sfb

Bildungszentrum
Mensch
Technologie
Wirtschaft

Mikrocontroller

ABW-Fachkurs EM

167011.01

PDF-Auszug aus

SFB

Mikrocontroller
ABW-Fachkurs EM

Grundlagen Mikrocontroller

Dieser unvollständige PDF-Auszug des Lehrmittels enthält nur die Kapitel der folgenden Autoren:

Thomas Pfründer (TP) und
Hansjörg Kern (HJK)

sfb

Bildungszentrum
Mensch
Technologie
Wirtschaft

Mikrocontroller

ABW-Fachkurs EM

Dieses Lehrmittel ist auf umweltschonendes Papier gedruckt.
Für Kopie und Laser. Ohne Chlor und Chlorverbindungen gebleicht.
Aus Sägerestholz und Durchforstungsholz. Ohne optische Aufheller.

Herausgeber
sfb Bildungszentrum
Bernstrasse 394
8953 Dietikon

Druck
M. Erne Offsetdruck AG, Zürich

167011/01/1999

Elektronik

Mikrocontroller

Lehrplan EM

Mikrocontroller

Inhalt

Theorie/Praktikum

Lernziel
Als Teilnehmer lernen Sie,

Lektion 1 Einführung
Begrüssung
Organisation
Einleitung / Überblick

- sich in der Klasse zu äussern
- sich in den örtlichen Verhältnissen zurechtzufinden
- sich ein klares Bild über den Kursverlauf und die Hilfsmittel zu machen

Lektion 2 Digitaltechnik
Einführung Digitaltechnik

- die Unterschiede zwischen Analog- und Digital-Signalen zu nennen
- die wichtigsten Anwendungen der Digital-technik zu nennen
- die Grundbegriffe in der Digitaltechnik zu nennen

Lektion 3 Hardware des u-Controllers
Aufbau eines u-Controller-Systems
Controller und externe Peripheriebausteine

- die Begriffe Hardware, Software zu unterscheiden
- die Hardware und den Aufbau eines uC-Systems zu beschreiben
- die Signalflüsse anhand eines Blockschemas zu erkennen

Lektion 4 Software des u-Controllers
Register eines uC-Bausteines
ROM- und RAM-Speicher
Programmablauf,-ausführung

- den prinzipiellen Ablauf eines Programmes zu erklären
- die Funktion von ROM- und RAM-Speicher während der Programmausführung zu erkennen
- zwischen Programmspeicher und Datenspeicher zu unterscheiden

Lektion 5,6 Einführung LEMPS
System-Inbetriebnahme
Entwicklungsumgebung
Terminalprogramm
Demo-Programme

- das LEMPS-System in Betrieb zu nehmen
- sich in der PC-Entwicklungsumgebung IDE zurechtzufinden
- ein Demo-Programm zu laden und zu starten

Lektion 7,8 Zahlensysteme
Binärsystem
Hexadezimalsystem

- die wichtigsten Codearten aufzuzählen
- Codewandlung zu erklären
- Binär-, BCD- und Hexcode zu berechnen und zu wandeln

Lektion 9,10 Grundlagen Schaltalgebra
Verknüpfungsarten / Logik
UND, ODER, NICHT, Inverter
NAND, NOR, EXOR

- die Gesetze und Rechenregeln der Schaltalgebra zu nennen

Lektion 11 Digitale Bausteine
Elektrische Eigenschaften
Logikfamilien

- die wichtigsten elektrischen Eigenschaften von Logikbausteinen zu nennen
- die Unterschiede der verschiedenen Logikfamilien aufzuzählen

Lektion 12 Messen von Signalen
Pegel
verschiedene Signalformen

- mit Hilfe von verschiedenen Messgeräten die typischen Signale in einem uC-System zu messen

Lektion 13 Binärspeicher

RS-Flip-Flop
JK-Flip-Flop
D-Flip-Flop
I/O Latch
ROM, RAM

- das Funktionsprinzip eines binären Speichers zu nennen
- Anwendungen von Speichern in einem uP-System aufzuzählen

Lektion 14,15 Funktionen vom LEMPS

uC-System-Blöcke
integrierte Peripherie
externe Peripherie
Testprogramme

- die Begriffe und Funktionen der LEMPS-Bausteine und uC-Blöcke zu nennen
- die integrierte Peripherie von externer Peripherie zu unterscheiden
- die Funktion der Peripherie mit Hilfe von Testprogrammen zu verfolgen

Lektion 16 Zwischentest

Besprechung

- das eigene Wissen zu überprüfen und Lücken zu schliessen

Lektion 17,18 Programmierung des HC11

CPU-Register, Akku A, B,
Register X, Y, SP, PC, CCR
Memory Map, internes und externes ROM / RAM
RESET, Restart Vector
Zustand nach RESET

- die Funktion der Register zu erklären
- die Speicherbelegung, Vorgänge bei der Initialisierung zu beschreiben
- den Systemzustand nach einem RESET zu beschreiben

Lektion 19,20 Instruction Set des HC11

Instruction Set des HC11
Befehle LDAA, STAA, JMP
Jumps, Subroutinen
Adressierungsarten des HC11

- die Bedeutung der mnemonischen Schreibweise der Befehle zu erkennen
- die Assemblerbefehle LDAA, STAA, JMP anzuwenden
- verschachtelte Programmabläufe zu erklären
- einige Adressierungsarten des HC11 zu nennen

Lektion 21 Problemanalyse

Struktogramme
Fussdiaglamme

- einige Werkzeuge zur Problemanalyse zu nennen und zu beschreiben

Lektion 22 Programm eingeben

einfaches Programm erstellen

- die Handhabung des Terminalprogrammes zu vertiefen

Lektion 23,24 Entwicklungsumgebung

Editor, HC11-Assembler
Assembler-Direktiven
Download
Alternative: C- oder Pascal-Hochsprache
Programmformular

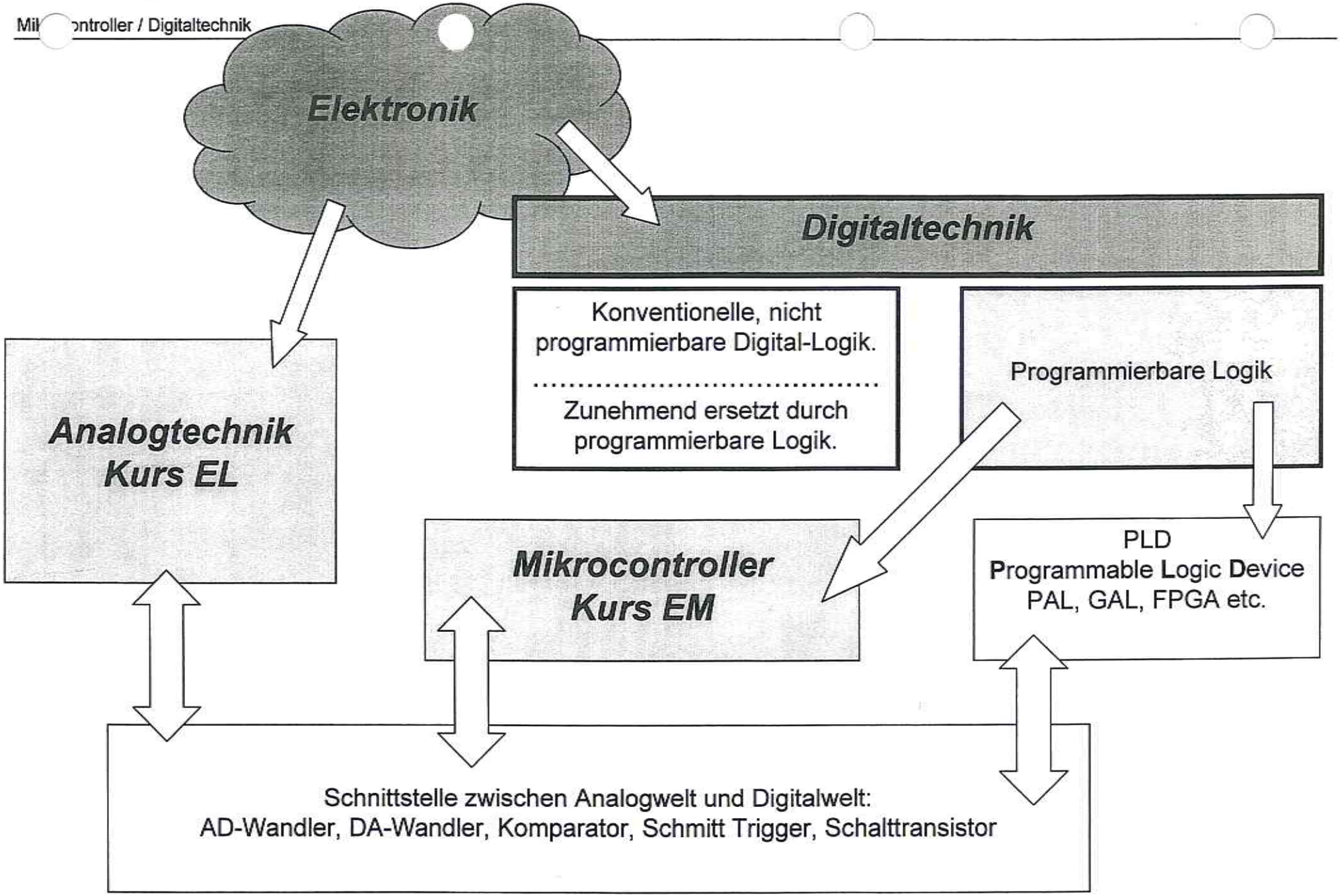
- die Funktionen des Editor- und Compiler-/Assemblerprogrammes zu nennen
- Quelltext zu assemblieren/compilieren und herunterzuladen
- die C- resp. Pascal-Hochsprache als Alternative zum systemabhängigen Assemblerdialekt zu nennen
- ein Programmformular im Editor zu eröffnen um ein neues Programm zu schreiben

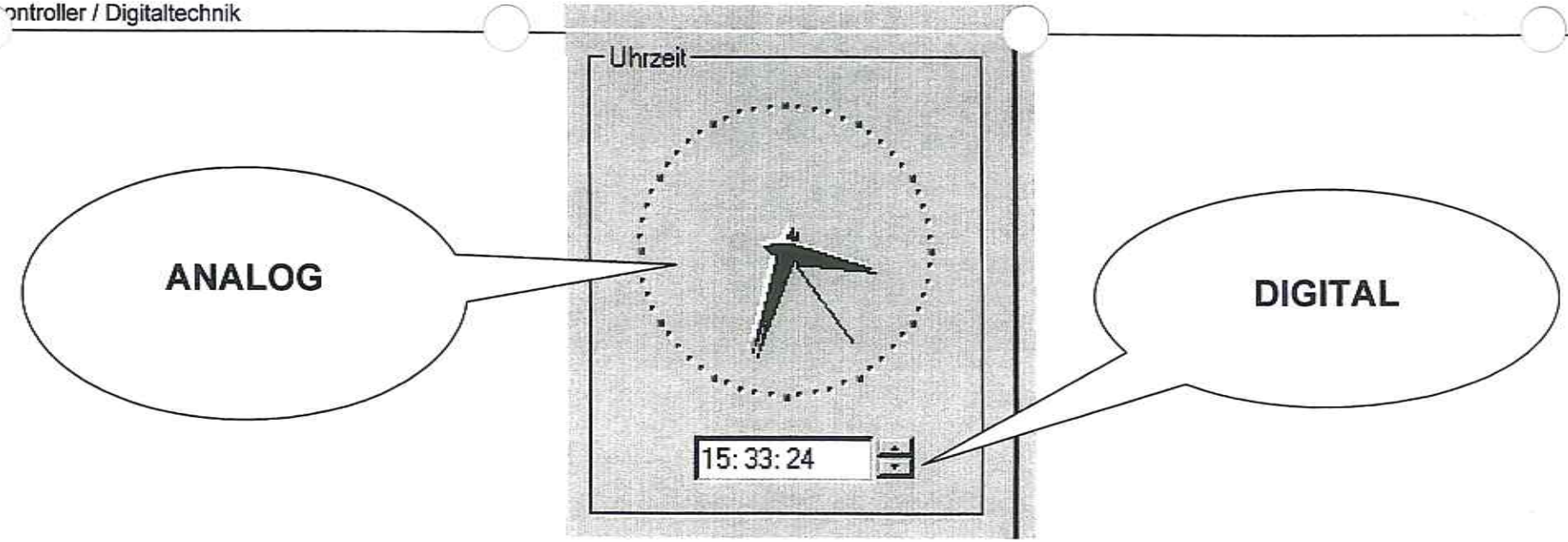
Lektion 25 - 27 Programmierung des HC11

LEMPS-Monitorsubroutinen
Lade-, Speicher-, Transferbefehle
Verzweigungen auf Null
Vergleichsoperatoren
Indizierte Adressierung

- die Funktion einiger Monitorroutinen des LEMPS zu nennen
- Lade-, Speicher-, Transferbefehle zu nennen
- weitere Programmelemente zu nennen und anzuwenden

Lektion 28 Zwischentest	
Besprechung	- das eigene Wissen zu überprüfen und Lücken zu schliessen
Lektion 29,30 Programmierung des HC11	
Unterprogramme mit JSR, RTS Stack-Anwendungen Logische Operationen Bitmanipulation	- weitere Programmelemente zu nennen und anzuwenden
Lektion 31 Analogsignale und uC	
analoge uC-Eingänge MUX, Kanalauswahl AD-Wandler PWM als DA-Wandler	- Eingabe-, Verarbeitungs- und Ausgabemöglichkeiten von Analogsignalen mit Hilfe des uC zu nennen
Lektion 32 Projektarbeit	
Projektarbeit vorstellen	- die verschiedenen Projekte kennen
Lektion 33 Zeitglieder	
Counter Timer PWM	- die Realisierung von Timerfunktionen mit Hilfe des uC zu beschreiben
Lektion 34 Progr.-Unterbrechungen	
Interrupt Interrupt Vectors COP, Watchdog	- die Echtzeit- und Ueberwachungsfunktionen des uC zu nennen und zu beschreiben
Lektion 35,36 Projektarbeit	
Projektstand	- den Projektstand zu formulieren - die Lücken für die Fortführung des Projektes zu schliessen
Lektion 37 - 39 Projektarbeit	
Arbeiten im Team	- die eigene Projektarbeit systematisch zu bearbeiten
Lektion 40 - 42 Projektarbeit	
Arbeiten im Team	- die eigene Projektarbeit systematisch zu bearbeiten - die Projektarbeit zu dokumentieren
Lektion 43 - 44 Präsentation	
Besprechung Projektarbeiten	- die Projektarbeit in der Gruppe zu präsentieren
Lektion 45 Kursabschluss	
Rückblick Ausblick	- ein Feedback über den ganzen Kurs zu geben - die Inhalte der weiteren ABW-Fachkurse, insbesondere des Kurses EV zu nennen


Elektronik
Digitaltechnik
Konventionelle, nicht programmierbare Digital-Logik.
Zunehmend ersetzt durch programmierbare Logik.
Programmierbare Logik
Analogtechnik Kurs EL
Mikrocontroller Kurs EM
PLD
Programmable Logic Device
PAL, GAL, FPGA etc.
Schnittstelle zwischen Analogwelt und Digitalwelt:
AD-Wandler, DA-Wandler, Komparator, Schmitt Trigger, Schalttransistor



Eigenschaften analoger Einrichtungen,
Vorteile, Nachteile:

.....
.....
.....
.....

