erfolgt auf einem LCD, das über den GPIO des Mikrocontrollers angesprochen werden kann.

1.6 Aufgaben

Im Rahmen dieses Praktikumsversuchs sind die folgenden Aufgaben zu bearbeiten. 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 C erstellt werden.

Aus dem Datenblatt des ADuC7026 sollten Sie sich vor dem Versuch insbesondere die Baugruppen Comperator, GPIO, Interrupt System, und Timers (Timer1 sowie Timer2) anschauen.

Verinnerlichen Sie sich das detallierte Blockdiagramm des ADuC7026. Falls Sie es bei einer Abfragung zeichnen müssen, dann können die Pinne zusammengefasst werden.

1.6.1 Vorbereitungsaufgabe

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?

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

d) Wie beeinflusst die Anzahl der Zähne auf dem Zahnrad die Messgenauigkeit bei beiden Verfahren?

e) Beschreiben Sie ARM-Register, die Organisation des Speichers sowie die einzelnen Speicherblöcke Flash/EE, SRAM und MMR.

f) Mit welchem MMR-Register kann die Zählfrequenz des Timer1 skaliert werden? Schreiben Sie einen Befehl, der die Zählfrequenz $f_{\text{clk\_source}}/256$ realisiert. Welche maximale Torzeit kann mit dem Timer1 realisiert werden? (Hinweis: alle Stellen des Registers sind original auf 0 gesetzt)

g) Erläutern Sie die Signale RS, R/W und Enable des LCD-Displays.

R/W \ RS	0	1
0		
1		

Welches Register liefert die Befehle RS, R/W und Enable über den Datenbus (DB0 bis DB7) ans LCD Display? Nennen Sie die jeweiligen Bits mit dem die Befehle ausgeführt werden.

h) Schreiben Sie im C-Code eine Code-Sequenz mit der Sie *core clock frequency* mit 10,44 MHz konfigurieren. Warum wird das Register POWCON in einer bestimmten Sequenz geändert?

i) In dem 32-Bit Register GP2DAT sollen die Bits 8 bis 15 gesetzt werden. Zudem sollen die Bits 24 bis 31 dem Byte C3h entsprechen. Schreiben Sie dafür zwei alternative Befehle. Beschreiben Sie außerdem das Register GP2DAT und den ausgeführten Befehl.

1.6.2 LCD Display

Bei diesem Versuch soll das LCD Display zur Darstellung der Messergebnisse vorbereitet werden. Öffnen Sie das Projekt …\Aufgabe_LCD\Aufgabe_LCD.Uv2. Binden Sie folgende Dateien in dieses Projekt ein:

startup_RV.s
irq_arm.c
LCD_utility.h
delay_utility.h

In LCD_utility.h sollen Funktionen zur Displaysteuerung implementiert werden. In delay_utility.h steht die Funktion delay() zur Verfügung.

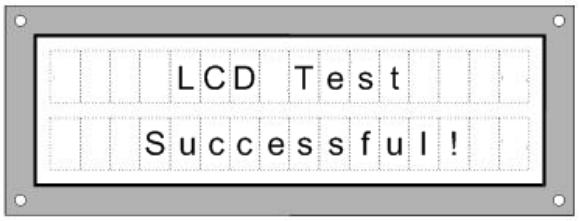
Falls noch nicht geschehen, stellen Sie den Compiler richtig ein, damit nach einem Reset der Mikrocontroller richtig startet: Project Workspace → rechte Maustaste auf Target 1 klicken → Options for Target 'Target 1' → Linker → tippen Sie in Feld Misc controls `--first startup_RV.o(Reset)` ein.

a) Öffnen Sie LCD_utility.h, programmieren Sie zwei Funktionen lcd_data() und lcd_command(), welche für die Ausgabe eines einzigen Daten- bzw. Befehlbytes zuständig sind.

Wie in Abbildung 15 beschrieben, muss das RS Signal für ein Befehlbyte auf 0 und für ein Datenbyte auf 1 gesetzt werden. Das R/W Signal muss auf 0 gehen, um das Schreiben eines Bytes zu ermöglichen. Das Enable Signal (E) soll zuerst auf 1 gesetzt werden, anschließend soll das Byte an den Datenbus (DB7-DB0) geschrieben werden.

