HAW Hamburg
Prof. Dr. Franz Korf
Prof. Dr. Martin Becke

Themen: Digital I/O, Bitmanipulation, Polling und einfache Zeitmessungen

Letzter Abgabetermin: Im Laufe des vierten Praktikumstermins

Aufgabe:[1] Bei einem rotierenden Gegenstand wird mit Hilfe eines inkrementellen Drehgebers der Drehwinkel ermittelt. Oft besteht ein Drehgeber aus einer Scheibe mit Schlitzen (s. Abbildung 1). Wenn sich die Scheibe dreht, wird abwechselnd der Weg für einen Lichtstrahl von einer LED zu einer Photodiode versperrt und wieder frei. Zur Ermittlung der Drehrichtung wird eine zweite Photodiode ein wenig versetzt angebracht. Diese Anordnung ist in Abbildung 1 dargestellt.

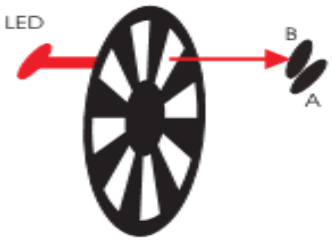

Abbildung 1: Drehgebers mit versetzten Photodioden

Solch eine Anordnung ist im TI-Labor als Lehrmodell mit einer Drehscheibe aufgebaut (s. Abbildung 2). In Teams liegt das Datenblatt des Drehgebers vor. Für die Entwicklung Zuhause werden die Signale des Drehgebers durch ein Programm auf dem Raspberry Pi Pico nachgebildet/simuliert. Details dazu finden Sie in Teams. **Für die Abgabe behalten wir uns vor das Setup von Pico auf Drehgeber (s. Abbildung 2) zu wechseln.**

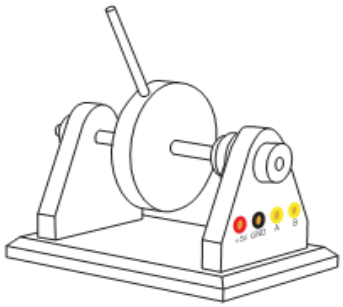

Abbildung 2: Drehgeber im TI Labor

Wie in Abbildung 3 gezeigt, lassen sich die Impulsverläufe an Kanal A und B (Drehgeber) in vier Phasen a, b, c und d einteilen: Bei einer Vorwärtsbewegung werden die Phasen in der Reihenfolge a, b, c und d durchlaufen, danach folgt wieder die Phase a. Bei einer Rückwärtsbewegung werden die Phasen in umgekehrter Reihenfolge durchlaufen. Durch Beobachtung der Phasen lässt sich nun sehr einfach die Bewegungsrichtung der Scheibe und die Anzahl der durchlaufenen Phasen feststellen. Ist die aktuelle Phase b so ergeben sich bei der nächsten Beobachtung der Signale die Fälle aus Tabelle 1. Bevor sie mit der Entwicklung beginnen, versuchen Sie die Systematik zu verstehen und die Grenzen das Verfahrens zu identifizieren.

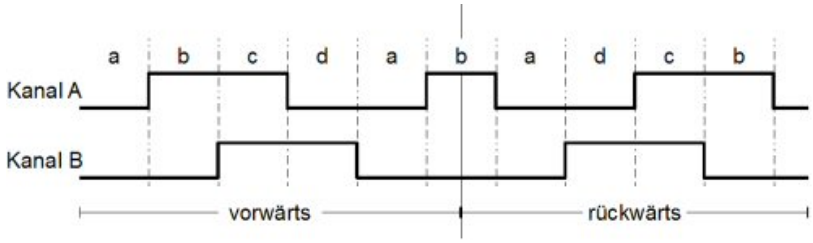

Abbildung 3: Ausgangssignale des Drehgebers in Abhängigkeit des Drehwinkels

[1] Diese Aufgabe basiert auf Unterlagen zum GSP Praktikum von Prof. Dr. Heiner Heitmann, HAW Hamburg.