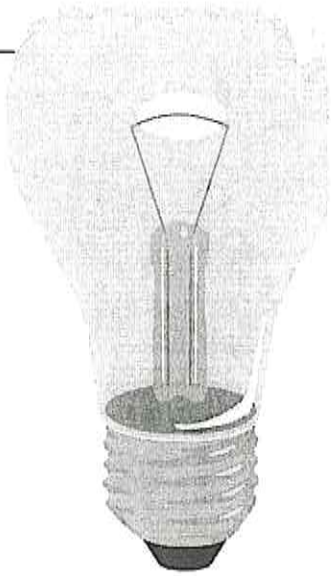
Eigenschaften digitaler Einrichtungen,
Vorteile, Nachteile:

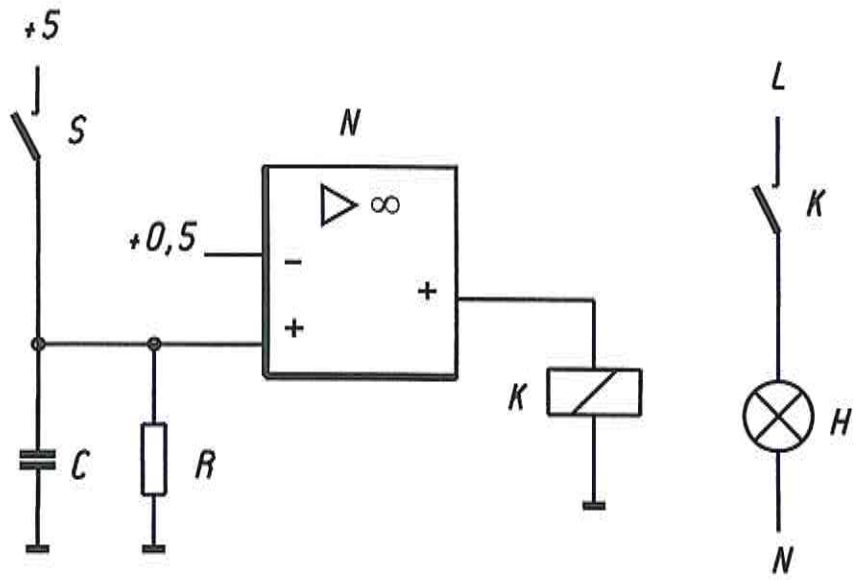
.....
.....
.....
.....

Weitere Geräte und Einrichtungen, die in analoger und digitaler Ausführung existieren:

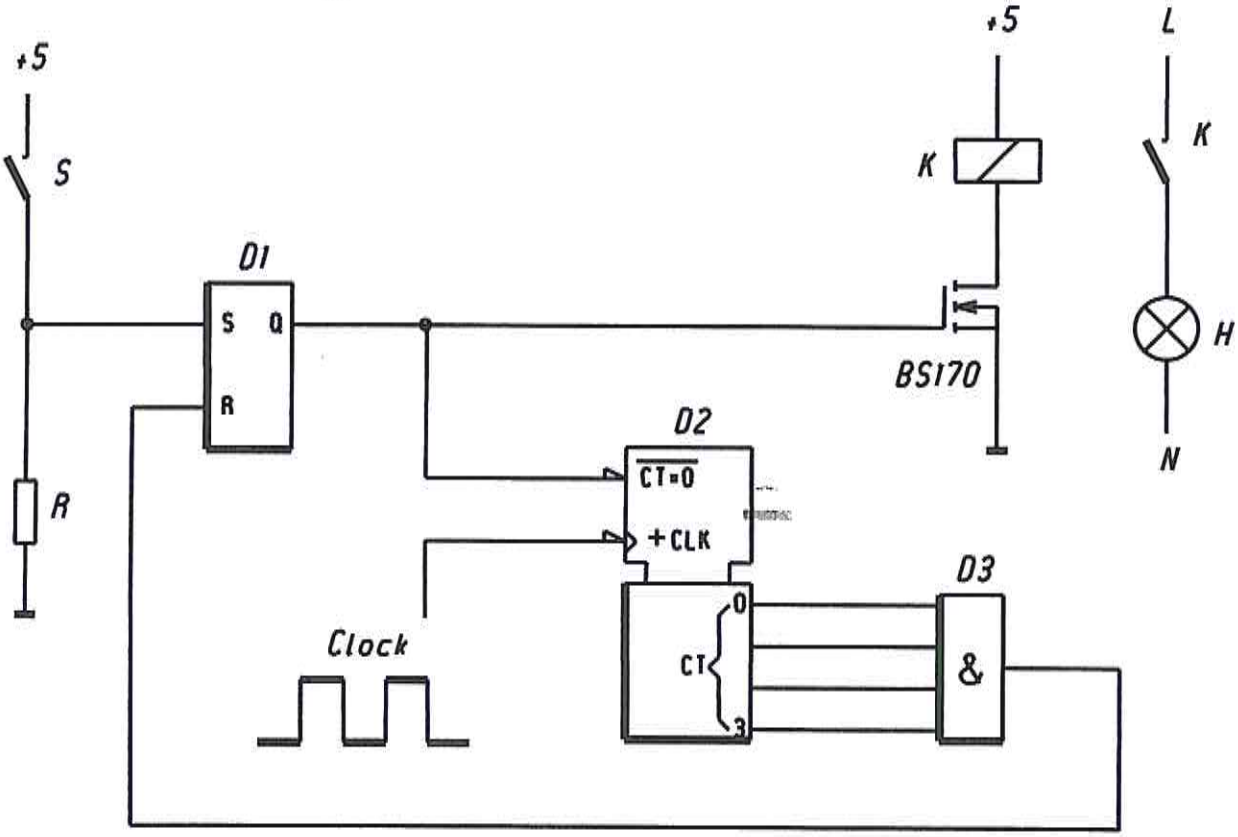
.....
.....

Licht-Timer in analoger Technik



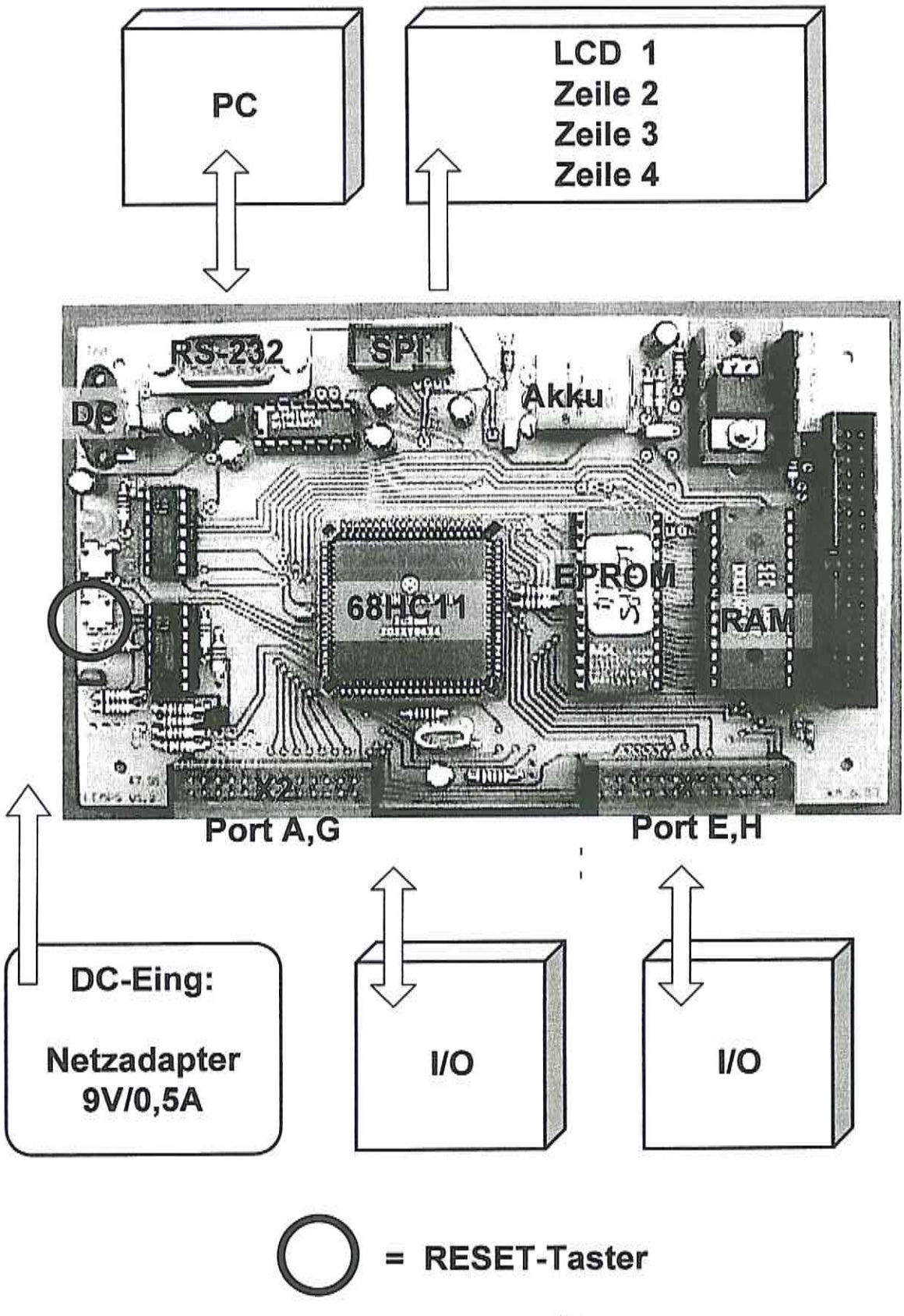


Licht-Timer in digitaler Technik

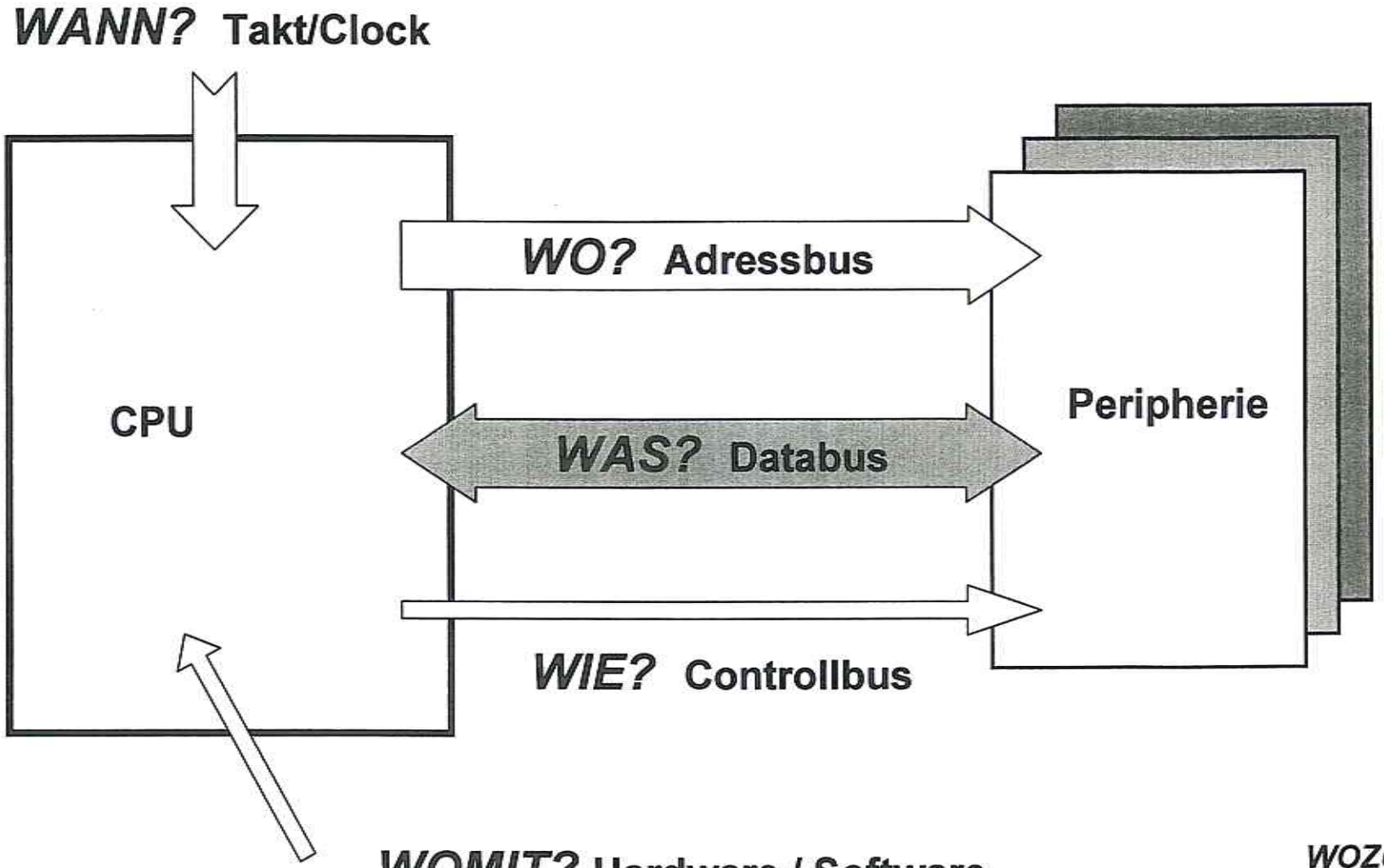


Wie könnte der Timer mit dem Mikrocontroller realisiert werden?