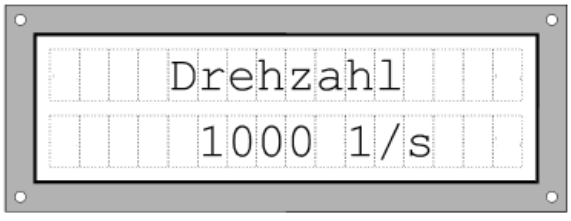
Um die Holdzeit zu erfüllen soll die delay() Funktion ausgeführt werden, bevor das Enable Bit (E) wieder auf 0 geht. Nach der fallenden Flanke ist das Byte erfolgreich vom LCD-Kontroller übernommen worden. Schließlich erfolgt noch mal die delay() Funktion und das Setzten des E Bits für die Freigabe eines neuen Befehls oder neuer Daten.

Binden Sie nun LCD_a.c ins Projekt ein, diese enthält Programmschnitte zur Konfiguration des GPIO, zur Initialisierung des LCD-Kontrollers, und zum Test des LCDs. Überprüfen Sie, ob die Datei LCD_utility.h eingebunden ist und die Funktionen lcd_command(), lcd_data() und delay() deklariert sind. Kompilieren Sie das Projekt und laden Sie die *Hex*-Datei in den µC. Wenn die beiden Funktionen richtig sind sollten Sie folgende Texte auf dem Display sehen:



b) Ein beliebiger Integer soll auf dem Display angezeigt werden. Nehmen Sie nun LCD_a.c aus dem Projekt, binden Sie stattdessen LCD_b.c ein.

Eine Integer value wurde definiert und hat den Wert 1000. Erstellen Sie einen Abschnitt im code in der endlosen *While*-Schleife, um diesen Wert in eine vierstellige BCD-Zahl umzuwandeln. Vier Variablen b_1000, b_100, b_10, b_1, entsprechen der Tausender, Hunderter, Zehner und Einer, sind schon definiert. Sie sollen über die Funktion lcd_data() auf dem LCD erscheinen. Stellen Sie das LCD nun so ein, dass das 4-stellige Ergebnis in der 2. Zeile ab 6. Spalte angezeigt wird. Das Ziel:



Tipp: Jeder Ziffer der 4 stelligen Dezimalzahl (0-9999), wird einzeln konvertiert, addiert mit 48 und über die Funktion lcd_data(…) am Display ausgegeben.

1.6.3 Frequenzmessung

Die digitale Frequenzmessung soll mit Hilfe des Mikrocontrollers ADuC7026 realisiert werden. Öffnen Sie das Projekt …\Aufgabe_Frequenzmessung\ Frequenzmessung.Uv2. Binden Sie die folgenden Dateien in dieses Projekt ein:

startup_RV.s,
irq_arm.c,
Frequenzmessung.c,
delay_utility.h und
My_IRQ_Function.h.

Kopieren Sie ihre LCD_utility.h in den gleichen Ordner. Fügen Sie anschließend Ihren vorher erstellten Code, der in der *while*-Schleife eine Integer-Zahl in BCD konvertiert, in Frequenzmessung.c ein. Dabei ersetzten Sie die Variable value gegen frequency.

Falls noch nicht geschehen, stellen Sie den Compiler richtig ein, damit nach einem Reset der Mikrocontroller richtig startet: Project Workspace → rechte Maustaste auf Target 1 klicken → Options for Target 'Target 1' → Linker → tippen Sie --first startup_RV.o(Reset) in Feld Misc controls ein.

a) In Frequenzmessung.c müssen Timer2 und Comparator konfiguriert werden. Die entsprechende Interrupts sind auch freizuschalten:

Timer2:

- Clock source: external crystal (32,768kHz)
- Count down
- Enable Timer2
- Operation mode: periodic
- Format: binary
- Prescale: 1
- Berechnen Sie den Wert, für die in My_IRQ_Function.h definierte globale Variable reload_g, und laden Sie damit das Register T2LD. T2LD soll so geladen werden, dass die Zeit bis zum Überlauf genau einer Sekunde entspricht
- Setzten Sie die in My_IRQ_Function.h definierte globale Variable puls_counter_g auf 0

Comparator:

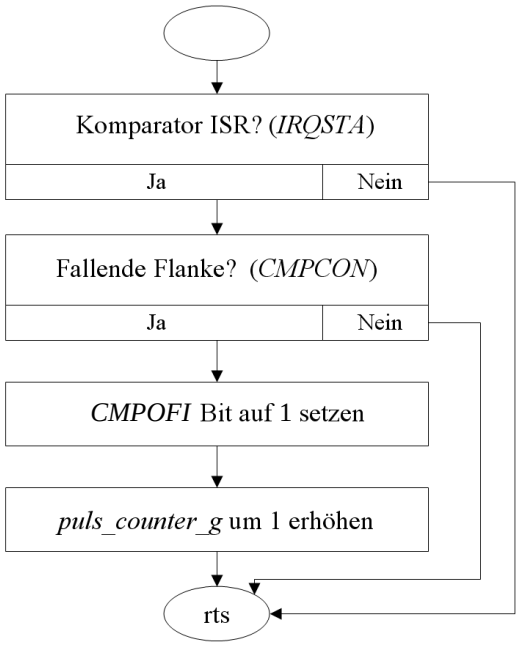
- Enable Comparator
- Negativer Eingang: $AV_{DD}/2$
- Komparatorausgang: IRQ
- Logik des Ausgangs: 1 wenn CMP0 > CMP1
- Reaktionszeit: 5µs
- Hysteresis einschalten
- Komparator-Interrupt wird bei fallender und steigender Flanke ausgelöst

Interrupts:

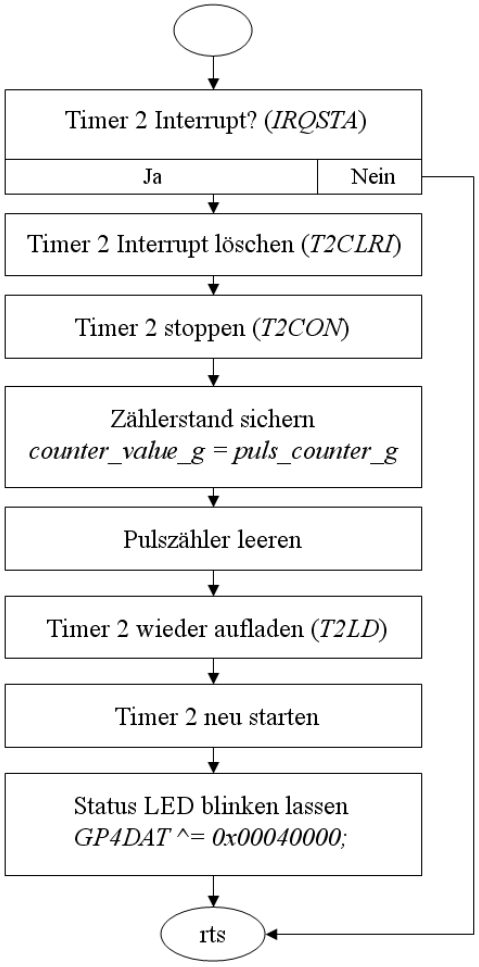
Comparator und Timer2 als Quelle des Interrupt zulassen (Register IRQEN)

b) Öffnen Sie nun My_IRQ_Function.h. Programmieren Sie für jeden Interrupt eine Interrupt-Service-Routine nach Flussdiagrame:

Komparator-ISR:



Timer2-ISR:



1.6.4 Periodendauermessung

In dieser Aufgabe soll die Periodendauermessung implementiert werden. Öffnen Sie das Projekt …\Aufgabe_Periodendauermessung\Periodendauermessung.Uv2. Binden Sie die folgenden Dateien ins Projekt ein:

startup_RV.s,
irq_arm.c,
Periodendauermessung.c,
delay_utility.h und
My_IRQ_Function.h.

