Hard- und Software-Engineering

WS 2006/2007

Praktikumsaufgaben

Versuch 1: Elementare Assemblerprogrammierung

Software: Atmel AVR Studio (http://www.atmel.com).

Der typische Entwicklungsgang:

1. Entwickeln auf PC,
2. Herunterladen auf Starterkit,
3. Laufenlassen und Fehler suchen. Typischerweise zurück zu Punkt 1.

In Assembler programmieren:

- der Assembler ist eine symbolische Maschinensprache.
- jedem Befehl entspricht eine Abkürzung (Assembler Mnemonic).
- weitere symbolische Bezeichnungen können nach eigener Wahl eingeführt werden:
  - Deklarationen (= bloße Ersetzungen; es wird nichts gespeichert): **equ**-Anweisungen (symbolischer Name = Konstante), **def**-Anweisungen (symbolischer Name = Register),
  - Reservieren von Speicherplatz: **byte**-Anweisung,
  - Konstanten: **db**- und **dw**-Anweisungen (Konstanten als Bytes oder Worte). Allgemeinbegriff: **data**-Anweisung.
  - Makros (= Befehlsfolgen zur Mehrfachnutzung unter einem gemeinsamen Namen): Anweisungen **macro** und **endmacro**.

Eine Befehlszeile kann folgende Angaben enthalten:

- eine Marke (Label). Nur dann erforderlich, wenn der Befehl Sprungziel ist.
- die Mnemonic des jeweiligen Befehls (obligatorisch),
- ggf. erforderliche Operanden (symbolisch oder als Direktwerte),
- ggf. einen Kommentar (durch ein Semikolon (;) abgetrennt).

Moderne Assembler sind Freiformassembler; sie erzwingen typischerweise keine besondere Formatierung. Es dürfen beliebig viele Leerzeichen eingefügt werden (Anwendung: zum Erzielen eines gefälligen Aussehens). Groß- oder Kleinschreibung ist gleichgültig.

Versuchsaufbau:

- Geräte: Starterkit AVR STK 500, Oszilloskop.
- Tasten des Starterkits an Port A
- LEDs des Starterkits an Port D,
- Impulsausgaben über Port C.

Tastenabfrage und LED-Ansteuerung sind beide aktiv Low.

Eine elementare Befehlsausstattung:

- LDI. Lädt einen Direktwert in ein Register. Beispiel: LDI r17, $22.
- IN. Eingabe in ein Register. Beispiel: IN r17, pind.
- OUT. Ausgabe eines Registerinhaltes. Beispiel: OUT portd, r17.
- RJMP. Unbedinge Verzweigung. Beispiel: RJMP anfang.
- COM. Bitweise Negation (Einerkomplement). Beispiel: COM r17.
- RCALL. Unterprogrammruf. Beispiel: RCALL warten.
- RET. Rückkehr aus Unterprogramm.
- DEC. Registerinhalt um 1 vermindern. Beispiel: DEC r17.
- SBIW. Einen Direktwert von einem 16-Bit-Wort abziehen, das in zwei aufeinanderfolgenden Registern enthalten ist. Beispiel: SBIW r26, 10.
- BRNE. Verzweigen, wenn Operanden ungleich bzw. Ergebnis ungleich Null. Beispiel: BRNE anfang.
- BREQ. Verzweigen, wenn Opernaden gleich bzw. Ergebnis gleich Null. Beispiel: BREQ ende.
- SBI. Setzen Bit in E-A-Port. Beispiel: SBI porta,3.
- CBI. Löschen Bit in E-A-Port. Beispiel: CBI porta,3.
- SBIC. Bit in E-A-Port abfragen und Folgebefehl überspringen, wenn Null (C = cleared). Beispiel: SBIC porta,3.
- SBIS. Bit in E-A-Port abfragen und Folgebefehl überspringen, wenn Eins (S= set). Beispiel: SBIS porta,3.
- NOP. Nichts tun; nur Zeit verbrauchen (Leerbefehl).

Hinweis: Um den Unterprogrammruf nutzen zu können, ist der Stack zu initialisieren.