Die Mikrocontroller-Leiterplatte LEMPS

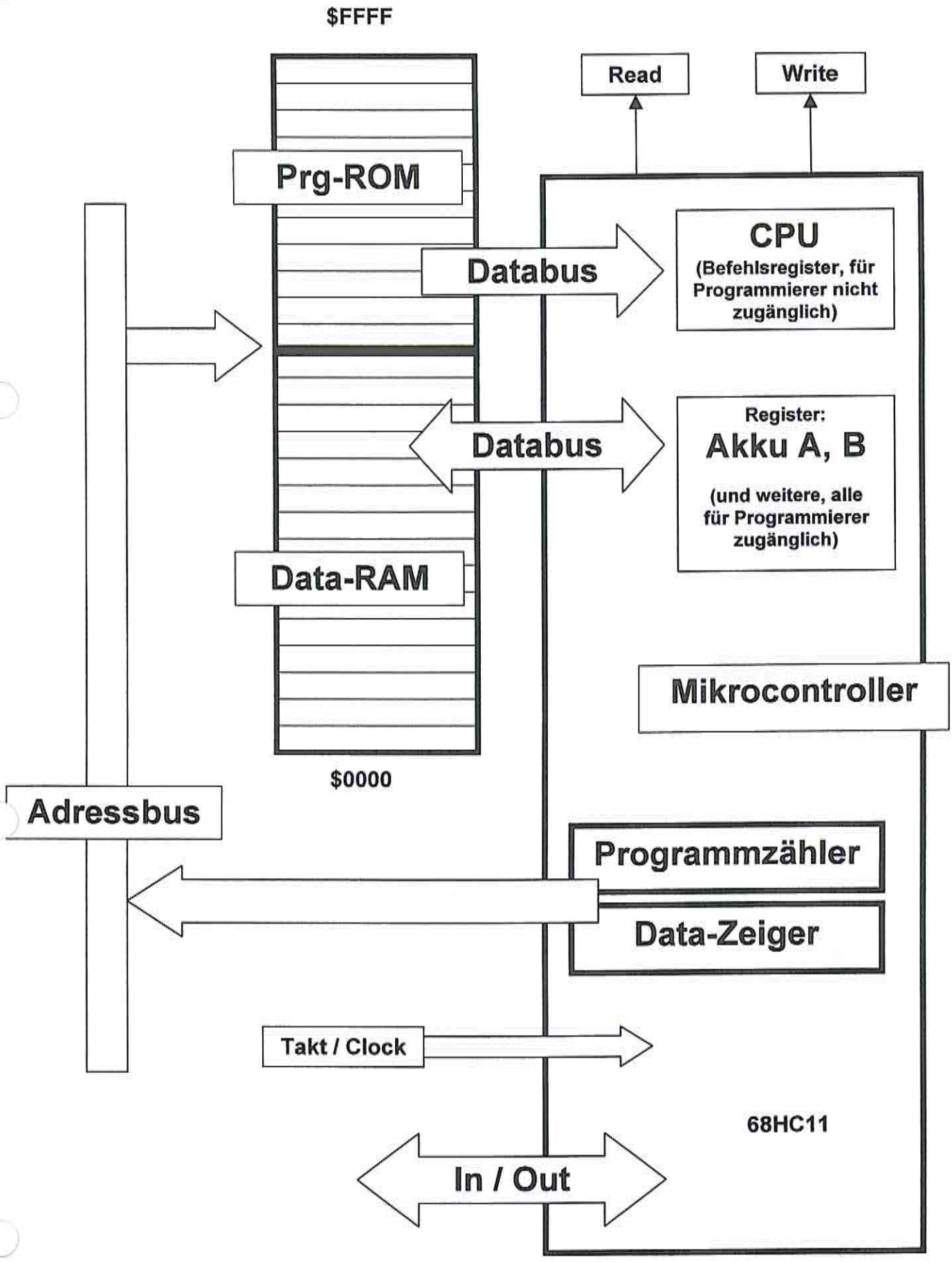


Die Ws des Mikrocontrollers

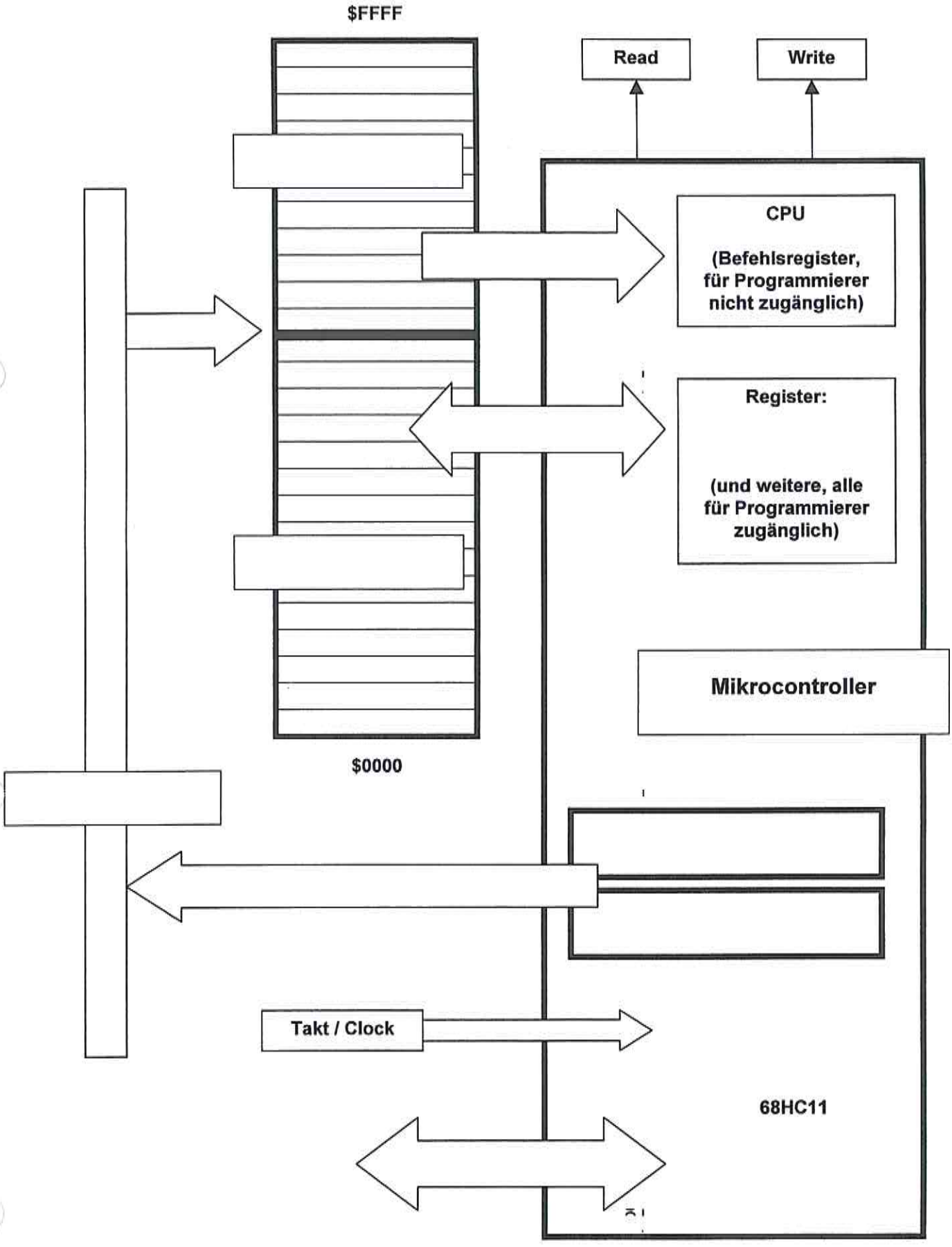


WOZU?
Wer weiss...

Mikrocontroller: Speicher, Register, Programmablauf



Mikrocontroller: Speicher, Register, Programmablauf



Begriffe

Editor, Terminalprogramm, Monitorprogramm, Binärcode, Bit, Byte, Quelltext, Maschinencode, Assembler, Mnemonic, Assemblerdirektiven, Adresslabel, Programm, Programmcode, Befehl, Opcode, Operand, Adresse hh ll, Adressbus, Databus, Controllbus, CPU, Zyklus, Befehls-Decodierung, Akku A, Akku B, Programmzähler, RAM, EPROM, ROM, I/O-Port, Peripheriebaugruppe, Read-/Write-Signal

Was ist ein Mikrocontroller-Programm?

Ein uC-Programm ist eine Liste von Befehlen (Instruktionen), die im Programmspeicher des uC-Systems (ROM, manchmal RAM) als Byte-Folge lokalisiert ist. Jede Instruktion - z.B. 3 Bytes lang - ist eine Anweisung an die CPU des Mikrocontrollers, eine bestimmte Operation auszuführen. Ein Programm kann eine bis zig-tausende Instruktionen enthalten.

Was ist eine Mikrocontroller-Instruktion?

Eine uC-Instruktion besteht aus einem bis fünf Bytes, je nach Instruktionsart. Das erste Byte (bei einigen Befehlen auch das zweite) enthält den Befehlscode (Operationscode, Opcode). Die weiteren Bytes, falls vorhanden, enthalten die Operanden der Instruktion.

10 Beispiele von HC11-Instruktionen (von über 300 möglichen Instruktionen):

Operation	*Adressierung*	*Kurzform Mnemonic*
Store (speichere, schreibe) Akku A (8 Bit)	nach der Speicherstelle mit der Adresse hh ll (= opr)	STAA (opr)
Load (lade, lese) Akku B (8 Bit)	mit dem Inhalt der Speicherstelle mit der Adresse hh ll (= opr)	LDAB (opr)
Set Bit(s), setze Bit(s) auf log. 1, das oder die	mit log. 1 im Mask-Byte (= msk) bezeichnet sind, an der Speicherstelle mit der Adresse 00 dd (= opr)	BSET (opr) (msk)
Store Register D (16 Bit)	nach der Speicherstelle mit der Adresse hh ll (= opr) und hh ll + 1	STD (opr)
Shift (schiebe) Bits in Akku A	um eine Stelle nach links	LSLA
Branch (verzweige) im Programm	nach der Prog-Speicherstelle Adr + rr (= rel), falls das Resultat der vorherigen Operation gleich Null war	BEQ (rel)
Clear (lösche) Akku A	alle Bits in Akku A auf log. 0 rücksetzen	CLRA
Logisch OR Akku A	mit dem Inhalt der Speicherstelle mit der Adresse hh ll (= opr) und überschreibe Akku A mit dem Resultat	ORAA (opr)
Add (addiere) zu Akku B	die Zahl ii (1 Byte = opr) und überschreibe Akku B mit dem Resultat	ADDB (opr)
Springe zum Unterprogramm	dessen Start bei der Prog-Speicherstelle mit der Adresse hh ll (= opr) ist	JSR (opr)

- hh ll = 16-Bit-Adresse einer Speicherstelle, die sich im externen oder internen RAM, im EPROM oder im I/O- und Registerblock-Bereich befinden kann.
- dd = 8-Bit-Direkt-Adresse (Speicherbereich $0000 bis $00FF, das High-Address-Byte wird von der CPU als $00 angenommen).
- rr = 8 Bit relative Adresse, die eine Sprungdistanz ab der momentanen Programm-Adresse bezeichnet.

Wie wird ein Mikrocontroller-Programm erstellt?

Prammierung mit Hilfe der Assemblersprache (= Assemblerprogrammierung)

Der Quelltext (Sourcetext, Source Code) eines uC-Programms wird üblicherweise mit Hilfe eines Schreibprogramms (Editor) auf dem PC erstellt. Der Quelltext enthält den Ablauf der Befehle (Assemblerbefehle in Kurzform: Mnemonics), sowie Adresslabels, Kommentare und Assemblerdirektiven (dies sind Anweisungen an das Umsetzerprogramm, das im nächsten Schritt in Aktion tritt).
Nach der Eingabe des Quelltextes wird das Umsetzerprogramm (Assembler oder Compiler genannt) gestartet, das den Quelltext in Maschinencode (Maschinensprache, Object Code) umwandelt. Maschinencode ist ausführbarer Binärcode, den die CPU des uC-Zielsystems "verstehen" und abarbeiten (ausführen) kann. Dazu muss der Code vorher mit Hilfe eines Terminalprogrammes über eine Schnittstelle (z.B. RS 232) vom PC in den Programmspeicher (in diesem Fall RAM) des Zielsystems geladen werden. Nach dem Startkommando über das Terminalprogramm ("GOTO ...") wird das Programm durch den Controller abgearbeitet.
Im Mikrocontroller-Zielsystem muss ein Monitorprogramm (Monitor) im EPROM enthalten sein, das die Schnittstelle bedient und Befehle (Kommandos, Tokens), die der Benutzer vom PC-Terminalprogramm her schickt, auswertet.
Der Code im Zielsystem wird getestet und bei Bedarf korrigiert (im Quelltext!) und neu geladen (→ SW-Design, Schulung, Kurs!). Ist der Code fehlerfrei (ohne "Bugs"), kann er in ein EPROM / ROM "gebrannt" (einprogrammiert) werden, das als Programm-Speicher ins Zielsystem eingesetzt wird.

Hochsprachenprogrammierung

Beispiele für Hochsprachen: alle Basic-Dialekte, C, Turbo Pascal etc.
Die Programmerstellung geschieht auf die gleiche Weise wie die Assemblerprogrammierung. Vorteil: Der Programmierer braucht sich (im Normalfall) nicht mehr mit den Eigenheiten des jeweiligen Controller-/Prozessortyps auseinanderzusetzen. Eine Hochsprache ist unabhängig vom Controller-/Prozessortyp.

Zusammenspiel von Hardware und Software

Controllbus

Ein Steuerbus mit den Signalen Read und Write (Byte-lesen-Operation, Byte-schreiben-Operation, immer CPU-bezogen), Reset und Clock sorgt für den korrekten Datentransfer zwischen der CPU und den Peripheriebaugruppen (RAM, EPROM, I/O-Port, Registerblock).

Adressbus

Die CPU legt 2 Adressbytes auf den Adressbus (16 Bit) und selektiert damit (zeigt auf) das momentan erforderliche Byte in einer Peripheriebaugruppe. Adressen sind 16-Bit-"Zeiger" auf Daten- und Programmbytes.

Datenbus

- Byte-lesen-Operation, Read-Signal ist aktiv:
Daten- oder Programmbytes (je nach Instruktionszyklus), werden von der momentan selektierten Peripheriebaugruppe auf den Datenbus gelegt und von der CPU eingelesen.

