Protokoll: Z8-ADU-Interface - IT 8

Gruppe 5: Byron Worms, Raik Dankworth, Martin Knoll

29. Mai 2012

1 Was ist der Z8?

Bei dem Z8 handelt es sich um eine Reihe von älteren Mikrokontrollern, die sich dadurch auszeichnen, dass sie einen recht simplen Aufbau besitzen aber dennoch einen recht großen Funktionsumfang aufweisen.

Was ist ein Mikrokontroller? Bei einem Mikrokotroller handelt es sich um ein Ein-Chip-Computersystem, was neben der logischen und arithmetischen Einheit zusätzliche Bausteine/-Baueinheiten aufweisen kann, wie zum Beispiel: Ports, Timer, Programmspeicher, Datenspeicher und Komparatoren.

2 Was macht den Z8 aus?

Die Registeranzahl variiert in der Z8-Reihe zwischen 128 und 4096 Registern mit der Bandbreite von 8-Bit, wobei jedes dieser Register als „General-Purpose-Register“ (kurz: GPR) angesehen wird. Es gibt folglich keine Akkumulatoren oder ähnlich speziellen Registern, dennoch sind einige für gewisse Funktionen vorreserviert (0-3: Port-Register, die letzten 16: Steuerregister).

Für die Kommunikation nach außen bietet die Z8-Reihe vier Ports, wobei nicht festgelegt ist welcher Port als Ausgangs- bzw. Eingangsport fungiert.

Für Aufgaben, die zeitabhängig laufen müssen oder eine gewisse Anzahl von Iterationen benötigen, gibt es zwei Funktionseinheiten, die entweder als Timer oder als Zähler konfiguriert werden können.

Wie in jedem Mikroprozessor auch gibt es die ALU und eine Interrupt-Einheit.

Einer der größten Unterschiede zu herkömmlichen CPUs bzw. Mikroprozessoren ist die Abspeicherung von Daten und dem Programm. In der Z8-Reihe teilen sich Daten und Programm gemeinsam einen Speicher, der 64kB groß sein kann. Der Speicher kann nur einmal beschrieben werden (ROM-Speicher).

3 Programmierung des Z8

Es gibt folgende Möglichkeiten den Z8 zu programmieren:

- Mit Maschinencode (binäre Programmierung)
- Assembler
- Mit Hilfe von Compilern auch durch Hochsprachen (z.B. C)

4 ADU

Als ADU bezeichnet man eine Schaltung, welche analoge Eingansspannungen in diskrete Werte umwandelt. Hierzu wird die zu messende Spannung schrittweise mit einer Referenzspannung Vref verglichen. Der Messwert bezeichnet das Verhältnis der gemessenen Spannung zur Referenzspannung. Die maximal messbare Spannung ist somit durch die Referenzspannung beschränkt.

Der bei uns verwendete ADU wandelt das analoge Eingabesignal in ein digitales 8-Bit Signal um. Da es nur mögliche 256 Werte bei 8-Bit gibt, muss man drauf achten, dass man vor Einsatz des ADUs den analogen Wertebereich richtig skaliert hat (Minimalspannung =

0, Maximalspannung = 255). Desweiteren besitzt der ADU 2 getrennte Eingabekanäle, die durch das Setzen eines Flags ausgewählt werden können. Die Umwandlung beginnt mit einen eingehenden Low-Flanke (vorher muss es eine High-Flanke gegeben haben!) an dem „CEN“ Eingang. Sobald der „INT“ Ausgang auf Low schaltet, ist die Umwandlung abgeschlossen und können von dem Programm ausgelesen werden. Die Gültigkeit ist bis zur nächsten High-Flanke des „CEN“ gewährleistet. In unserem Fall kann das Abfragen des „INT“ Signales durch ein NOP-Befehl ersetzt werden.

5 Ziel des Versuchs

Es ging darum die Hardwarestruktur und den Befehlssatz der Z8 Reihen kennen zu lernen. Das schließt das Sammeln von Erfahrung bzgl. der Programmierung auf der Maschineebene ein sowie das Testen unter Echtzeitbedingungen. Zusätzlich werden wir Grundkenntnisse zur Programmierung des ADU-Interfaces kennenlernen und anwenden.

6 Aufbau des Versuchs

Da der Programmspeicher im Z8 nur einmal beschrieben werden kann (ROM), wurde in dem Versuch ein Emulator genutzt (Z86CCP00ZEM). Anderenfalls müsste man nach jedem Beschreiben des Z8s einen neuen Mikrokontroller-Chip nutzen, da der Alte nicht wiederbeschreibbar ist. Der Grund für die Entscheidung den Z8 zu emulieren und nicht zu simulieren ist Folgender: Bei der Simulation wird versucht ein Teil des originalen Problems wiederzugeben, wobei die Simulation nicht zu 100% mit dem eigentlichen Hardwareverhalten übereinstimmen muss (so tun als ob). Der Emulator kann durch einen Z8 ersetzt werden, ohne das Unterschiede bemerkbar sind (nachahmen).