Ablauf:

1. Starterkit anschließen und einrichten.
2. AVR Studio starten. Neues Projekt.
3. Ggf. benötigte Dateien hinzufügen.
4. Quelltext erfassen. Ggf. bereits vorhandene Textteile in die Assemblerdatei übernehmen.
5. Übersetzen und ggf. im Simulator laufen lassen.
6. Ggf. in den Prozessor des Starterkits übertragen (Programmieren) und dort laufen lassen.

1. Teilversuch: Tastenabfrage und LEDAnzeige

a) Jede Betätigung der acht Tasten soll dazu führen, daß die betreffenden LEDs für die Dauer des Tastendrucks aufleuchten.

b) Alle LEDs sollen leuchten. Jede Betätigung der acht Tasten soll dazu führen, daß die betreffenden LEDs für die Dauer des Tastendrucks verlöschen.

c) Jede Tastenbetätigung soll dazu führen, daß die betreffende LED abwechselnd ein- und ausschaltet. Klappt das so einfach? Was ist erforderlich?

   Vorübung:
   Mit Software Zeit herstellen.

d) LED-Lauflicht.

2. Teilversuch: Impulserzeugung / Pulsweitenmodulation

Es ist eine ununterbrochene Impusfolge mit gleichbleibender Periodendauer abzugeben. Die Impulsbreite soll durch Tastenbetätigung einstellbar sein. Mit diesen Impulsen sind die LEDs anzusteuern (Helligkeitssteuerung). Impulsfolgefrequenz: rund 400 Hz. Anzeige mittels Oszilloskop. Des weiteren Demonstration der Drehzahlsteuerung eines Gleichstrommotors. Die Impulslänge soll durch die Tasten des Starterkits einstellbar sein. 1. Taste: 1/8 der Periodendauer, 2. Taste: 2/8 der Periodendauer usw. (8. Taste = volle Periodendauer = Dauersignal).

Tasteneingabe über Port A.

LED-Anzeige über Port D. Bei 1/8 Periodendauer ist nur die erste LED an, bei 2/8 die erste und die zweite usw. (Bandskala).

Ausgabe: über Port C (aktiv Low). Impuls = 00H, Pause = FFH.

Funktionsnachweis: Oszilloskop, Motoransteuerung (Experimentiertafel).

1. Initialisieren. Impulsdauer auf Null. Null ausgeben.

2. Tasten abholen.

3. Wenn irgendeine Taste betätigt, dann neue Impulsdauer laden.

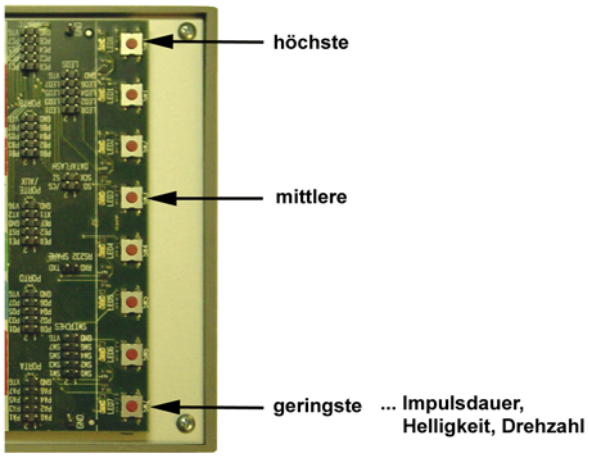
4. Wenn Impulsdauer ungleich Null, dann Impuls einschalten.

5. Bit für Bit der Impulsdauer abfragen. Ist der Impuls eingeschaltet, das betreffende Bit der LED-Ausgabe setzen (so daß zugehörige LED leuchtet). Ist der Impuls ausgeschaltet, das betreffende Bit der LED-Ausgabe löschen (so daß zugehörige LED nicht leuchtet).

6. Wenn Bit gesetzt, dann Impuls ausschalten (nur nicht, wenn Bit 7).

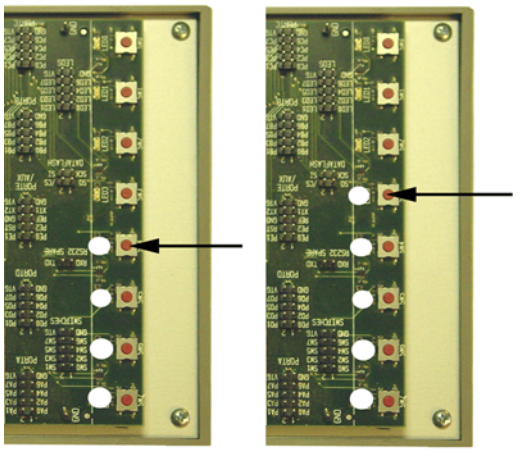
7. Wenn alle Bits abgefragt, zurück zu Schritt 2.