- Byte-schreiben-Operation, Write-Signal ist aktiv:
Datenbytes, die von der CPU auf den Datenbus gelegt wurden, werden in die momentan selektierte Peripheriebaugruppe eingespeichert (geschrieben).

Zeitlicher Ablauf bei der Befehlsausführung in einem Mikrocontroller

Der Ablauf wird am Beispiel des folgenden Befehls dargestellt:
STAA (EXT), heisst: Speichere (kopiere) den Inhalt von Akku A nach der Speicherstelle mit der Adresse hh ll.
Dies ist ein 3 Bytes langer Befehl.

Ablauf in Kurzform

CPU- und Bus-Aktivitäten beim Abarbeiten des Befehls STAA (EXT):

Zyklus	*16 Bit-Adressbus führt*	*8 Bit-Databus führt*	*Hex*	*Selektierter Speicher*	*Read-/Write-Signal aktiv*
1	PC (Prg-Counter)	Opcode	$B7	Prog	Read
2	PC+1	Operand 1	hh	Prog+1	Read
3	PC+2	Operand 2	ll	Prog+2	Read
4	hh ll	Inhalt von Akku A	data	Data, I/O, Regs	Write

Ablauf ausführlich

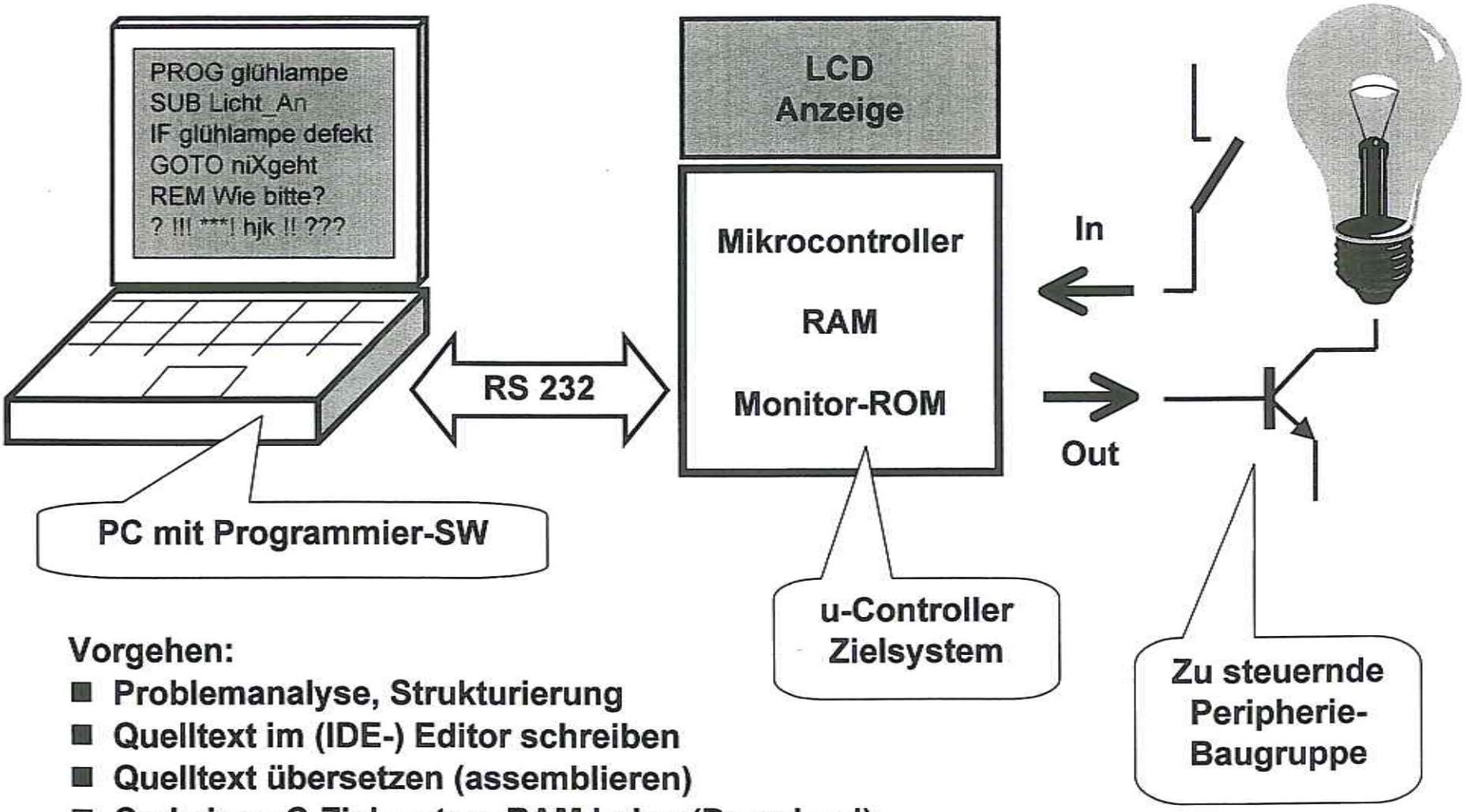
CPU-Aktivitäten

- Erstes Befehlsbyte im Programm-Speicher lesen, auf das der ProgramCounter (PC) gerade zeigt. Genauer: Den PC-Stand auf den Adressbus ausgeben, das Read-Signal aktivieren und das erste Byte zum Decodieren ins CPU-Befehlsregister einlesen.
- Opcode-Byte decodieren: Welcher Befehl liegt vor? (Hier: **STAA**, siehe oben)
- Increment PC (Programmzähler um 1 erhöhen = PC+1)
- Zweites Befehlsbyte lesen: es findet sich an der Prg-Speicherstelle PC+1 und stellt das hochwertige Adressbyte (hh) dar, das zu diesem Befehl gehört.
- Increment PC (Programmzähler um 1 erhöhen = PC+2)
- Drittes Befehlsbyte lesen bei PC+2, dieses stellt das niederwertige Adressbyte (ll) dar, das zu diesem Befehl gehört.
- Adresse hh ll auf den Adressbus legen, Inhalt von Akku A auf den Datenbus legen und den Schreibbefehl (Write-Signal) aktivieren. Das Byte auf dem Datenbus wird jetzt in die Speicherstelle mit der Adresse hh ll eingespeichert (geschrieben).
- Befehl ist fertig ausgeführt.
- Kleine Kaffeepause ...
- Increment PC (Programmzähler um 1 erhöhen = PC+3)
- Weiter geht's mit dem nächsten Befehl (wieder erstes Opcode-Byte bei PC+3 abholen etc.)

System-RESET, Speisung einschalten (Power Up)

Nach dem Einschalten der System-Speisung oder einem System-RESET des Mikrocontrollers enthält der Programmzähler keinen gültigen Wert und wird daher von der CPU zuerst initialisiert. Dabei werden dem Programmzähler die Werte zugewiesen, die in den Speicherstellen $FFFE und $FFFF deponiert sind. Die CPU beginnt also mit der Programmausführung bei der Programm-Speicheradresse, die unter $FFFE und $FFFF abgelegt ist (Restart-Vector, Start-Zeiger).

Was braucht es um ein uC-System zu programmieren und zu betreiben?



Vorgehen:

- Problemanalyse, Strukturierung
- Quelltext im (IDE-) Editor schreiben
- Quelltext übersetzen (assemblieren)
- Code ins uC-Zielsystem-RAM laden (Download)
- Ausführen des Code im uC-Zielsystem

Merkblatt zum EM-Praktikum

Praktikumsmaterial

Für die Dauer des Kurses steht Ihnen (zum Teil in Zweiergruppen) wertvolles Praktikumsmaterial zur Benutzung während des Kurses und zu Hause zur Verfügung.
Dessen Handhabung verlangt einige Beachtung von Sicherheits- und Sorgfaltsmassnahmen damit die Lebensdauer des Materials optimal wird.

Wir bitten Sie folgende Punkte zu beachten:

Die PCs

stehen Ihnen gebrauchsfertig konfiguriert zur Verfügung.

- Wir bitten Sie, **keine Software-Installationen und Setup-Aenderungen** darauf vorzunehmen.
- Die PCs dürfen nur für die im Kurs vorgesehenen Anwendungen genutzt werden.

- Benutzen Sie ausschliesslich die für den Kurs bestimmten Peripheriegeräte.
- Die Notebook-PCs nur im vollkommen ausgeschalteten Zustand in der Schutztasche transportieren.

- Vermeiden Sie Erschütterungen des Notebook bei laufender Festplatte.
- LCD-Bildschirme sind druck- und daher berührungsempfindlich.

- Bitte keinen Kaffee (u. ä.) in die Tastatur giessen, trinken Sie ihn, Sie haben mehr davon.
- PC Texas Instruments Extensa 650CDT:
  Lesen Sie bitte die Benutzungshinweise in der TI-Animation unter
  / Windows-Startmenü / Programme / Notebook Extensa Info /.

LEMPS-Mikrocontroller-Lernsystem

- Mikroelektronik ist grundsätzlich **immer ESD-gefährdet:** elektrostatische Entladungen können die Bauteile zerstören.

ESD-Schutzregeln

- **Die Leiterplatten-Module müssen immer - auch während der Arbeiten im Praktikum - im Koffer bleiben.**
- **Alle beteiligten Personen, Komponenten, Messgeräte und Werkzeuge sollten beim Arbeiten mit dem System immer gleiches elektrisches Potential aufweisen (0 Volt/GND/Erde).**
- **Berühren Sie keine Komponenten und deren Anschlüsse auf der Leiterplatte.**
- **Berühren Sie immer kurz die Alu-Halterung der Leiterplatte im Koffer, bevor Sie den RESET-Taster betätigen, Steckverbindungen auf der Leiterplatte vornehmen oder Messpunkte kontaktieren.**
- **Führen Sie die Messübungen erst nach einer Instruktion durch den Fachlehrer durch.**

Allfällige Kosten für die Instandstellung von Praktikumsmaterial das unsachgemäss behandelt wurde, werden dem verantwortlichen Teilnehmer verrechnet.

sfb/ABW Praktikumsmaterial, EM-Kurs

Einverständniserklärung

Bitte diese Seite ausfüllen, unterschreiben und Ihrem Fachlehrer zurückgeben.

Folgendes Material habe ich zur Benutzung für die Dauer des EM-Kurses erhalten und bin mit den Sicherheits- und Sorgfaltsmassnahmen vertraut und damit einverstanden.

Erhalten	*Stück*	*Bezeichnung*
	1	Notebook-PC TI Extensa 650CDT (o. ähnliches Modell) inkl. Netzgerät
	1	Praktikumskoffer enthaltend:
	1	LEMPS Mikrocontroller-Lernsystem inkl. LC-Display, I/O-Modul
	1	Netzgerät zu LEMPS
	1	RS 232-Kabel
	1	Universaltester, Pen-Format mit Voltmeter und Logiksonde

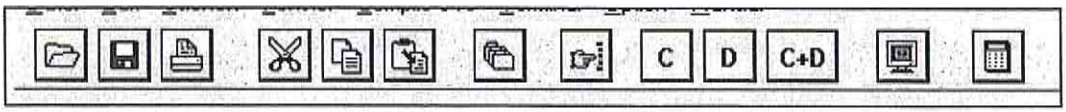
Serie-Nummer des PC:

.....

Ort, Datum, Unterschrift:

.....

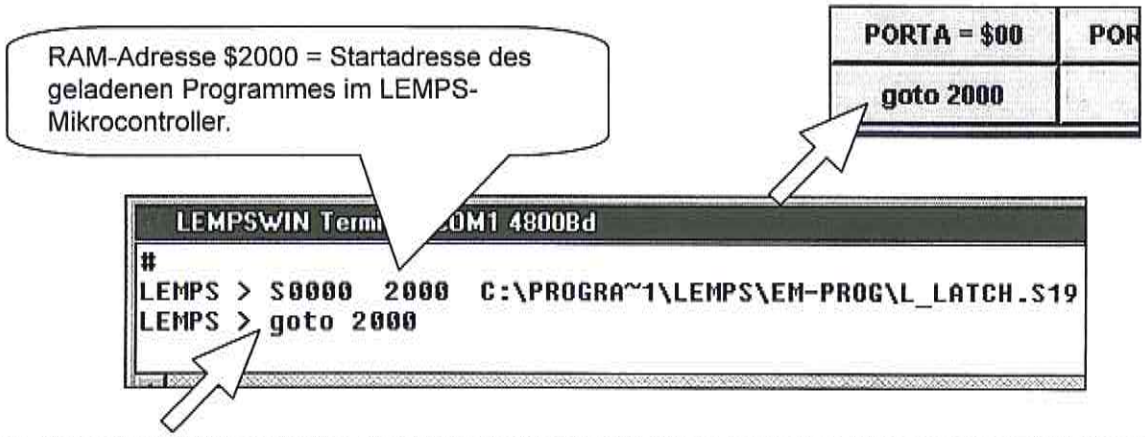
LEMPS Kurzanleitung



Assemblieren/compilieren, laden, starten

Gehen Sie bitte gemäss den folgenden Schritten vor, um eine ASC-Datei (Programmquelltext im Editor der LEMPS-Umgebung) zu assemblieren/compilieren, laden, starten.

- LEMPS-Hardware vorbereiten: Netzgerät anschliessen, PC/Notebook-Verbindung mit RS 232-Kabel zum LEMPS herstellen, PC hochfahren.
- PC-LEMPSwin-Umgebung starten (Windows-Startmenü)
- Gewünschte ASC-Datei (Quelltext) öffnen, Knopf **(1)**
- LEMPS-Reset-Taster betätigen
- C+D -Knopf **(3)** klicken (Compile und Download). Jetzt sollte das Fenster des Terminalprogrammes erscheinen:



- Das übersetzte Mikrocontroller-Programm befindet sich nun in Form von Binärcode im RAM-Speicher des LEMPS und wartet auf den Start-Befehl:
Im Terminalprogramm Token "GOTO 2000" übermitteln, Abkürzung: Knopf GOTO 2000 klicken.

Bei anderen Start-Adressen muss der entsprechende Wert an Stelle "2000" stehen.

LEMPS-Originaldateien und LEMPS-Konfiguration wiederherstellen

Werden beim Arbeiten mit LEMPS versehentlich LEMPS-Original-Quelltextdateien verändert und gespeichert (ein Assembliervorgang speichert ohne Vorwarnung die geöffnete ASC- und zugehörige INC-Dateien ab), können sie wiederhergestellt werden mit: / Windows-Startmenü / LEMPS neu installieren /.
Dieser Vorgang stellt auch die LEMPSwin-Standard-Konfiguration wieder her.

