PRP2 - Praktikumsaufgaben 1: Zugriffsmodul auf Festo-Anlage

A r b e i t s g r u p p e	
PRP-Gruppe (1,2,3 oder 4) / Team	PRP2- /
Name, Vorname	
Name, Vorname	

Lernziele:

- Ansteuerung von I/O-Ports durchführen
- Funktionsbibliothek zur Ansteuerung eines Laufbandes erstellen
- Inbetriebnahme

Allgemeine Hinweise

Vorbereitung:

- Bitte bereiten Sie die nachfolgenden Aufgaben soweit wie möglich Zuhause vor.
- Um den größten Lerneffekt zu erzielen, wird dringend empfohlen, dass zunächst jeder von Ihnen alleine versucht, die Aufgaben zu lösen. (Tauschen Sie sich bei Problemen mit Ihrer Gruppe oder mit anderen Kommilitonen aus.)
- Während des Praktikums sind die Lösungen von Ihrem Team gegebenenfalls zu vervollständigen und zur Abnahme zu präsentieren.

Erfolgreiche Teilnahme / Abnahme:

- Erreichen einer Mindestpunktzahl bei allen Eingangsfragen.
- Die erfolgreiche Abnahme aller Praktikumstermine
- Die Abnahme erfolgt individuell und nicht pauschal für jedes Team.
- Es besteht Anwesenheitspflicht bei allen Praktikumsterminen.
- Der Code muss den in der Vorlesung aufgestellten und zur Verfügung gestellten Programmierrichtlinien entsprechen.
- Mündliche Abnahme:
    - Vorführung der vollständigen Lösungen
    - Befragung zu den Lösungen sowie verwandten theoretischen Themen

Beachten: Verwenden Sie ausschließlich die in der Aufgabenstellung angegebenen Bezeichner und Namen (z. B. für Funktionen, Variablen).

1 Lernziele

- Bit-Manipulationen durchführen können
- I/O-Ports ansprechen können
- Funktionsbibliothek erstellen können
- Layering
- Inbetriebnahme durchführen / Debugging mit technischem System

2 Das Transfersystem

Ein Transfersystem der Firma Festo besteht aus einem Laufband mit diversen Sensoren und Aktoren, die per C-Programm ausgewertet und angesteuert werden sollen. Die Sensoren und Aktoren des Transfersystems werden über eine USB Schnittstelle angesprochen.

3 Ansprechen der I/O-Karte

Programmiert wird eine externe digitale I/O-Karte, die 3 Ports bedienen kann.

Um ein Port zu initialisieren, muss zum Programmanfang die Funktion

```
cbDConfigPort( <boardnummer>, <port>, <übertragungsrichtung>);
```

aufgerufen werden.

Die Boardnummer ist in unserem Fall immer die 0.

Um die drei Ports ansprechen zu können gibt es die vordefinierten symbolischen Konstanten :

```
FIRSTPORTA
```

```
FIRSTPORTB
```

```
FIRSTPORTCH
```

```
FIRSTPORTCL
```

Mit diesen Konstanten werden Port A und Port B in ihrer Gänze (alle 8 Bits) angesprochen. Im Port C gibt es 4 Signale, die als Eingang angesteuert werden müssen und 4 Signale, die als Ausgang angesteuert werden.

Die oberen 4 Bits werden mit FIRSTPORTCH (H=>hight) und die unteren 4 Bits mit FIRST-PORTCL (L=>low) angesprochen. Die oberen 4 Bits (High-Nibble) erscheinen im Call-by-Reference Parameter als Low-Nibble; sie werden automatisch an die richtige Position geschoben.

Die Übertragungsrichtung eines Ports wird mit den Konstanten `DIGITALOUT` und `DIGITALIN` festgelegt.

Um Daten an ein Port zu schicken, muss die Funktion

```
cbDOut( <boardnummer>, <port>, <wert>);
```

aufgerufen werden, wobei der Wert immer den gesamten Port beschreibt (8 Bit). D.h. der Wert muss immer ein unsigned – Wert mit mindestens 8 Bit sein und es werden immer die untersten 8 bzw. im Port C die untersten 4 Bit eingetragen. Den genauen Datentypen entnehmen Sie der Signatur der Funktion.

Um Daten von einem Port einzulesen, muss die Funktion

```
cbDIn(<boardnummer>, <port>, &<variable>);
```

aufgerufen werden. Die Variable muss vom Datentyp unsigned mit mindestens 8 Bit sein. Es werden immer vom angegebenen Port alle 8 Bit in die Variable eingetragen bzw. vom Port C die 4 Bits.

4 Aufgaben

Erstellen Sie eine Funktionsbibliothek, mit der die Festo-Anlage angesteuert werden kann. Es soll nur die Ansteuerung bzw. das Auswerten der Sensoren ermöglicht werden. Eine Prozesssteuerung soll noch nicht implementiert werden.

4.1 „Durchstich-Implementierung“

Machen Sie sich mit der Programmierung der I/O-Karte und der Anlage vertraut. Erstellen Sie dazu ein minimalistisches Programm, das einen beliebigen Sensor (Schalter oder Lichtschranke) einließt und den Wert auf der Konsole ausgibt. Nehmen Sie das Programm ggf. nur im Debugger in Betrieb. Das Programm dient nur dem Kennenlernen des Systems und braucht nicht vorgeführt werden.