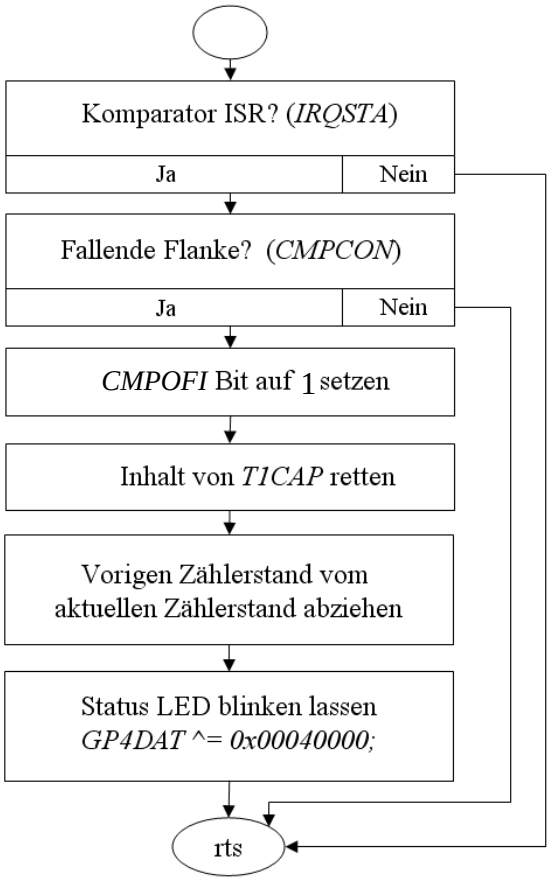
Ihre LCD_utility.h soll auch in den gleichen Ordner kopiert und ins Projekt eingebunden werden. Öffnen Sie anschließend Periodendauermessung.c. Fügen Sie anschließend Ihren vorher erstellten Code, der in der *while*-Schleife eine Integer-Zahl in BCD konvertiert, in Periodendauermessung.c ein. Dabei ersetzten Sie die Variable value gegen frequency.

Falls noch nicht geschehen, stellen Sie den Compiler richtig ein, damit nach einem Reset der Mikrocontroller richtig startet: Project Workspace → rechte Maustaste auf Target 1 klicken → Options for Target ‘Target 1’ → Linker → tippen Sie --first startup_RV.o(Reset) in Feld Misc controls ein.

a) In Periodendauermessung.c soll der Komparator wie in Aufgabe Frequenzmessung konfiguriert werden, der entsprechende Interrupt ist auch freizuschalten. Für die Periodendauermessung soll Timer1 wie folgend konfiguriert werden:

- Event select bit: enable time capture of an event
- Event: comparator interrupt
- Clock select: core clock
- Count up
- Enable Timer 1
- Operation mode: free-running mode
- Format: binary
- Prescale: 16

b) Öffnen Sie nun My_IRQ_Function.h. Programmieren Sie für Komparator Interrupt eine Interrupt-Service-Routine nach folgendem Flussdiagram:



c) Die Drehzahl soll nun in der Einheit Umdrehungen/min auf dem Display dargestellt werden. Führen Sie die notwendigen Änderungen in der while-Schleife in Periodendauermessung.c durch.

1.7 Anhang:

Tabelle 2: Bit Configuration

Bit Name	**Setting / Status**	
I/D	0 = Decrement cursor position	1 = Increment cursor position
S	0 = No display shift	1 = Display shift
D	0 = Display off	1 = Display on
C	0 = Cursor off	1 = Cursor on
B	0 = Cursor blink off	1 = Cursor blink on
S/C	0 = Move cursor	1 = Shift display
R/L	0 = Shift left	1 = Shift right
DL	0 = 4-bit interface	1 = 8-bit interface
N	0 = 1/8 or 1/11 Duty (1 line)	1 = 1/16 Duty (2 lines)
F	0 = 5x7 dots	1 = 5x10 dots
BF	0 = Can accept instruction	1 = Internal operation in progress

Tabelle 3: LCD Befehlsatz

Instruction	Code										description	Execution time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0)	1.64ms
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position (address 0). Also returns display being shifted to the original position. DDRAM contents remains unchanged	1.64ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D), specifies to shift the display (S). These operations are performed during data read/write.	40µs
Display On/Off control	0	0	0	0	0	0	1	D	C	B	Sets On/Off of all display (D), cursor On/Off (C) and blink of cursor position character (B).	40µs
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM contents remains unchanged.	40µs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N) and character font(F).	40µs
Set CGRAM address	0	0	0	1	CGRAM address						Sets the CGRAM address. CGRAM data is sent and received after this setting.	40µs
Set DDRAM address	0	0	1	DDRAM address							Sets the DDRAM address. DDRAM data is sent and received after this setting.	40µs
Read busy-flag and address counter	0	1	BF	CGRAM / DDRAM address							Reads Busy-flag (BF) indicating internal operation is being performed and reads CGRAM or DDRAM address counter contents (depending on previous instruction).	0µs
Write to CGRAM or DDRAM	1	0	write data								Writes data to CGRAM or DDRAM.	40µs
Read from CGRAM or DDRAM	1	1	read data								Reads data from CGRAM or DDRAM.	40µs