Neben dem Emulator wurde eine weitere Platine genutzt, auf der sich der ADU-Wandler, ein 7-Segment-Anzeige-Modul und ein Balken-Anzeigen-Modul sich befinden. Der Ausgang des ADUs ist über ein Adapterkabel mit dem Emulator über den Port 0 verbunden. Die beiden Anzeigen-Module sind über Port 2 und Port 1 mit dem Emulator verbunden.

7 Durchführung

7.1 Aufgabe 1

Diese Aufgabe befasste sich damit, dass wir zyklisch den ADU starten und dessen Ergebnis auf die Balken-Anzeigen mappen. Dazu musste Port 0 als Eingabeport konfiguriert werden und Port 2 als Ausgabeport. Port 3 wurde so eingestellt, dass er als Ausgabeport fungiert und Port zwei mit aktiven pull-ups betrieben wird. Das Skalieren der Analogen Signale war bereits vorgegeben. Als Eingabekanal wurde Kanal 0 ausgewählt. Nach der Initialisierung der Ports/Register, wurde in einer Schleife zyklisch eine Low-Flanke ausgelöst. Da Port 2 sowohl die 7-Segment-Anzeigen als auch den Balken-Anzeige ansteuert, mussten die unteren 4-Bit genullt werden, sodass auf der 7-Segment-Anzeige keine Ausgabe erscheint.

```
TITLE    "Aufgabe 1"
org      0ch
```

```
        LD      P01M, #45h
        LD      P2M, #00h
        LD      P3M, #01h

        LD      P0, #00h
        LD      P1, #00h
        LD      P2, #00h
        LD      P3, #00h

m1:
        LD      P3, #20h
        LD      P3, #00h
        NOP
        LD      04h, P0
        AND     04h, #0F0h
        LD      P2, 04h
        JP      m1

        END
```

7.2 Aufgabe 2

Hier wurde Aufgabe 1 so erweitert, dass auch die 7-Segment-Anzeige funktionierte. Dazu musste der Port 1 ebenfalls als Ausgabeport definiert werden und das digitale Signal in den BCD Code umgewandelt werden. Der BCD Code versteht nur Zahlen von 0 – 9. Die größte Zahl in unserem Fall ist die 255, wodurch es maximal 3 BCD Zahlen geben kann. Anfangen tut man nun so, dass man schaut wie oft die Zahl 100 in die gegebene Zahl passt. Ist die Zahl kleiner als 100, wird danach geprüft, wie oft die Zahl 10 in die noch verbliebende Zahl reingeht. Ist die Zahl dann kleiner als 10, hat man die letzte Stelle des BCD Codes.

Dadurch dass Port 1 die 10er und 1er Stellen aufnehmen wird, muss zu Anfang die 10er Stelle gedreht werden, sodass diese in den höherwertigen Bits liegen. Anschließen werden die 10er und die 1er Stellen mit einem OR verbunden und an die Ausgabe weitergeleitet. Die 100er Stelle wurde mit dem Ergebnis aus Aufgabe 1 mit einem OR verbunden, sodass die Ausgabe auf der 7-Segment-Anzeige und der Balken-Anzeige gleichzeitig stattfand.

```
TITLE   "Aufgabe 2"
org     0ch

        LD      P01M, #45h
        LD      P2M, #00h
        LD      P3M, #01h

        LD      P0, #00h
        LD      P1, #00h
        LD      P2, #00h
        LD      P3, #00h
```