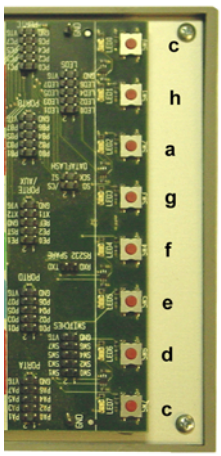
Alles so einstellen (Simulator), daß die Periodendauer ungefähr 2,5 ms entspricht (400 Hz).



3. Teilversuch: Musikinstrument

Die Tasten werden zyklisch abgefragt. Eine betätigte Taste veranlaßt die Abgabe eines Tonsignals (Ports B und D; aktiv Low), das über Lautsprecher oder Schallgeber hörbar gemacht werden kann. Das Tonsignal ist eine symmetrische Rechteckwelle (Tastverhältnis 50:50) mit jeweils bestimmter Frequenz (gemäß chromatischer Tonleiter (s. Abb.; Frequenzangaben in Hz). Wenn kein Ton abgegeben wird, Ausgänge auf High (damit der angeschlossene Lautsprecher keinen Gleichstrom zieht). Dieses Projekt bitte unter eigenem Namen wiederfindbar abspeichern - im nächsten Versuch geht‘s damit weiter ...



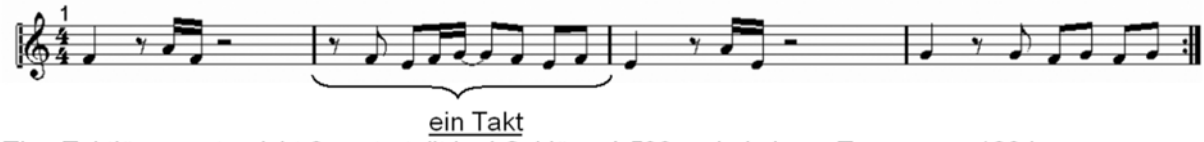


	f [Hz]	$t_{p/2}$ [µs]
c	525,25	952
h	493,88	1012
a	440	1136
g	392	1276
f	349,23	1432
e	329,63	1517
d	293,67	1703
c	261,63	1911

Zusatzaufgabe: Noten abspielen

In Fortsetzung von Teilversuch 3 (Musikinstrument) sollen Noten abgespielt werden. Die Notenfolge steht (als Zeichenkette codiert) im Programmspeicher.

Kurzausbildung Musik



Eine Taktlänge entspricht 2s unterteilt in 4 Schläge á 500ms bei einem Tempo von 120 bmp

Noten

𝅝	=	ganze Note	= 4/4 =	2000 ms
𝅗𝅥	=	halbe Note	= 2/4 =	1000 ms
♩	=	viertel Note	= 1/4 =	500 ms
♪	=	achtel Note	= 1/8 =	250 ms
𝅘𝅥𝅯	=	sechzehntel Note	= 1/16 =	125 ms

Pausen

Tempo: 120 bpm

𝄻	=	ganze Pause	= 4/4 =	2000 ms
𝄼	=	halbe Pause	= 2/4 =	1000 ms
𝄽	=	viertel Pause	= 4/4 =	500 ms
𝄾	=	achtel Pause	= 4/4 =	250 ms
𝄿	=	sechzehntel Pause	= 4/4 =	125 ms

andere Zeichen:

𝄇 = Wiederholungszeichen

Note	Frequenz (Hz)	Halbperiode (µs)	$^1/_{16}$ Note (125 ms) entspricht ... Perioden
c	525,25	952	66
h	493,88	1012	62
a	440	1136	55
g	392	1276	49
f	349,23	1432	44
e	329,63	1517	41
d	293,67	1703	37
c	261,63	1911	33

Codierungsbeispiel:
Jede Note wird zusammen mit der nachfolgenden Pause in 3 Bytes codiert. Eingabe als Zeichenkette (db-Direktive):

1. Notenwert (Frequenz): c, d, e, f, g, a, h, k.
2. Notendauer: 1, 2, 4, 8, 6.
3. Pausendauer: 1, 2, 4, 8, 6.

Ende der Notenkette: 0.