Remarks: DDRAM = Display Data RAM; CGRAM = Character Generator RAM; DDRAM address = cursor position

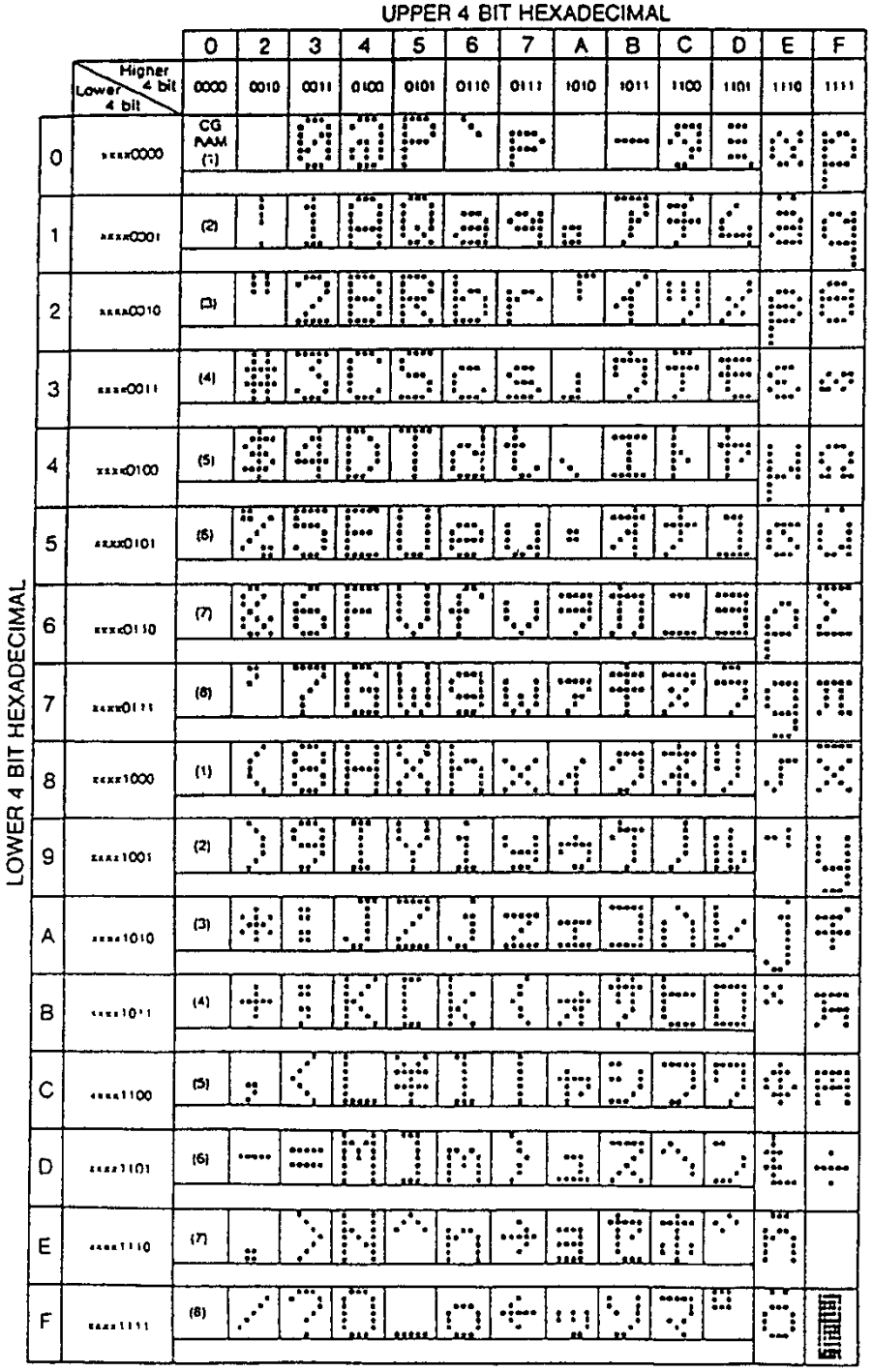


Abbildung 17: Zeichensatz des LCD