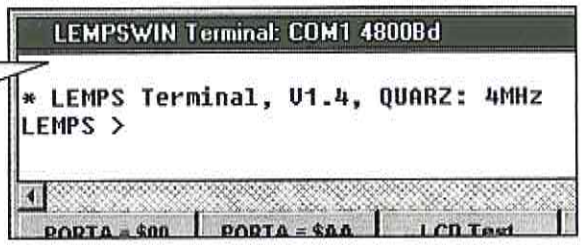
RS 232-Uebertragungstest

Mit den folgenden Schritten können der LEMPS-System-RESET und die RS 232-Verbindung auf fehlerfreie Funktion überprüft werden. Vorgehen wie folgt:

- Terminalprogramm öffnen, Knopf **(4)**
- LEMPS-Reset-Taster betätigen, LEMPS sollte mit folgender Rückmeldung antworten:

Rückmeldung des LEMPS nach dem Betätigen des RESET-Tasters:

Der Datenaustausch zwischen LEMPS und dem PC funktioniert einwandfrei.



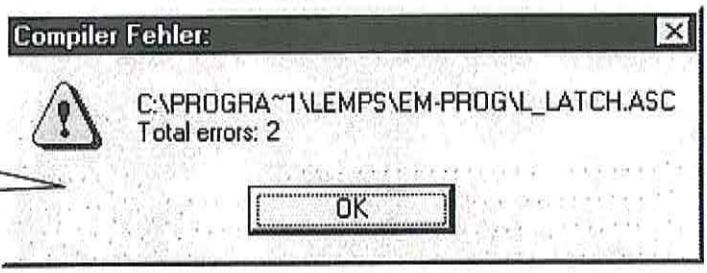
Mögliche Uebertragungs-, und andere Fehler

Folgende Fehlermeldungen im Umgang mit LEMPS, können unter Umständen Ihr Selbstvertrauen strapazieren:

Keine Panik! Die LEMPSwin-Konfiguration muss angepasst werden:

(Für EM-Kurs, TI-Notebook:)
LEMPSwin / Option / System-Einstellungen / 4800 Baud und COM1-Port anwählen /.

Möglicherweise haben Sie vor dem Programm-Download den LEMPS-Reset-Taster nicht betätigt oder das RS 232-Kabel nicht eingesteckt.



! ?

Quelltext-Fehler (ASC-Datei), die eine Fehlermeldung des Assembler/Compilers zur Folge haben, können angezeigt werden mittels Knopf **(2)**.

Mögliche Ursachen für Compiler-Fehler:

- Benötigte Include-Dateien (INC) fehlen: INC-Dateien sollten im LEMPS\SYSTEM\-Verzeichnis abgelegt sein.
- Syntax-Fehler im Quelltext, siehe Online Manual.

Einführung ins LEMPS-Lernsystem, Uebungen

Begriffe
LEMPSwin-Umgebung, Editor, Terminalprogramm, Quelltext, Monitor, Token, Prozessorregister, ASCII-Zeichen, Terminalmodus, assemblieren, Motorola Assembler, Disassembler, Hexzahlen, RS 232-Schnittstelle

Aufgabe: LEMPS-System-Einstieg

Arbeiten Sie die folgende "Blitz-Anleitung" durch und versuchen Sie, dabei auftauchende Fragen und Probleme mit Hilfe der Kursunterlagen (zu Hause) und mit Hilfe des Fachlehrers (im Kurs) zu klären.
Ziel: In kurzer Zeit einen Einblick ins Mikrocontrollersystem LEMPS zu erhalten.

Anleitung zur Aufgabe:

Blitzeinstieg in LEMPS und LEMPSwin

Im folgenden Schnelleinstieg in die Benutzung von LEMPS werden Sie einige **Tokens** (nachfolgend auch Befehle oder Terminalbefehle genannt) anwenden, um LEMPS zu konfigurieren, programmieren und zu überwachen. Benutzen Sie notfalls auch die LEMPS-Kurzanleitung ab Seite 5.3.
Sie werden

- LEMPS in Betrieb nehmen
- sich die Register der CPU anzeigen lassen
- im Terminalmodus ein kleines Programm eingeben (Code in hexadezimaler Form ins RAM schreiben)
- Programmcode im RAM disassemblieren und als Mnemonic-Liste zurückbekommen
- sich die Tokenliste (als Hilfe) ansehen
- das oben eingegebene Programm als fertig vorbereitete Quelltextdatei (ASC-Datei) öffnen, assemblieren, laden und starten.

LEMPS in Betrieb nehmen

- LEMPS-Hardware vorbereiten: Netzgerät anschliessen, PC/Notebook-Verbindung mit RS 232-Kabel zum LEMPS herstellen, PC hochfahren.
- PC-LEMPSwin-Umgebung starten (Windows-Startmenü)
- Terminalprogramm öffnen (Knopf **(4)** in der Kurzanleitung)
- Fenstergrössen ändern, bis beide Fenster auf dem Bildschirm sichtbar sind.
- LEMPS-Reset-Taster betätigen, LEMPS-Rückmeldung:

```
* LEMPS Terminal Vxx, QUARZ: xMHz
LEMPS>
```

Die Register der CPU anzeigen lassen

Mit dem Befehl "regi" können Sie den Inhalt der Prozessorregister anschauen. Geben Sie auf der Kommandozeile des Terminalprogrammes den Befehl "regi" ein und drücken Sie die Enter-Taste:

```
LEMPS> regi
```

Im Terminalmodus ein kleines Programm eingeben (Code in hexadezimaler Form ins RAM schreiben)

Mit dem Befehl "write" können Sie Daten an eine beliebige RAM-Adresse schreiben. Mit dem folgenden Beispiel schreiben Sie Ihr erstes Programm:

```
LEMPS> write 0000 86 21 BD 14 0C BD 14 4B 4C 7E 00 02
```

Schliessen Sie diese Zeile mit der Enter-Taste ab. Das Programm wird dann, beginnend bei der Adresse $0000, im Speicher abgelegt.

Mit dem Befehl "read" können Sie nun überprüfen, ob dieses Programm wirklich im Speicher ist:

```
LEMPS> read 0000 0008
```

Sie erhalten nun die folgende Bestätigung:

```
0000 : 86 21 BD 14 0C BD 14 4B 4C 7E 00 02 .. .. ..
```

Dieses kleine Programm mit dem Namen "ASCDump" gibt im Sekundentakt ASCII-Zeichen beginnend mit Charakter #33 an den PC zurück. Starten Sie das Programm vom LEMPS-Terminal aus mit dem "goto" Befehl wie folgt:

```
LEMPS> goto 0000
```

Mit dem RESET-Taster auf dem LEMPS-Print unterbrechen Sie das Programm ASCDump und kehren in den Terminalmodus zurück.

Programmcode im RAM disassemblieren und als Mnemonic-Liste zurückbekommen

Die oben ins RAM eingegebenen Hexzahlen für das Programm ASCDump werden Ihnen nicht viel sagen. Sie können jedoch den jetzt im RAM gespeicherten Programmcode mit Hilfe des Disassemblers (ein weiteres Hilfsprogramm, resp. Token im LEMPS-Monitor) in die Assemblersprache (Mnemonics) zurückübersetzen und ausgeben lassen (vom RAM zum PC).
Geben Sie dazu den folgenden Befehl im LEMPS-Terminal ein (schliessen Sie wie immer mit Enter ab):

```
LEMPS> dis 0000
```

Die Tokenliste (als Hilfe) ansehen

Falls Sie einmal einen dieser Befehle, resp. Tokens vergessen sollten, geben Sie im LEMPS-Terminal den folgenden Befehl ein:

```
LEMPS> help
```

Der Prozessor gibt Ihnen auf diese Aufforderung eine Liste mit den wichtigsten implementierten Terminalbefehlen, resp. Tokens aus. Eine Liste dieser Befehle finden Sie auch im Kurs-Anhang.

Das oben eingegebene Programm "ASCDump" als fertig vorbereitete Quelltextdatei (ASC-Datei) öffnen, assemblieren, laden und starten

Vorab:

- Gewünschte ASC-Datei (Quelltext) ASCDUMP.ASC öffnen, (Knopf **(1)** in der Kurzanleitung)

Dies ist das Ihnen jetzt bekannte vollständige Programm ASCDump, welches die ASCII-Zeichen vom LEMPS an den PC sendet.
In der vorliegenden Quelltextdatei sehen Sie die Auflistung aller, für dieses Programm benötigten Mnemonics (Assemblerbefehle), Adresslabels, Kommentare und Assemblerdirektiven. In dieser Form ist der Quelltext bereit, um assembliert und anschliessend ins LEMPS-RAM geladen ("download") zu werden. Diese Arbeitsschritte, die Sie vorhin manuell durchgeführt haben (Hex-Zahlen-Eingabe ins RAM) können Sie mit den nachfolgenden Schritten einfach und mühelos erledigen. Gehen Sie dazu wie folgt vor:

- LEMPS-Reset-Taster betätigen
- Bringen Sie das Editor-Fenster ASCDUMP.ASC in den Vordergrund (F6 oder NächstesDokument-Knopf)
- C+D -Knopf **(3)** klicken (Compile und Download)
- Programm starten über die Kommandozeile des Terminalprogrammes:

```
LEMPS> goto 0200
```

Der Quelltext wird nun assembliert, als Maschinencode ins LEMPS-RAM geladen und da von der CPU ausgeführt.

Die Quelltextdatei kann nun selbstverständlich jederzeit abgeändert, neu assembliert und heruntergeladen werden. Auf diese Weise werden Programmanpassungen und -änderungen realisiert, Fehler behoben ("debuggen") und Programmparameter geändert (Parametrierung).

Mit diesen Schritten haben Sie sich bereits wichtiges Grundwissen betreffend der Handhabung eines Mikrocontrollers und dessen Programmumgebung angeeignet!

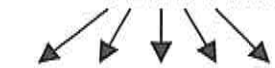
Zahlensysteme

Allgemeines

Im täglichen Leben benützen wir unsere Zahlenschreibweise meist, ohne uns Gedanken darüber zu machen, welch lange kulturelle Entwicklung nötig war, um zu unserer gewöhnlichen Dezimalschreibweise zu kommen.
Die einfachsten Zahlendarstellungen traten auf Kerbhölzern auf, die besonders zum Notieren von Schulden üblich waren – daher die Redewendung: „Er hat etwas auf dem Kerbholz." Die heute noch beim Jassen verwendete Strichmethode lehnt sich an ein altes Zählsystem an. Das bekannteste Beispiel für ein Zahlen- und Additionssystem ist die römische Zahlenschreibweise. Unser heutiges Dezimalsystem, eine Verknüpfung von Stellenwertschreibweise mit dem alten Zehnersystem, entstand jedoch in Indien (etwa im 6. Jahrhundert).

Beispiel additives Zahlensystem:
Bei einem additiven Zahlensystem, wie es die Römer verwendet haben, hatte jedes vorkommende Zahlensymbol, ungeachtet der Position, seinen fest zugeordneten Wert. Die ganze dargestellte Zahl ergibt sich dann durch Addition der Einzelwerte (additives System):

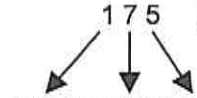
C L X X V bedeutet:



100 + 50 + 10 + 10 + 5 = 175

Beispiel Stellenwertsystem:
An die Stelle der additiven Verbindung der Zahlzeichen tritt die multiplikative Ordnung. Die darzustellende Zahl ist nicht nur von der Grösse der einzelnen verwendeten Zahlensymbole abhängig, sondern in noch viel grösserem Masse von deren Position innerhalb des Systems.

1 7 5 bedeutet im Dezimalsystem:



1*100 + 7*10 + 5 = 175

Begriffe eines Zahlensystems

Dezimal Basis 10	**Dual** Basis 2	**Oktal** Basis 8	**Hexadezimal** Basis 16
0	0	0	0
1	1	1	1
2		2	2
3		3	3
4		4	4
5		5	5
6		6	6
7		7	7
8			8
9			9
			A
			B
			C
			D
			E
			F

dezi = zehn
dual = zwei
oktal = acht
hexadezimal = sechs-zehn

Ziffernvorrat:
Die Anzahl der zur Verfügung stehenden Ziffern.

Stellenwert:
Der Wert einer Ziffer hängt von der Stelle ab, an der sie innerhalb einer Zahl steht.

Basis oder Grundzahl:
Sie ist die mathematische Grundlage eines Zahlensystems.
- Der Ziffernvorrat entspricht der Basis
- Die Stellenwerte sind die Potenzwerte der Basis

Dezimalsystem

Der Umgang mit Zahlen im täglichen Leben beschränkt sich auf das dezimale Zahlensystem, wie es aus der Schulzeit bekannt ist.

Dezimalzahl	1		6		4		
10er-Potenz	10^2		10^1		10^0		
Stellenwert	100		10		1		
Dezimalwert	1*100	+	6*10	+	4*1		$= 164_{10}$

Zur Kennzeichnung der Basis wird die Zahl 10 oder der Buchstabe D verwendet.

Beispiele für die Schreibweise: $164 = 164_{10} = 164_D = 164\ D$

Duales Zahlensystem (binäres Zahlensystem)

Trotz der enormen Leistungsfähigkeit des Dezimalsystems gibt es in der Digitalen Schaltungstechnik Problemstellungen, die nur durch den Einsatz binärer Zahlensysteme sinnvoll gelöst werden können.

Dualzahl	1		1		0		1		
2er-Potenz	2^3		2^2		2^1		2^0		
Stellenwert	8		4		2		1		
Dezimalwert	1*8	+	1*4	+	0*2	+	1*1		$= 13_{10}$

Zur Kennzeichnung der Basis wird die Zahl 2 oder der Buchstabe B verwendet.

Beispiele für die Schreibweise: $1101_2 = 1101_B = 1010\ B$ (%1101 für HC11-Assembler)

Begriffe

Ein Bit (binary digit) ist die kleinste Informationseinheit in der Datenverarbeitung, dargestellt mit 0 oder 1.
Ein Byte besteht aus 8 Bits. Es ist die kleinste Informationseinheit zur Darstellung eines codierten Zeichens, z.B. Buchstabe, Ziffer, Sonderzeichen, Befehl.