Versuch 2: Eingebaute Peripherie ausnutzen

Teilversuch 1: Nutzung der Zähler/Zeitgeber
Wiederholung der entsprechenden Aufgaben von Versuch 1, aber Implementierung der Zeitfunktionen mit Hilfe der Zähler/Zeitgeber.

Teilversuch 2: Nutzung der seriellen Schnittstelle
Ausgeben von Zeichen auf ein Bildschirm-Terminal. Empfangen und Anzeigen von Zeichen.

Teilversuch 3: A/D-Wandler
Digitale Anzeige einer variablen Spannung (binär auf den LEDs des Starterkits, ggf. auch auf Bildschirm-Terminal und auf LCD).

Vorversuch: LCD-Punktmatrixanzeige
Stellen Sie auf einer Punktmatrixanzeige einen Festtext nach eigener Wahl dar. Wir beschränken uns zunächst auf reine Ausgabeabläufe, wobei die jeweilige Kommandoausführungszeit im Prozessor abgewartet wird.

Versuch 3: CPLD-Schaltungsentwurf mit Xilinx ISE Entwicklungsumgebung

Teilversuch 1: Entwerfen Sie einen Siebensegmentdecoder für die BCD-Zahlen 0...9
Zur Erprobung ist ein BCD-Zähler (als fertiges Funktionselement aufzurufen) vorgeschaltet. Hierzu können die in den Digitaltechnik-Übungsstunden erarbeiteten Ergebnisse ausgenutzt werden. Es ist jedoch auch möglich, intuitiv zu entwerfen (Abb. 1) und die Optimierung dem System zu überlassen. Erprobung: Mit entsprechenden Übungsplattformen bzw. Starterkits (Abb. 2 bis 4).

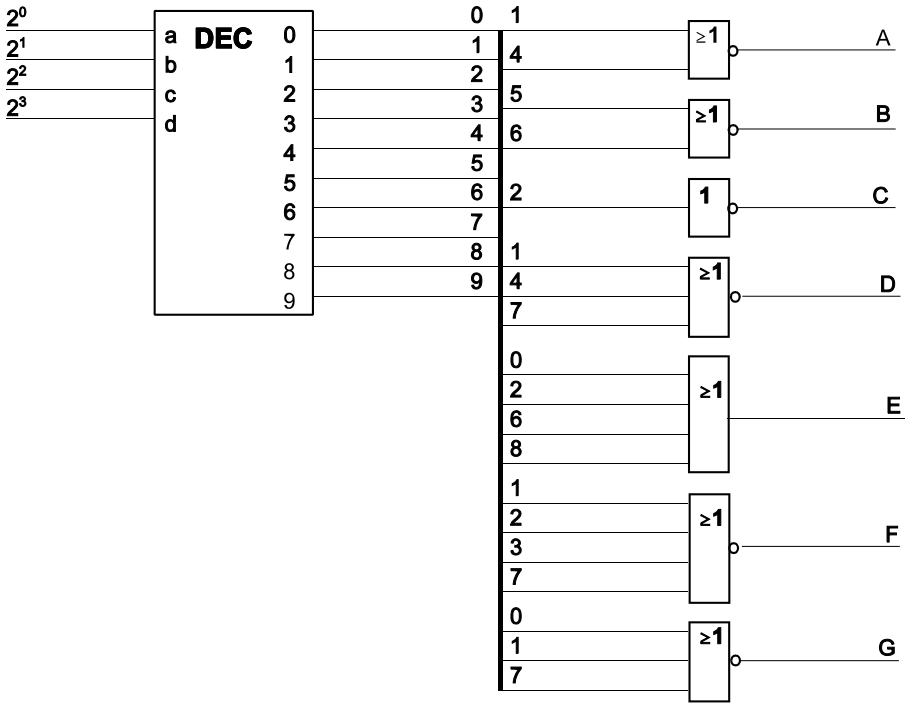


Abb. 1 Ein intuitiv entworfener Siebensegmentdecoder (Prinzipschaltung). Hinweise: 1. dem BCD-Decoder wird ein Dezimalzähler vorgeschaltet. 2. Die Siebensegmenanzeigen werden aktiv Low angesteuert (Negation aller Ausgangssignale)

Teilversuch 2:
Ersetzen Sie den fertigen BCD-Zähler durch einen (vollsynchronen) Binärzähler, der vorwärts und rückwärts zählen kann (ein einziger Takt + zwei Steuersignale (FWD, BWD)).