Zuordnung Signale - Phase		
Kanal A	Kanal B	Phase
0	0	a
1	0	b
1	1	c
0	1	d

Zuordnung Phasenwechsel - Ereignis	
Phasenwechsel	Ergebnis
Phase b →Phase a	Rückwärtslauf
Phase b →Phase b	Keine Änderung
Phase b →Phase c	Vorwärtslauf
Phase b →Phase d	Fehler

Tabelle 1: Phasen und Phasenübergänge

Funktionsumfang des Programms

- Unter Beachtung der Drehrichtung wird die Anzahl der Schritte / Phasenwechsel - nicht der Winkel in Grad - binär codiert an den LEDs D8 bis D15 ausgegeben. Gemäß dem Schaltplan (s. Teams) sind diese LEDs an den GPIOs PD0 bis PD7 angeschlossen. Die acht LEDs reichen nicht zur vollständigen Darstellung des Drehbereichs aus. Deswegen werden nur die niederwertigsten 8 Bits dieses Vorwärts-Rückwärtszählers angezeigt.
- Auf dem LCD-Display wird der Drehwinkel in Grad und die Drehgeschwindigkeit in Grad pro Sekunde ausgegeben. Der Winkel muss exakt angegeben werden, die Drehgeschwindigkeit darf im Nachkommabereich abweichen. Zur einfachen Bestimmung des Drehgeschwindigkeit benötigen Sie einen Timer zur Zeitmessung (s. `timer` Module aus der `ITS_BRD_LIB` Bibliothek).
- LED D23 (angeschlossen an PE7) leuchtet, wenn als letztes eine Drehung in Vorwärtsrichtung erkannt wurde.
- LED D22 (angeschlossen an PE6) leuchtet, wenn als letztes eine Drehung in Rückwärtsrichtung erkannt wurde.
- Wurde ein Fehlerfall erkannt, leuchtet LED D21 (angeschlossen an PE5). Die LED wird durch Löschen des Fehlers zurückgesetzt (s.u.).
- Mit der Taste S7 (angeschlossen an PF7) kann der Zählerstand jederzeit auf 0 zurückgesetzt werden.
- Mit der Taste S6 (angeschlossen an PF6) wird der Fehlerzustand gelöscht.
- Die Umrechnungen des binären Zählerstandes in die Anzeigewerte soll ausschließlich mit Integer-Arithmetik realisiert werden. Achten Sie darauf, dass eine maximale Genauigkeit erzielt wird.
- Der Drehgeber bzw. der Pi Pico (zur Simulation des Drehgebers) werden über die Anschlüsse IN0 (angeschlossen an GPIO PF0) und IN1 (angeschlossen an GPIO PF1) mit den ITS-Board verbunden.

  Bei der Nutzung des Drehgebermodells im Labor beachten Sie, dass der Drehgeber mit **3,3V** versorgt werden muss. Das ITS Board liefert den benötigten Strom (s. Bild der Verkabelung in MS Teams). Ein Labornetzteil wird nicht verwendet.

  Mit Blick auf die Qualitätssicherung Ihres Programms ist der Drehgeber ungenau. Bitte nutzen Sie den auf dem Pi Pico implementierten Rechteckgenerator zum Test Ihres Programms (s. MS Teams).

Analyse des Zeitverhaltens

Wenn Sie das Programm entwickelt und getestet haben, werden Sie feststellen, dass beim schnellen Drehen des Drehgebers die Software auf dem ITS Board dahingehend Probleme bereitet, dass nicht alle Phasenwechsel erkannt werden. In diesem Fall ist die Ausführung der Software zwischen dem Auslesen zweier aufeinanderfolgender Phasen langsamer als der Phasenwechsel. In Aufgabe 5 werden wir dieses Problem mit Hilfe von Interrupts signifikant lindern.

In dieser Aufgabe sollen Sie ausmessen **und dokumentieren**, welche Teile der SW die meiste Rechenzeit benötigen. Im Labor können Sie die Zeitmessungen mit dem Oszilloskop auf dem Labortisch durchführen (ein einfaches Video zur Benutzung des Oszilloskops finden Sie unter MS Teams). Zur Messung schließen Sie einen freien GPIO Pin des ITS Boards an einen Kanal des Oszilloskops an (Anschluss der Masse nicht vergessen). Möchten Sie zum Beispiel die benötigte Rechenzeit eines Codestücks bestimmen, setzen Sie an seinem Anfang den GPIO Pin auf high. Am Ende des Codestücks wird der GPIO Pin wieder auf low geschaltet. Die mit dem Oszilloskop gemessene Impulsbreite entspricht (bis auf eine shr kleine Abweichung für die GPIO Aufrufe) der Ausführungszeit des Programmstücks.