1 Bit -> 1 oder 0
1 Byte -> 8 Bit (z.B. 10011000)
1 kB -> 1 KiloByte = 2^{10} Byte = 1024 Byte
1 MB -> 1 MegaByte = 2^{10} kB = 1024 kByte

Hexadezimales Zahlensystem

Die Darstellung von grossen Dezimalzahlen im dualen Zahlensystem ist unübersichtlich und fehlerträchtig. Deshalb hat man einzelne Bits zusammengefasst und auf der Basis des Darstellungsbereichs dieser Bitgruppe ein neues Zahlensystem aufgebaut.
Als sinnvolle Teilung zeigt sich die Zusammenfassung von 4 Bit des Dualsystems. Durch die Darstellungsmöglichkeit von nunmehr 16 Zuständen ($2^4 = 16$), spricht man hier vom Hexadezimalsystem (10 Ziffern von 0 bis 9 und 6 Buchstaben von A bis F), das in der Rechnertechnik nicht mehr wegzudenken ist.

Hexadezimalzahl	1		B		4		
16er-Potenz	16^2		16^1		16^0		
Stellenwert	256		16		1		
Dezimalwert	1*256	+	11*16	+	4*1		$= 436_{10}$

Zur Kennzeichnung der Basis wird die Zahl 16 oder der Buchstabe H verwendet.

Beispiele für die Schreibweise: $1B4_{16} = 1B4_H = 1B4\ H$ ($1B4 für HC11-Assembler)

Umrechnung der Zahlensysteme

Die Umrechnung von einem Zahlensystem in ein anderes Zahlensystem kann zum Beispiel mit einem Taschenrechner, dem Bordrechner von LEMBSwin oder mit dem wissenschaftlichen Rechner im Windows als Hilfsmittel erfolgen.

Tabelle für die Zahlenwerte von 0 bis 15:

Dezimal	Dual	Hexadezimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Übungen

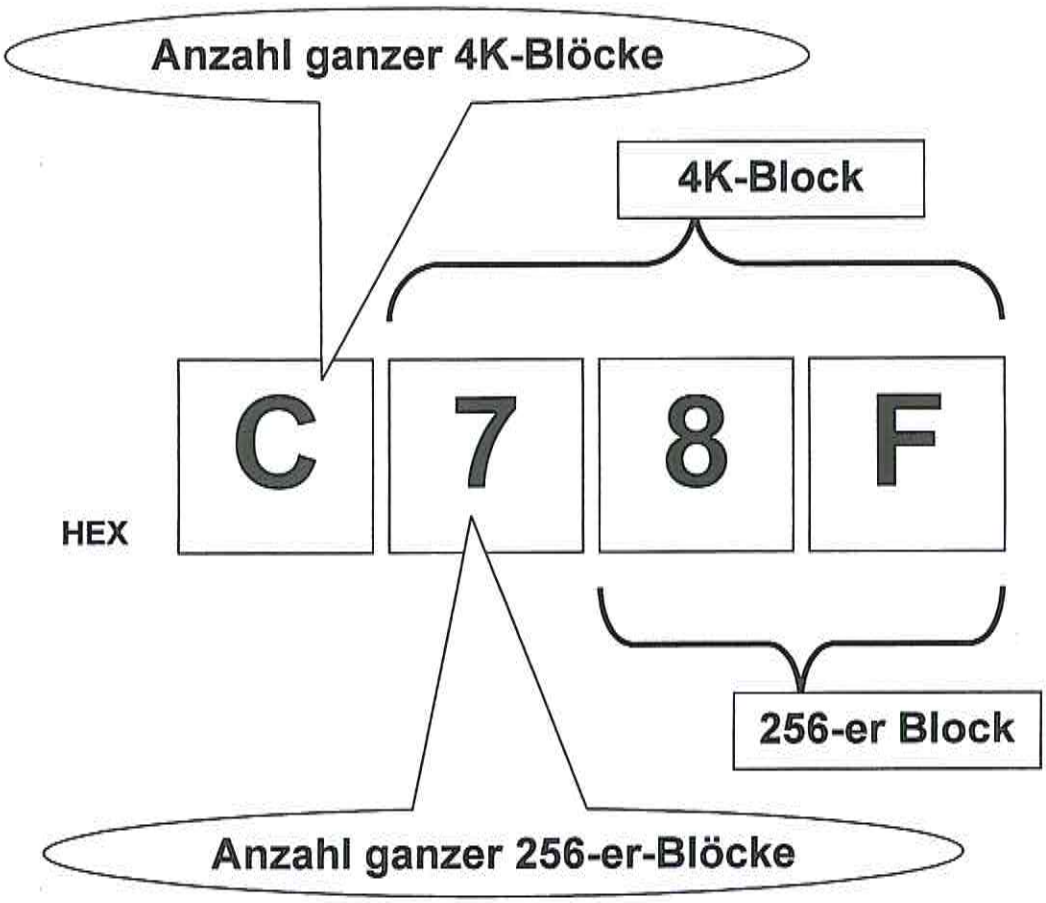
Aufgabe 1:
Ergänzen Sie die folgende Tabelle

Dezimal	Dual	Hexadezimal
129		
		6B
	10001001	
		D1
	11011011	
223		

Aufgabe 2:
Warum ist in der Digitaltechnik der Einsatz vom binären Zahlensystem sinnvoll und wichtig?

Hexadezimalsystem

Beispiel: 4-stellige Zahl



	C	7	8	F
Wert	8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1
BIN	1 1 0 0	0 1 1 1	1 0 0 0	1 1 1 1

Uebungen Zahlensysteme

Benutzen Sie für folgende Uebungen falls nötig den Anhang "Speicheraufteilung LEMPS".

Aufgabe 1: Speichergrösse

Die CPU des HC11-Mikrokontrollers kann max. 64K Speicherstellen (Programm- und Datenspeicher) adressieren (ansprechen).
a) Wie viele Adressbits sind dazu notwendig?
b) Wie viele maximale Speicherstellen sind es exakt (Dezimalzahl)?

a) .....

b) .....

Aufgabe 2: Breite des Adressbusses

Wie viele Adressleitungen (Adressbusbreite) müsste die HC11-CPU aufweisen, damit sie die doppelte Anzahl Speicherstellen adressieren könnte?

.....

Aufgabe 3: RAM-, EPROM -Select

a) Mit welchem Adressbit zeigt die LEMPS-HC11-CPU eindeutig an, ob sie gerade die untere (Adressen $0000 bis $7FFF) oder die obere Speicherhälfte (Adressen $8000 bis $FFFF) anspricht?
b) Welche Adressbereiche (in Hex-Werten) sind den Speichern RAM und EPROM zugewiesen?

a) .....

b) .....

Aufgabe 4: Speicherblock-Adressen

Der Anhang "Speicheraufteilung LEMPS" zeigt die Aufteilung des 64K-Speicherraumes in 4K-Blöcke. Wie lauten die Anfangs-Adressen der 1K-Blöcke im Raum von $0000 bis $1FFF in Hex und Dezimal?

1 1K-Block-Nr	**2**	**3**	**4**	**5**	**6**	**7**	**8** .....
Anf-Adresse Hex **$0000**	**$0400**	.....	.....	.....	.....	.....	.....
Anf-Adresse Dez **0**	**1024**	.....	.....	.....	.....	.....	.....

Aufgabe 5: Bitmuster ausgeben

Am LEMPS-Output-Port A können 8 LEDs angesteuert werden.
a) Welches Bitmuster muss nach Port A geschrieben werden, um die Hex-Zahl $AA mit den 8 LEDs binär darzustellen?
b) Wie lautet der Dezimalwert von $AA?

a) .....

b) .....

Aufgabe 6: Zahlenformat

Die Dezimalzahl 67D kann mit Hilfe des LED-Port-A auf folgende Arten dargestellt werden:
a) Binärformat
b) BCD-Format
Wie sehen die zugehörigen Bitmuster aus und wie lauten deren Hex-Werte?

a) .....

b) .....

Aufgabe 7: PC-E/A-Adressen unter Windows 95

Hexadezimale Adressangaben finden sich auch für die PC-Systemeinstellungen unter Windows 95.
a) Bei welcher E/A-Adresse befindet sich die COM1-Schnittstelle Ihres PCs?
b) Adresse der LPT1-Schnittstelle?
Angaben bitte in hexadezimalen und dezimalen Werten. Die Angaben finden sich im Menü
/ Windows-Startmenü / Einstellungen / Systemsteuerung / System / Geräte-Manager /
/ Anschlüsse (COM und LPT) / ... / Ressourcen /
(Bitte verändern Sie keine Einstellungen!)

a) .....

b) .....

Aufgabe 8: 8 Bit Datenregister

Wie viele unterschiedliche Bit-Muster oder Kombinationen können mit dem 8-Bit-LED-Port angezeigt werden?

.....

Codes

Definition: Unter Code versteht man die eindeutige Zuordnung zwischen zwei Sätzen verschiedener Symbole.

Allgemeines:
In der Digitaltechnik arbeitet man mit Signalen, die Informationen in Form binärer Signale aufnehmen und bei Bedarf wieder bereitstellen können. Trotz der Symbolbeschränkung der Binärzeichen von 0 und 1 sind die Codierungsprobleme in der Digitaltechnik nicht ganz leicht überschaubar. Mit den Zeichenelementen 0 und 1 kann für die einzelnen Anforderungen resp. Probleme ein spezieller Code aufgebaut werden.

Beispiele von Anforderungen:
- Rechnen
- Messen
- fehlerfreie Übertragung usw.
- Anzeigen

Beispiele für Codes:
- Das von Samuel Morse entwickelte Morsealphabet

a = • –
b = – • • •
c= – • – •
d = – • •
usw.

- Telex (Serieller Code)
- ASCII (7 Bit)
- Dezimalsystem
- Sprache
- Blindenschrift (Braille Schrift)

Tetradische Codes

Tetradische Codes besitzen als dargestellte Ziffer immer 4 Bit.

Übersichtstabelle:

Dualcode	BCD-Code (Werte)	Aiken-Code (Werte)	Excess-3-Code (Werte)
0000	0	0	xxxxx
0001	1	1	xxxxx
0010	2	2	xxxxx
0011	3	3	0
0100	4	4	1
0101	5	xxxxx	2
0110	6	xxxxx	3
0111	7	xxxxx	4
1000	8	xxxxx	5
1001	9	xxxxx	6
1010	xxxxx	xxxxx	7
1011	xxxxx	5	8
1100	xxxxx	6	9
1101	xxxxx	7	xxxxx
1110	xxxxx	8	xxxxx
1111	xxxxx	9	xxxxx

X: nicht zulässig, pseudo Tetraden

BCD-Code (Binary Coded Decimal)

Der BCD-Code ermöglicht die direkte Verschlüsselung von Dezimalzahlen ohne Umrechnung in das Dualzahlensystem. Mit 4 Bit können 2^4 = 16 Zeichen dargestellt werden, also auch die 10 Ziffern 0 bis 9. Für die Darstellung der Dezimalzahlen benötigt man 4 Bit pro Dezimalstelle.

Bsp: Wandeln Sie die Zahl 438_D in den BCD Code: 438_D =

Aiken-Code (Aiken Am. Erfinder)

Der Aiken-Code ist für elektronische Zähler gut geeignet. Bei der Addition gelten spezielle Sonderregeln.

Bsp: Drücken Sie die Zahl 296_D im Aiken-Code aus: 296_D =

Excess-3-Code

Der Excess-3-Code hat keine Stellenbeschreibung für die Ziffern 0 bis 9. Die Verschiedenen Kombinationen der binären Zustände 1 und 0 sind den Dezimalziffern nur zugeordnet.
Dieser Code ist zum Rechnen durch einfache Korrekturen geeignet.

Bsp: Drücken Sie die Zahl 517_D im Excess-3-Code aus: 517_D =

Verschiedene Codes

Dezimal	1 aus 10 Code	2 aus 5 Code	Gray-Code
0	0000000001	11000	0000
1	0000000010	00011	0001
2	0000000100	00101	0011
3	0000001000	00110	0010
4	0000010000	01001	0110
5	0000100000	01010	0111
6	0001000000	01100	0101
7	0010000000	10001	0100
8	0100000000	10010	1100
9	1000000000	10100	1101

1 aus 10 Code

Die Dezimalziffern werden bei diesem Code in eins aus 10 Bits verschlüsselt.

2 aus 5 Code

Beim 2 aus 5 Code hat ein Codewort immer zwei Bit mit dem Wert 1. Dieser Code ermöglicht eine Prüfung auf Fehler.

Gray-Code

Dieser Code ist einschrittig, d.h. beim Weiterzählen ändert sich immer nur eine Stelle im Codewort.
Anwendung: Codescheiben als Messystem für Längen und Winkelmesssungen. Beim Wechseln von Codestellen entstehen keine Fehlinformationen, da beim Übergang von einer Zahl zur nächsten immer nur 1 Bit ändert.

Code-Wandlung

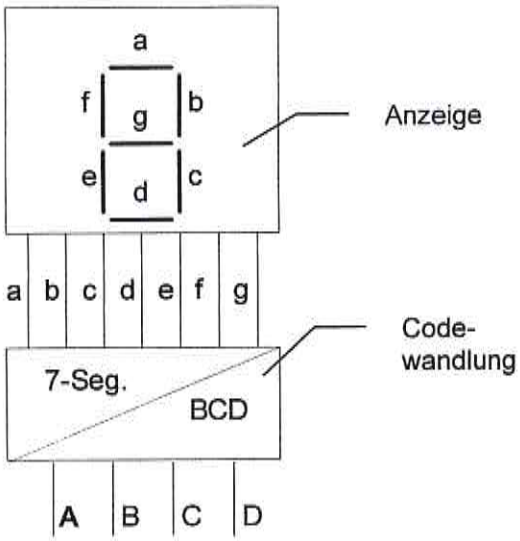
Ein Code-Wandler (Code-Umsetzer) ist eine Schaltung, die einen vorhandenen Datencode in einen anderen Code transferiert. Dies wird vielfach mit kombinatorischer Logik realisiert. Grundlegend ist in der kombinatorischen Digitaltechnik, dass einer Änderung der Eingangsvariablen unmittelbar die Änderung der Ausgangsvariablen folgt.

Vorgehen bei der Entwicklung der Schaltung:

- erstellen der Wahrheitstabelle
- erstellen der Schaltfunktionen
- minimieren der Schaltfunktionen z.B. mit Hilfe der Schaltalgebra oder KV-Diagramm
- realisieren der Schaltung (Schema zeichnen)

Beispiel: Siebensegment-Anzeige

Anzeigeelemente mit sieben ansteuerbaren Teilelementen (Siebensegment-Anzeige) dienen zur Darstellung von Zeichen, meistens Ziffern. In dem vorliegenden Beispiel soll zur Zifferndarstellung des BCD-Codes eine Siebensegment-Anzeige verwendet werden.