Teilversuch 3:
Ergänzen Sie den Zähler durch eine Steuerschaltung, die den Incrementalgeber der Übungsplattform auswertet (Abb. 5).

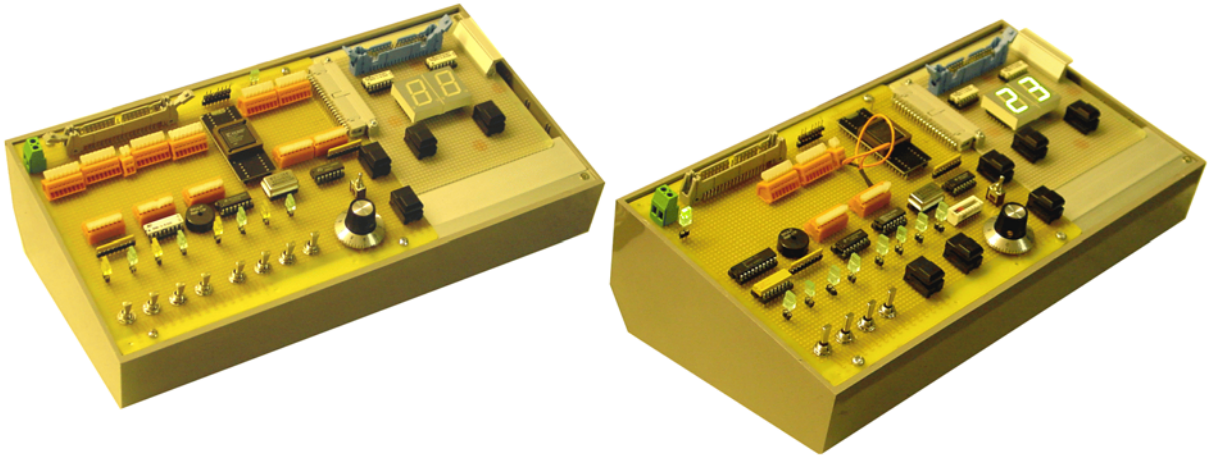