Unterlagen, die vor dem Praktikum auf git.haw-hamburg.de hochgeladen werden müssen.

- Erstellen Sie ein schriftliches Konzept als PDF und speichern sie es in dem Ordner der Readme. Dieses Konzept muss folgende Themen beinhalten:
  - Sinnvolles Modulkonzept (Aufgabe2-Modulkonzept.pdf)
  - Auflistung der möglichen Eingabezustände und die Reaktion auf diese (Aufgabe2-Zustände.pdf)
  - Konzept zur einheitlichen Behandlung von Fehlerzuständen (Aufgabe2-Fehler.pdf)

Abnahme der Aufgabe im Praktikum

- Das Konzept muss vorliegen.
- Kommentierter Quellcode, der dem C-Coding Style (s. MS Teams) entspricht, muss vorliegen.
- Ihre Tests des Programms müssen zusammen mit den Testergebnissen vorliegen. Eine Liste fasst die Testergebnisse für unterschiedliche Winkelgeschwindigkeiten und Winkel gut zusammen. Überlegen Sie sich sinnvolle Testfälle.
- Die oben geforderte Analyse des Zeitverhaltens muss vorliegen.
- Sie müssen Ihr Programm erklären können.

Hinweise

- Auch hier sind die in Aufgabe 1 gegebenen Hinweise zur Vorgehensweise bei der Lösung einer Programmieraufgabe hilfreich.
- Ein gutes Modulkonzept legt die Schnittstellen fest. Teilen Sie auf Basis der Module

den Programmieraufwand im Team auf und diskutieren Sie anschließend den Code.

- Für die Winkelberechnung benötigen Sie die Anzahl der Schlitze der Scheibe des Drehgebers. Der Drehgeber hat 300 Schlitze und daraus ergeben sich 1200 Zustände (Je Schlitz 4 Zustände) für eine 360° Drehung (!).
- Verwenden Sie das DDC Konzept aus der Vorlesung. Diese einfache und klare Struktur erleichtert die Programmentwicklung.
- Der Zugriff auf I/O erfolgt über Bit Manipulationen. Gehen Sie hier nach einem einheitlichen Verfahren vor (s. Vorlesung).
- Die Berechnung des binären Zählerstands soll mit Integer-Arithmetik möglichst genau erfolgen. Wird der Zähler in der Einheit 0.1° gespeichert, dann kann eine exakte Berechnung implementiert werden.
- Sie müssen den Zustand des Drehgebers kodieren. Ein einfacher und sicherer Weg ist der Einsatz einer Finite State Machine (FSM). Das ist in Prinzip ein endlicher Automat ohne Endzustände. Typische Zustände dieser FSM sind StartState, ErrorState, PhaseX {Links, Rechts, ND). Sie können den Zustand des Drehgebers auch anders kodieren.
- Eine einfache Berechnung der Drehgeschwindigkeit zählt in einem vorgegebenen Zeitfenster die Anzahl der Phasenübergänge und berechnet aus diesen Daten die Winkelgeschwindigkeit. Für eine hohe Genauigkeit ist es sinnvoll, dass man ggf. das Zeitfenster bis zum nächsten Phasenwechsel verlängert.
- Verwenden Sie zur Berechnung des Zeitfensters die Funktionalität des `timer` Moduls aus der `ITS_BRD_LIB`. In diesem Modul ist ein HW Timer so eingestellt, dass er ein 32 Bit Register 90 mal in einer Mikrosekunde um 1 inkrementiert. Hat das Register den Wert `0xFFFFFFFF` erreicht, dann findet ein Überlauf statt und der nächste Registerwert ist `0x00000000`.

  Im Prinzip entspricht die Differenz zweier Werte, die aus dem Timer Register gelesen wurden, der Zeitspanne, die zwischen den beiden lesenden Zugriffen auf das Timer Register vergangen ist. Natürlich muss dabei der Überlauf beachtet werden.