Wahrheitstabelle:

BCD-Code				Anz	7-Segment-Code						
D	C	B	A	Dez	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	0	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	0	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	0	0	1	1

Digitaltechnik, Uebersicht

Zur Darstellung der zwei möglichen Zustände in einem digitalen System werden die Zeichen 0 und 1 (Binärzeichen) verwendet. Gerechnet wird im **Dualsystem** oder **Hexadezimalsystem**.
Gelegentlich werden Werte zum besseren Verständnis in "unser" Dezimalsystem umgerechnet.

Theoretische Behandlung der Digitaltechnik
Die Grundverknüpfungen **AND, OR, NOT** können untersucht und zu komplexen Schaltungen kombiniert werden.
Dies kann mit folgenden Hilfsmitteln geschehen:
Schaltalgebra (Boolesche Algebra), Funktion, Wertetabelle, Karnaugh-Diagramm.

Praktische Behandlung der Digitaltechnik
Die praktische Ausführung und Kombination der theoretischen Verknüpfungen, geschieht mit Hilfe von binären (Hardware-) Elementen die als **AND, OR, NOT, NAND, NOR, XOR -Elemente**, sowie als **Flipflops, Zähler, Register, Speicher, Prozessoren** etc. aufgebaut werden.

Logische Werte für die Schaltalgebra

Logischer Wert / Zustand	*Aussage / Zustand*
0 (Null)	Falsch / False
1 (Eins)	Wahr / True

Signalpegel, -formen, physikalisch

Signalpegel	*5V-TTL-Familie*	*5V-CMOS-Familie*	*Allgemein*
L (Low / Lo / 0)	0 bis 0,7 V	0 bis 1,5 V	Phys. Wert niedrig
H (High / Hi / 1)	2,4 V bis 5 V	3,5 V bis 5 V	Phys. Wert hoch

Signalform	*Signal*	*Beispiel*	*Beispiel, Signal-Name*
	**Hi-Flanke**	Clock-Signal	C
	**Lo-Flanke**	Clock-Signal	C*
	**Hi-Puls**	Read-,Write-Signal Monoflop Auswahl (Select)-Signal	RD, WR Q CS
	**Lo-Puls**	Read-,Write-Signal Monoflop Auswahl (Select)-Signal	RD*, WR* Q* CS*

Verschiedene Schreibweisen für invertierte Signale (lies: "signal quer")
$\overline{\text{signal}}$ signal* /signal \signal signal-quer

Logikart, -vereinbarung

Positive Logik: (üblicherweise angewendet)
- Dem logischen Wert 0 ist der Pegel L zugeordnet
- Dem logischen Wert 1 ist der Pegel H zugeordnet

Wollen Sie tiefer in die Thematik eintauchen? Bitte sehr...

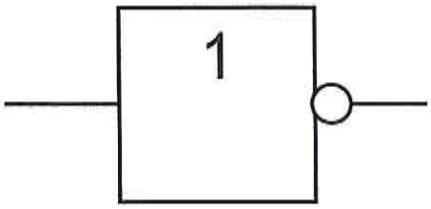
Vertiefung Digitaltechnik

Europalehrmittel, Kapitel Digitaltechnik

Stichworte
- Einführung in die Digitaltechnik, Grundlagen der Schaltalgebra
- **Grundschaltungen,** UND, ODER, NICHT, NAND, NOR, XOR (Antivalenzschaltung)
- Binäre Elemente mit besonderen Ausgängen, **Normalausgang**, Ausgang mit offenem Kollektor, **Tristate-Ausgang**, Busstrukturen
- Digitale Schaltkreisfamilien, **Logikfamilien**
- Binärcode, BCD-Code
- Laufzeiten und Leistungsbedarf von binären Elementen
- Kennzeichnung integrierter Schaltungen
- Binärspeicher, Realisierung eines Flipflops
- **Asynchrone und synchrone Flipflops**
- **Monoflops** und Verzögerungselemente
- **Binärzähler**, Zähler mit T-Flipflops, **Zähler mit IC**
- Synchrone **Schieberegister**, Schieberegister als IC
- **DA**-Umsetzer, **AD**-Umsetzer

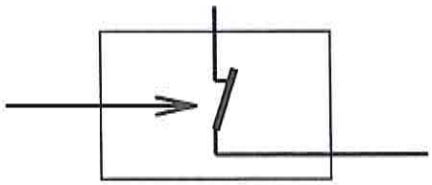
Digitale Grundschaltungen

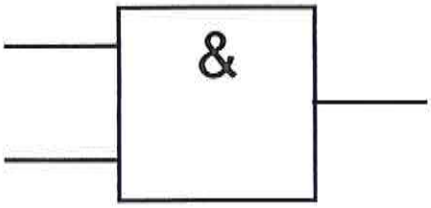
Gezeichnete Schalterstellungen = logischer Wert 0 (Null)



NOT/INV

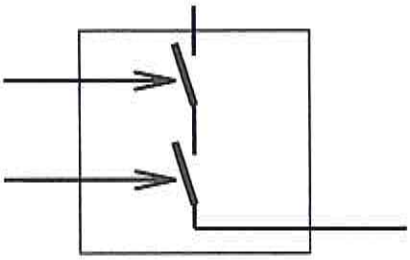
.....

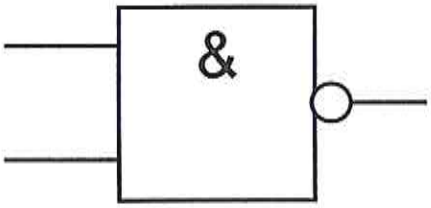




AND

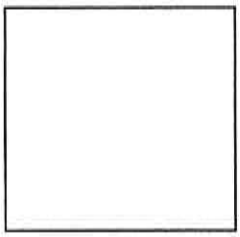
.....

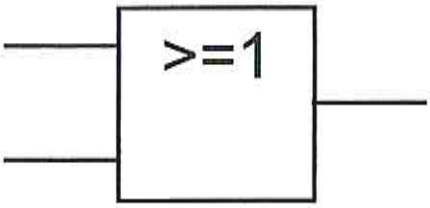




.....

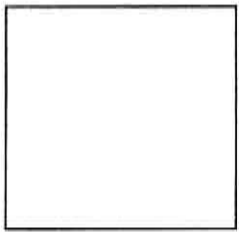
.....

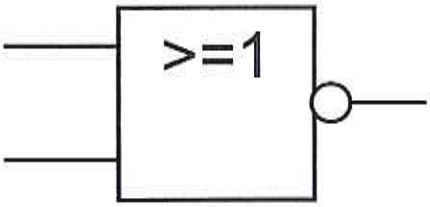




.....

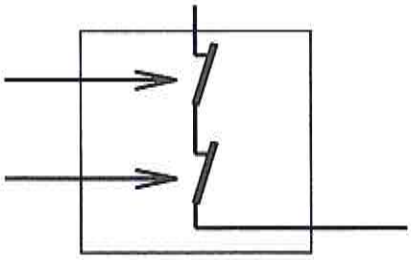
.....

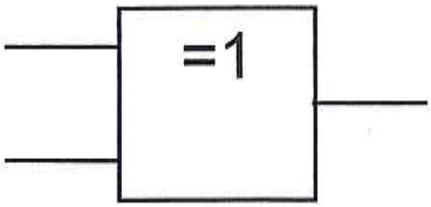




.....

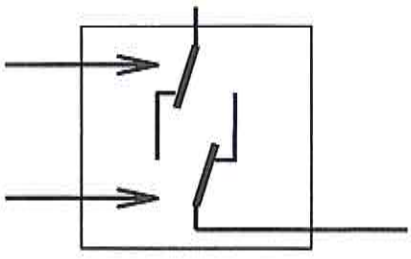
.....





XOR

.....

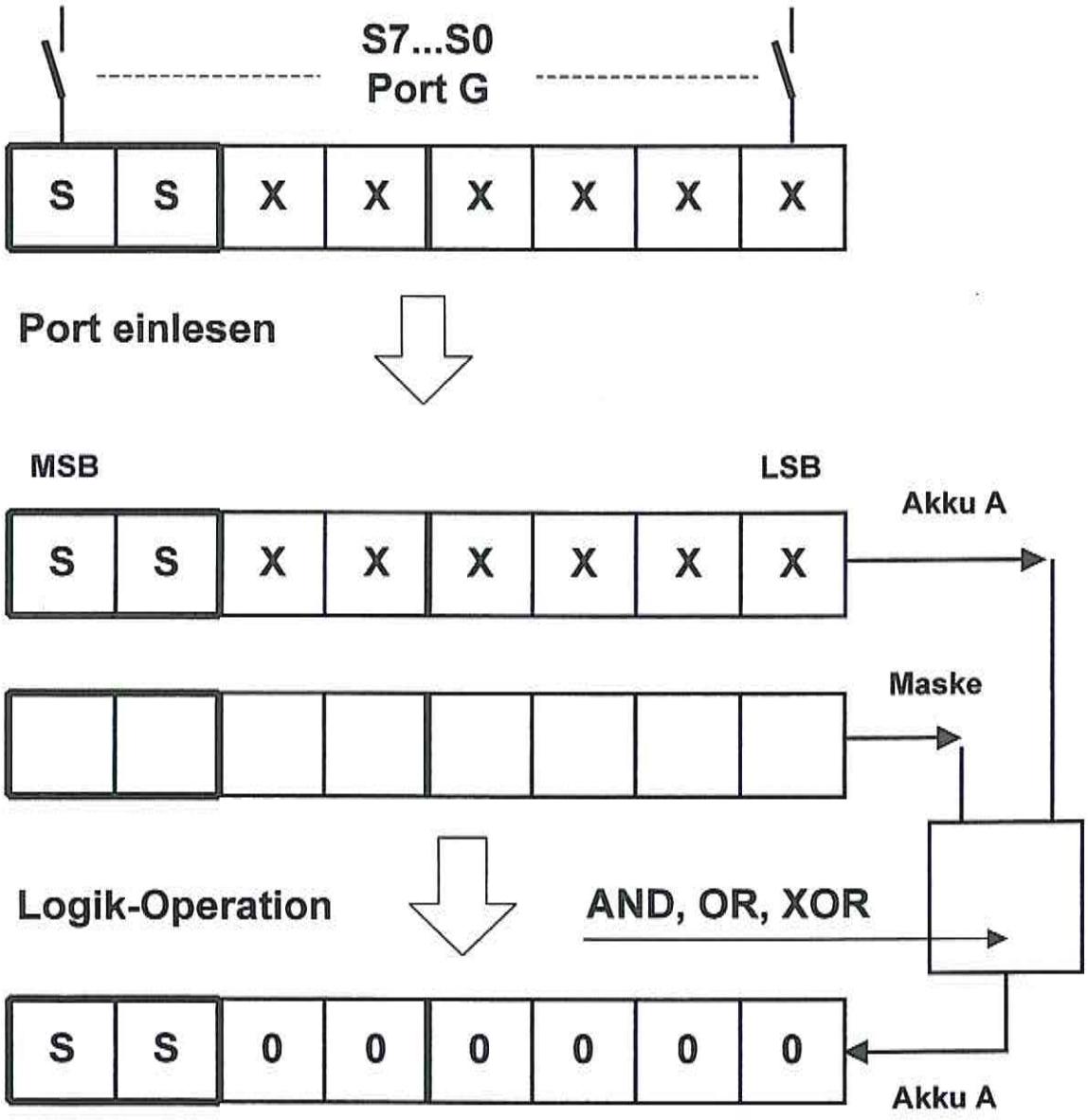


Anwendung der Logikverknüpfungen

Logikoperationen des Mikrocontrollers

Anwendungs-Beispiel: Schalterbits maskieren

Für eine Anwendung werden die Schalter S0 bis S7 von Port G in den Akku A der CPU eingelesen. Benötigt und ausgewertet werden dabei vom Programm nur die Schalter S6 und S7. Gefordert ist, dass die übrigen Schalterbits "X" (nach dem Einlesen in Akku A) immer den Wert 0 (= Low) aufweisen, auch wenn die zugehörigen Schalter betätigt werden.
Die folgende Grafik zeigt, wie der Akku A mit einer **Bitmaske** (Maskierungs-Byte) logisch verknüpft werden kann, um gezielt einzelne Bits zu setzen (log. Wert = 1) oder rückzusetzen (log. Wert = 0). Dabei wird jedes einzelne Bit des Akku A mit dem entsprechenden Bit der Maske mittels der gewünschten Logikoperation verknüpft. Der Inhalt des Akku A wird mit dem Resultat dieser Operation überschrieben.



Uebungen zu obigem Anwendungs-Beispiel

Aufgabe 1:

a) Wie sieht die Maske als binäre und hexadezimale Zahl aus?
b) Mit welcher Logikoperation muss sie mit dem Akku A verknüpft werden?

a) .....

b) .....

Aufgabe 2:

a) Welche Maske und
b) welche Logikoperation muss angewendet werden, falls die nicht ausgewerteten Schalterbits "X" immer den Wert 1 (= High) im Akku A aufweisen müssen?

a) .....

b) .....

Aufgabe 3:

a) Mit welcher Logikoperation und
b) welchem Byte-Operand kann erreicht werden, dass im Akku A für den Schalter S7 immer der inverse Wert des Schalterstandes vorhanden ist (Bit invertieren!)?

a) .....

b) .....

Grundfunktionen Digitaltechnik

Allgemeines

Die im folgenden beschriebenen Verknüpfungen und Funktionen können theoretisch beliebig viele Eingänge aufweisen. Nachfolgend wählt man zur Erklärung der Zusammenhänge Elemente mit je zwei Eingängen, ausgenommen bei speziellen Funktionen. Für Schaltungen mit mehr als zwei Eingängen gelten sinngemäss die gleichen Gesetze.

Die einzelnen Verknüpfungen und Funktionen werden im Praktikum in der LEMPS-Umgebung durch einzelne Programme realisiert und können mit Hilfe der I/O-Einheit untersucht werden. Die Schalter dienen dabei als Eingänge, die LEDs als Ausgänge.

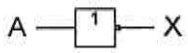
Dabei gelten die folgenden Definitionen:

Eingänge:	Schalterstellung 0 -> logischer Pegel 0
	Schalterstellung 1 -> logischer Pegel 1
Ausgänge:	LED aus -> logischer Pegel 0
	LED ein -> logischer Pegel 1

NICHT-Verknüpfung

Eine NICHT-Verknüpfung (Inverter) hat einen Eingang und einen Ausgang. In der Digitaltechnik ist der Ausdruck Inverter oder der Ausdruck NOT gebräuchlicher.

Symbol:



Es gilt: Am Ausgang eines Inverters liegt immer das invertierte Eingangssignal (Umkehrung).