Abb. 2 CPLD-Übungstafeln

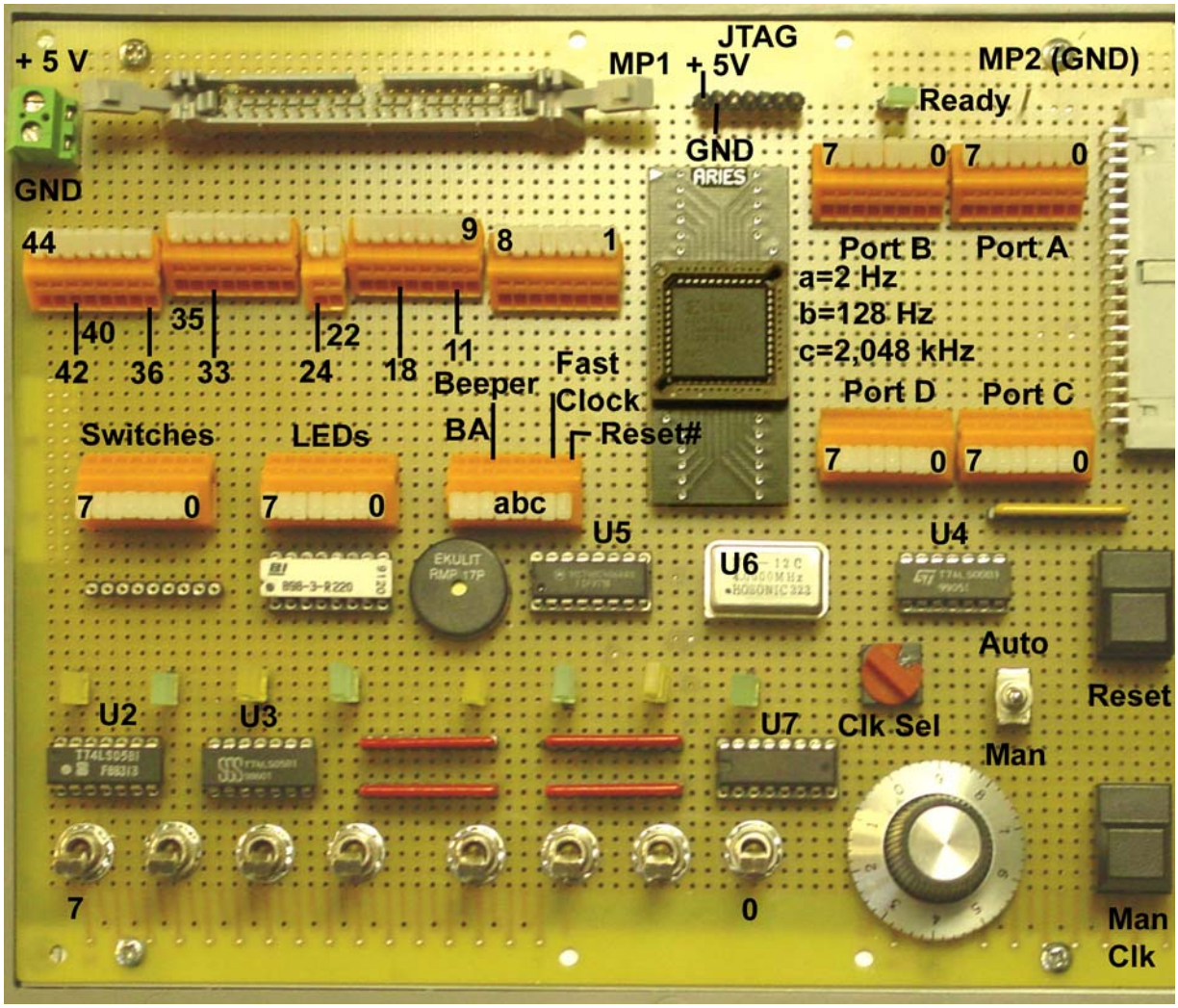


Abb. 3 CPLD-Übungstafel 06a. Übersicht

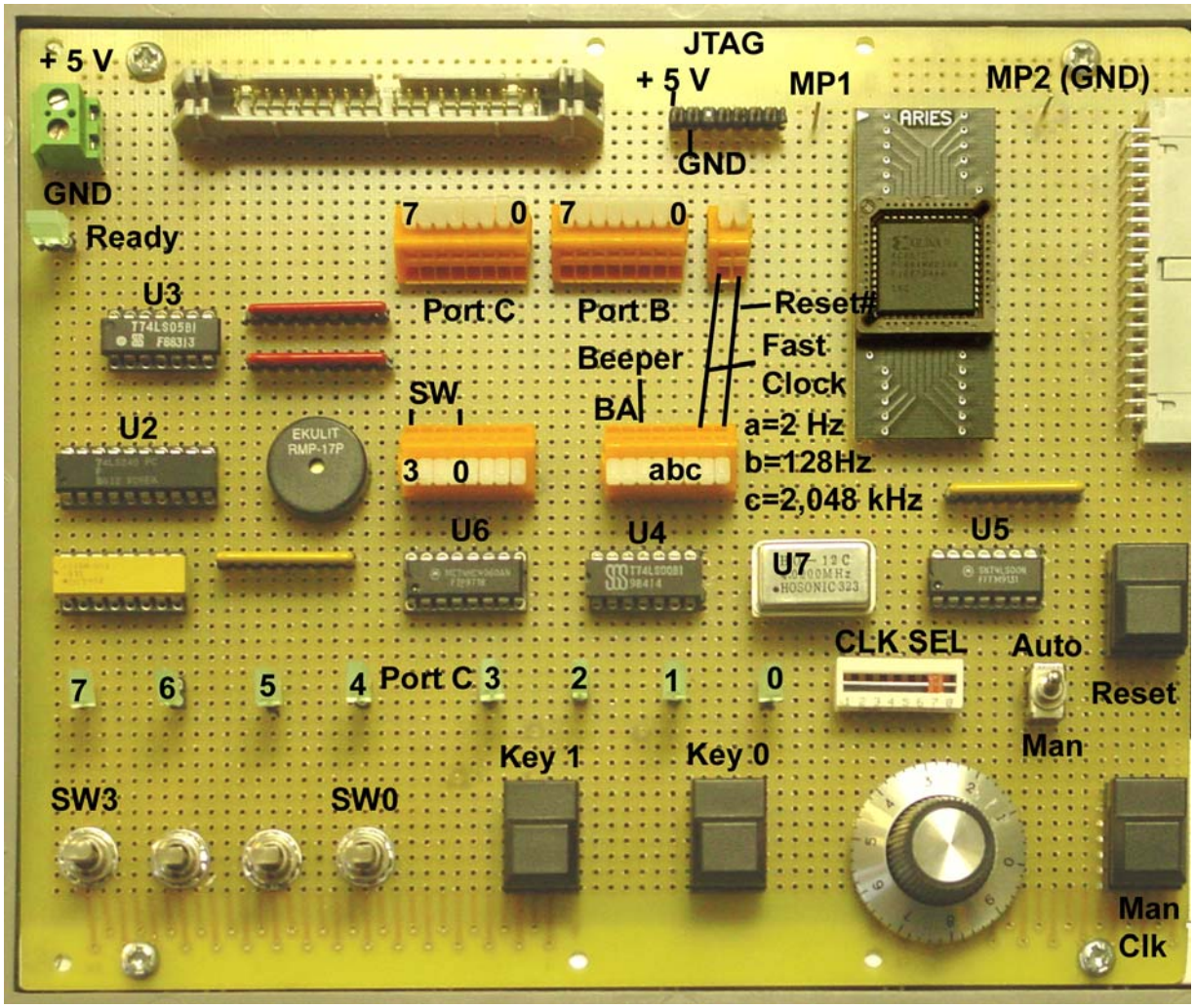