4.2 Modul

Schreiben Sie einen Modul, der die Funktionalität zum Erfassen der aktuellen Sensorsignale und zum Einstellen der aktuellen Aktorsignale bereitstellt.

Innerhalb des Moduls und somit vor den Nutzern des Moduls versteckt befinden sich zwei lokale Bit-Felder, das sogenannten Prozessabbild für die Sensoren und Aktoren. Die Bit-Felder in den Prozessabbildern speichern alle Sensorsignale, wie sie von den Ports ausgelesen wurden bzw. die die Aktorsignale, die zu den Ports geschrieben werden sollen. (siehe Portbeschreibung) Die Bit-Felder werden nicht exportiert.

Das Modul beinhaltet somit nachfolgende Funktionen und exportiert die Schnittstellen zur Nutzung für andere Module.

- Eine Initialisierungs-Prozedur, die die Ports entsprechend ihrer Funktion konfigurieren.
- Eine Prozedur *updateProcessImage*, die die aktuellen Sensorsignale aus den Eingabe-Ports einliest und im internen Prozessabbild ablegt.
- Eine Prozedur *applyOutputToProcess*, die den Inhalt des intern erstellte Prozessabbild für die Aktoren an die Ausgabe-Ports schreibt und damit die Aktoren der Anlage aktiviert/deaktiviert.

Erstellen Sie weitere Funktionen, die auf dem Prozessabbild arbeiten:

- Die Funktion `isBitSet` bekommt eine Bit-Maske übergeben. Sie prüft, ob die durch die Bit-Maske angegebenen Bit im Prozessabbild gesetzt sind. Rückgabewert ist true (alle angegebenen Bit gesetzt) oder false.
- Die Funktion `setBitInOutput` setzt im Prozessabbild für Aktoren die als Bit-Maske übergebenen Bit.
- Die Funktion `clearBitInOutput` löscht im Prozessabbild für Aktoren die als Bit-Maske übergebenen Bit.
- Die Funktion `resetOutputs` setzt im Prozessabbild alle Aktoren auf einen sicheren Wert und überträgt das Prozessabbild an die Ausgabe-Ports.

Die typische Verwendung aus Sicht des nutzenden Moduls (z. B. main()) ist das Einlesen der Sensoren in das Prozessabbild, die Auswertung des Prozessabbildes und vormerken der Reaktion im Prozessabbild der Aktoren und abschließende Ausgabe an die Anlage.

Erstellen Sie zu dieser Modul-Beschreibung die entsprechende Source- und Header-Datei. Das Modul stellt als eigenständige Header-Datei die Schnittstellen und die Makros mit den Bit-Mustern für das Maskieren der einzelnen Sensoren und Aktoren im Prozessabbild bereit. (Hinweise: Wenn Sie das Prozessabbild geschickt entwerfen, dann sind die Bit-Masken ähnlich zu denen in der Port-Beschreibung).

4.3 Inbetriebnahme/Test

Schreiben Sie jetzt ein Programm, das die main()-Routine enthält und die Header-Datei von dem obigen Modul importiert.

Testen Sie mit diesem Programm das Auslesen der Sensoren, bzw. das Ansteuern der Aktoren, indem Sie die folgenden Fälle durchspielen.

a) Das Laufband soll nach links laufen

b) Das Laufband soll langsam nach rechts laufen und die Ampel soll auf rot stehen

c) Das Laufband soll langsam nach rechts laufen, die Weiche soll auf sein und die Ampel soll auf gelb stehen.

d) Setzen Sie das Laufband mit einem Reset wieder auf den Ursprungszustand.

e) Testen Sie, ob ein Werkstück im Einlauf liegt.

f) Testen Sie, ob ein Werkstück im Bereich der Höhenmessung liegt.

g) Testen Sie, ob ein Werkstück im Bereich der Weiche liegt und ob es sich um ein Metallteil handelt.

h) Testen Sie, ob die Rutsche voll ist.

i) Testen Sie, ob die Stop-Taste gedrückt wurde

j) Testen Sie, ob die Reset-Taste gedrückt wurde.

k) Testen Sie, ob der Not-Aus-Taster gedrückt wurde

l) Lassen Sie die Q2 und die Start-Taste leuchten

Zur Vereinfachung, schreiben Sie eine Funktion, die alle aktuellen Sensorwerte auf der Konsole auflistet. Diese kann dann bei den Punkten e-k aufgerufen werden.

Machen Sie auch jeweils eine Gegenprobe.

5 Optional: Höhere Zugriffsfunktionen

Erstellen Sie ein weiteres Modul, das Funktionen bereitstellt, mit denen sich die Sensoren und Aktoren „einfacher“ ansprechen lassen. Diese Funktionen verbergen die Abfragen, bzw. die Manipulationen mittels Bit-Masken.

Beispiel:

Die Funktion `driveRightSlow` setzt für den Motor die entsprechenden Bit-Kombinationen für den langsamen Lauf des Laufbandes nach rechts.

Die Funktion `isItemAtJunction` liefert true, wenn der Sensor ein Bauteil bei der Weiche detektiert.

6 Abgabe

Das lauffähige Modul ist spätestens zum Beginn des nächsten Praktikumtermins vorzuführen. Die abgenommenen Source- und Header-Dateien sollen als ZIP-Paket abgegeben werden.