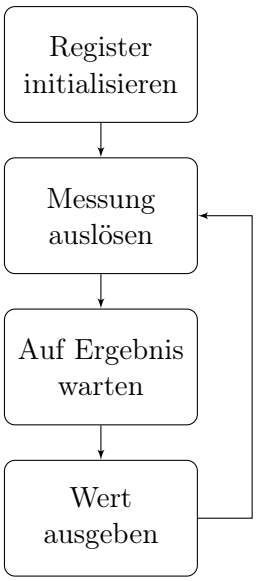

Abbildung 1: Aufgabe 1

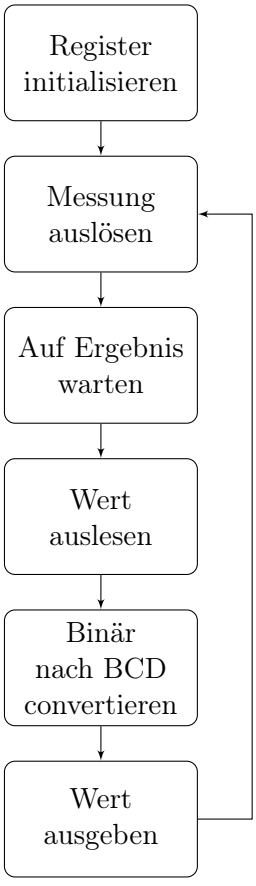

Abbildung 2: Aufgabe 2

```
m1:
        ; High-Low-Flanke generieren
        LD      P3, #20h
        LD      P3, #00h
        NOP

        ; Digitalwert auslesen
        LD      04h, P0
        LD      05h, 04h

        ; Zaehlregister initialisieren
        LD      06h, #00h
        LD      07h, #00h

        ; Hauptschleife
m2:
        CP      04h, #100                       ; -100
        JP      ULT, m3
        SUB     04h, #100
    INC     07h
        JP      m2

        ; - 10
m3:
        CP      04h, #10
        JP      ULT, m4
        SUB     04h, #10
        INC     06h
        JP      m3

        ; <10
m4:
        SWAP    06h
        OR      04h, 06h
        LD      P1, 04h

        ; Segment
        AND     05h, #0F0h
        OR      05h, 07h
        LD      P2, 05h
        JP      m1

        END
```

7.3 Aufgabe 3

Aufgabe 3 bestand darin die Aufgabe 2 so zu erweitern, dass man beide Eingabekanäle (CH0, CH1) verwenden konnte. Der Wechsel sollte ca. alle 4 Sekunden stattfinden. Um die 4 Sekunden zu implementieren, nahmen wir eine 16-Bit Zählschleife, die ca. 4 Sekunden Laufzeit auf dem System beansprucht. Innerhalb der Schleife wurde dann der Teil aus Aufgabe 2 ausgeführt. Zum Ende der Hauptschleife wurde einfach der Wert, der bestimmt welcher Eingabekanal im nächsten Durchgang genutzt wird, mit einem XOR gedreht und in A0 geschrieben.

```
        TITLE   "Aufgabe 3"
        org     0ch

        LD      P01M, #45h
        LD      P2M, #00h
        LD      P3M, #01h

        LD      P0, #00h
        LD      P1, #00h
        LD      P2, #00h
        LD      P3, #00h

        LD      08h, #00h

        ; Hauptschleife (Endlos)
m1:

        ; Zaehlschleife
s0:
        ; High-Low-Flanke generieren
        LD      P3, #20h
        LD      P3, 08h
        NOP

        ; Digitalwert auslesen
        LD      04h, P0
        LD      05h, 04h

        ; Zaehlregister initialisieren
        LD      06h, #00h
        LD      07h, #00h

        ; Hauptschleife
m2:
        CP      04h, #100                       ; -100
        JP      ULT, m3
        SUB     04h, #100
        INC     07h
```

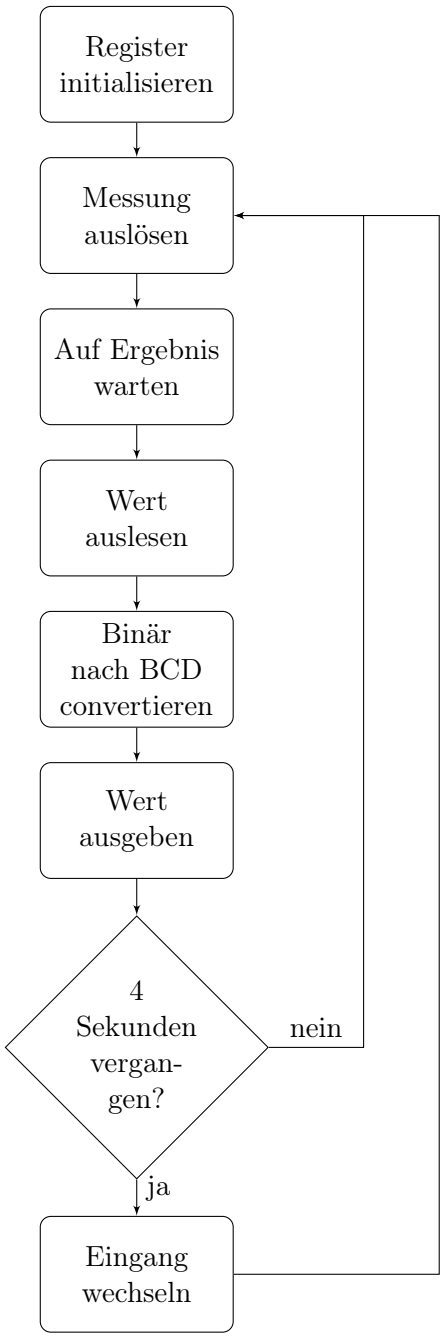

Abbildung 3: Aufgabe 3

```
        JP      m2

        ; - 10
m3:
        CP      04h, #10
        JP      ULT, m4
        SUB     04h, #10
        INC     06h
        JP      m3

        ; <10
m4:
        SWAP    06h
        OR      04h, 06h
        LD      P1, 04h

        ; Segment
        AND     05h, #0F0h
        OR      05h, 07h
        LD      P2, 05h

        ; Schleifenkopf
        DECW    10h
        JP      NZ, s0

        ; Eingang wechseln
        XOR     08h, #10h

        JP      m1

        END
```