Abb. 4 CPLD-Übungstafel 06b. Übersicht

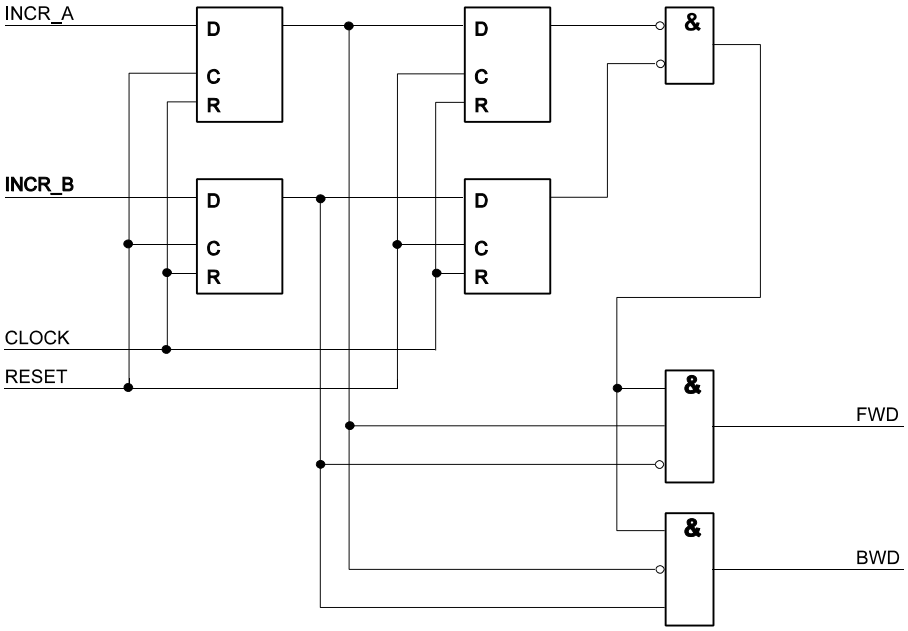


Abb. 5 Auswertung Incrementalgeber (Prinzipschaltung)