$X = \overline{A}$ (lies X = A nicht oder X = A quer)

Beispiel: Ist die Raumtemperatur zu hoch, dann ist die Heizung ausgeschaltet.

Aufgabe 1: Ausmessen der NICHT-Funktion

Die Ausgangszustände einer NICHT-Verknüpfung mit einem Eingang ist auszumessen und in der Wahrheitstabelle einzutragen.

Vorgehen: Nehmen Sie den PC und das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb und laden Sie das Programm L_NOT.ASC.

Hilfsmittel: PC, LEMPS, L_NOT.ASC

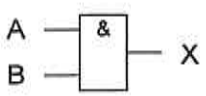
Wahrheitstabelle:

Eingang S7	Ausgang LED7
0	
1	

UND-Verknüpfung

UND-Verknüpfungen haben zwei oder mehrere Eingänge und einen Ausgang.

Symbol:



Es gilt: Die UND-Verknüpfung hat am Ausgang nur dann den Wert 1, wenn alle Eingangssignale gleichzeitig den Wert 1 haben.

$X = A \wedge B$ (lies X = A und B)

Beispiel: Eine Heirat ist nur dann gültig, wenn die Braut und der Bräutigam sich das Ja-Wort geben.

Aufgabe 2: Ausmessen der Funktion einer UND-Verknüpfung

Die Ausgangszustände einer UND-Verknüpfung mit zwei Eingängen sind auszumessen und in der Wahrheitstabelle einzutragen.

Vorgehen: Nehmen Sie den PC und das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb und laden Sie das Programm L_AND.ASC.

Hilfsmittel: PC, LEMPS, L_AND.ASC

Wahrheitstabelle:

Eingang S6	Eingang S7	Ausgang LED7
0	0	
0	1	
1	0	
1	1	

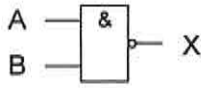
Aufgabe 3: Realisieren einer UND-Verknüpfung mit zwei Schaltern

Eine Lampe soll nur brennen, wenn der Schalter 1 und der Schalter 2 geschlossen ist. Zeichnen Sie ein entsprechendes Schema.

NAND-Verknüpfung

NAND-Verknüpfungen (Nicht-UND) haben zwei oder mehrere Eingänge und einen Ausgang.

Symbol:



Es gilt: Die NAND-Verknüpfung hat am Ausgang nur dann den Wert 0, wenn alle Eingangssignale gleichzeitig den Wert 1 haben.

$X = \overline{A \wedge B}$ (lies X = A und B quer)

Beispiel: Der Alarm in der Bank geht nicht los, wenn der Securitasmann um 21 Uhr und um 22 Uhr den Rundgang quittiert.

Aufgabe 4: Ausmessen der Funktion einer NAND-Verknüpfung

Die Ausgangszustände einer NAND-Verknüpfung mit zwei Eingängen sind auszumessen und in der Wahrheitstabelle einzutragen.

Vorgehen: Nehmen Sie den PC und das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb und laden Sie das Programm L_AND.ASC.

Hilfsmittel: PC, LEMPS, L_AND.ASC

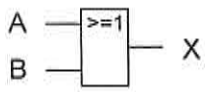
Wahrheitstabelle:

Eingang S6	Eingang S7	Ausgang LED 0
0	0	
0	1	
1	0	
1	1	

ODER-Verknüpfung

ODER-Verknüpfungen haben zwei oder mehrere Eingänge und einen Ausgang.

Symbol:



Es gilt: Die Oder-Verknüpfung hat am Ausgang nur dann den Wert 1, wenn mindestens ein Eingangssignal den Wert 1 hat.

$X = A \vee B$ (lies X = A oder B)

Beispiel: Im Briefkasten ist Post, wenn der Pöstler einen Brief, oder der Zeitungsverträger die Zeitung bringt.

Aufgabe 5: Ausmessen der Funktion einer ODER-Verknüpfung

Die Ausgangszustände einer ODER-Verknüpfung mit zwei Eingängen sind auszumessen und in der Wahrheitstabelle einzutragen.

Vorgehen: Nehmen Sie den PC und das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb und laden Sie das Programm L_OR.ASC.

Hilfsmittel: PC, LEMPS, L_OR.ASC

Wahrheitstabelle:

Eingang S6	Eingang S7	Ausgang LED7
0	0	
0	1	
1	0	
1	1	

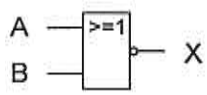
Aufgabe 6: Realisieren einer ODER-Verknüpfung mit zwei Schaltern

Eine Lampe soll nur brennen, wenn der Schalter 1 oder der Schalter 2 geschlossen ist. Zeichnen Sie ein entsprechendes Schema.

NOR-Verknüpfung

NOR-Verknüpfungen (Nicht-ODER) haben zwei oder mehrere Eingänge und einen Ausgang.

Symbol:



Es gilt: Die NOR-Verknüpfung hat am Ausgang nur dann den Wert 0, wenn mindestens ein Eingangssignal den Wert 1 hat.

$X = \overline{A \vee B}$ (lies X = A oder B quer)

Beispiel: Die Heizung ist ausgeschaltet, wenn die Zimmertemperatur erreicht, oder das Heizöl ausgegangen ist.

Aufgabe 7: Ausmessen der Funktion einer NOR-Verknüpfung

Die Ausgangszustände einer NOR-Verknüpfung mit zwei Eingängen sind auszumessen und in der Wahrheitstabelle einzutragen.

Vorgehen: Nehmen Sie den PC und das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb und laden Sie das Programm L_OR.ASC.

Hilfsmittel: PC, LEMPS, L_OR.ASC

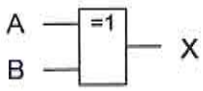
Wahrheitstabelle:

Eingang S6	Eingang S7	Ausgang LED 0
0	0	
0	1	
1	0	
1	1	

XOR-Verknüpfung

XOR-Verknüpfungen (Antivalenzschaltungen) haben meistens zwei Eingänge und einen Ausgang.

Symbol:



Es gilt: Die XOR-Verknüpfung hat am Ausgang dann den Wert 1, wenn nur an einem Eingangssignal der Wert 1 anliegt.

$$X = (\overline{A} \wedge B) \vee (A \wedge \overline{B})$$

Aufgabe 8: Ausmessen der Funktion einer XOR-Verknüpfung

Die Ausgangszustände einer XOR-Verknüpfung mit zwei Eingängen sind auszumessen und in der Wahrheitstabelle einzutragen.

Vorgehen: Nehmen Sie den PC und das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb und laden Sie das Programm L_XOR.ASC.

Hilfsmittel: PC, LEMPS, L_XOR.ASC

Wahrheitstabelle:

Eingang S6	Eingang S7	Ausgang LED 7
0	0	
0	1	
1	0	
1	1	

Zeitablaufdiagramm

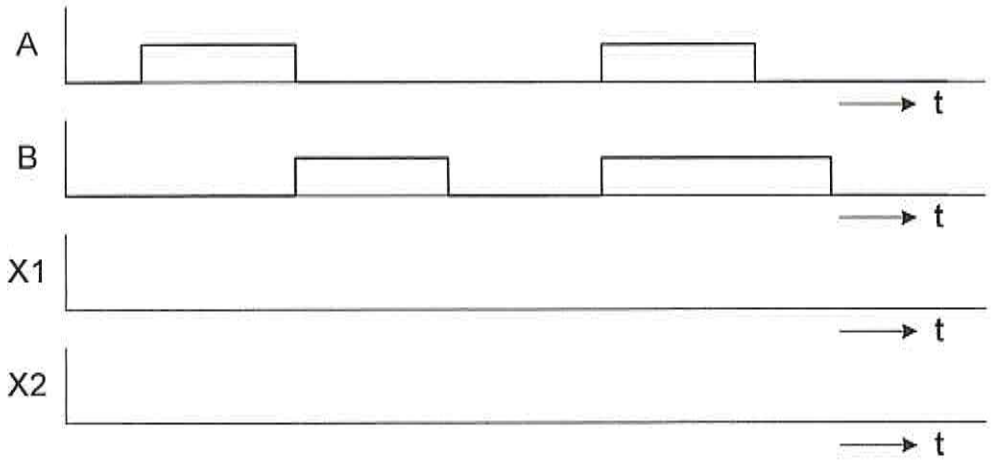
Die Funktion einer Steuerung oder eines Programmes wird häufig auch in Abhängigkeit der Zeit dargestellt. Das Zeitablaufdiagramm ist die Grundlage bei vielen zeit- und prozessgeführten Anlagen oder Geräten, z.B. bei Waschmaschinen oder Wäschetrocknern. Man kann damit auch die Funktionen von digitalen Verknüpfungssteuerungen oder die Funktion von Speichern verdeutlichen.

Aufgabe 9:

Ergänzen Sie das untenstehende Zeitdiagramm und vergleichen Sie die Resultate mit den Wahrheitstabellen der Aufgaben 3 und 5.

Für das Zeitdiagramm gilt: $X1 = A \vee B$ (X1 = A ODER B)
$X2 = A \wedge B$ (X2 = A UND B)

Zeitdiagramm:



Signale und Spannungen in einem Mikrocontroller-System

In einem Mikrokontroller-System unterscheidet man hauptsächlich zwischen analogen Spannungen und digitalen Signalen. Bei den analogen Spannungen interessieren die einzelnen Spannungspegel (z.B. welche Spannung besitzt der Akku?), bei den Digitalen Signalen meistens nur die Zustände (z.B. ist der Ausgang high oder low?) oder die zeitlichen Abhängigkeiten zueinander.

Beispiele:

Analoge Werte (Spannungen und Ströme)	Digitale Signale / Pegel
- Spannung vom Netzgerät (U_{in}) - Logikspeisung - Ladezustand (Spannung) vom Akku - Pegel der Spannungsüberwachung (Resetschaltung) - Analoge Eingangsspannungen (A-D-Wandler) - Analoge Ausgangsspannungen (D-A-Wandler)	- Oszillatorfrequenz von Q1 - Eingangspegel - Ausgangspegel - Daten- und Adressleitungen - Chip-Selectleitungen - Serielle Daten (TxD, RxD)

Messmittel

Für das Messen dieser verschiedenen Grössen stehen entsprechende Messmittel zur Verfügung (jedes mit Vor- und Nachteilen).

Beispiele:

Messmittel	Einsatzgebiete	Vorteile	Nachteile
Multimeter	- messen von Spannungen und Strömen - Service	- günstig - einfache Handhabung	- schnelle, zeitliche Abhängigkeiten nicht messbar
Logiksonde	- messen von Pegeln - Service	- günstig - einfache Handhabung	- keine analogen Werte - schnelle Änderungen schlecht messbar
KO	- messen von Spannungen und Pegeln - Darstellung von Zeitlichen Abhängigkeiten - Entwicklung, Service	- universellstes Instrument	- teuer
Logikanalyser	- gemeinsame Messungen der Daten- und Adressleitungen - Entwicklung	- komplexe, zeitliche Zusammenhänge messbar	- Handhabung - teuer

Für das Praktikum steht jedem Teilnehmer ein Universalinstrument PEN-Digital-Multimeter zur Verfügung, zusätzlich pro Gruppe ein KO.

PEN-Digital-Multimeter

Das PEN-Digital-Multimeter ist ein einfach zu gebrauchendes Multimeter in handlicher Form.

Messbereiche:

Gleichspannung	200mV...500V
Wechselspannung	2...750V
Widerstandsmessung	200Ω....20MΩ

Technische Daten:

Polaritätsanzeige
„1" blinkt bei Bereichsüberschreitung
<BAT> bei Batterie-Leeranzeige
Dioden Testfunktion
Durchgangstester
Data-Hold-Taste
TTL und CMOS Logic Level Indicator
(TTL = 5V; CMOS 3...18V)

-> Schalten Sie bitte den PEN-Tester nach Gebrauch wieder aus (Stellung „OFF")!

Spannungsmessungen mit dem PEN-Digital-Multimeter

Spannungsmessungen können mit dem PEN-Tester einfach durchgeführt werden. Drehschalter auf gewünschte Spannungsform einstellen (DCV oder ACV), der Messbereich und die Polarität wird automatisch gewählt und angezeigt.

Anschlüsse rot -> + U
schwarz -> GND (-U)

Aufgabe 1

Messen Sie mit dem PEN-Tester die folgenden Spannungen am LEMPS.

Vorgehen: Nehmen das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb (es muss kein Programm geladen sein). Drehschalter vom PEN-Tester in Stellung „DCV".

Hilfsmittel: LEMPS, PEN-Tester

Messungen von Spannungen:

Was?	Wo?	Sollwert U	Istwert U
Ausgangsspannung Netzgerät	Stecker auf Print	9-12 V	
Logikspeisung	Spannungsregler LM317 (Gehäuse)	5,0 V	
Akkuspannung	Anschlüsse Akku	3,6 V	
Eingangsspannung Pot1 (CCW)	Potentiometer 1 auf ADIOB	0 V	
Eingangsspannung Pot1 (CW)	Potentiometer 1 auf ADIOB	5,0 v	

Bemerkungen:

Pegelmessungen mit dem PEN-Digital-Multimeter

Für Messungen von Logikpegeln muss der PEN-Tester zusätzlich über das separate Kabel gespiesen werden. Er wird, wenn möglich, direkt an die Speisung von der zu messenden Logik angeschlossen.

Anschlüsse: weiss -> GND (0V)
rot -> +5V

Aufgabe 2

Messen Sie mit dem PEN-Tester die folgenden Pegel am LEMPS.

Vorgehen: Nehmen das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb (es muss kein Programm geladen sein). Drehschalter vom PEN-Tester in Stellung „LOGIK".

Hilfsmittel: LEMPS, PEN-Tester

Messungen von Pegeln:

Was?	Wo?	Sollwert Pegel	Istwert Pegel
Logikspeisung	Spannungsregler LM317 (Gehäuse)	H	
Logikspeisung	GND	L	
Resetsignal (normal)	z.B. IC 6 / Pin 13	H	
Resetsignal (Schalter gedrückt)	z.B. IC 6 / Pin 13	L	
Datenleitung D0	IC 4 / Pin 11	H + L	

Bemerkungen:

Aufgabe 3

Ermitteln sie die Schaltpunkte für die Pegel L und H vom PEN-Tester im LOGIK-Modus. Am Mittelabgriff des Potentiometers kann eine einstellbare Spannung (0..+5V) abgegriffen werden.

Vorgehen: Nehmen das LEMPS mit ADIOB am Portstecker X2 (oben) nach Anleitung in Betrieb (es muss kein Programm geladen sein). Drehschalter vom PEN-Tester in Stellung „LOGIK".

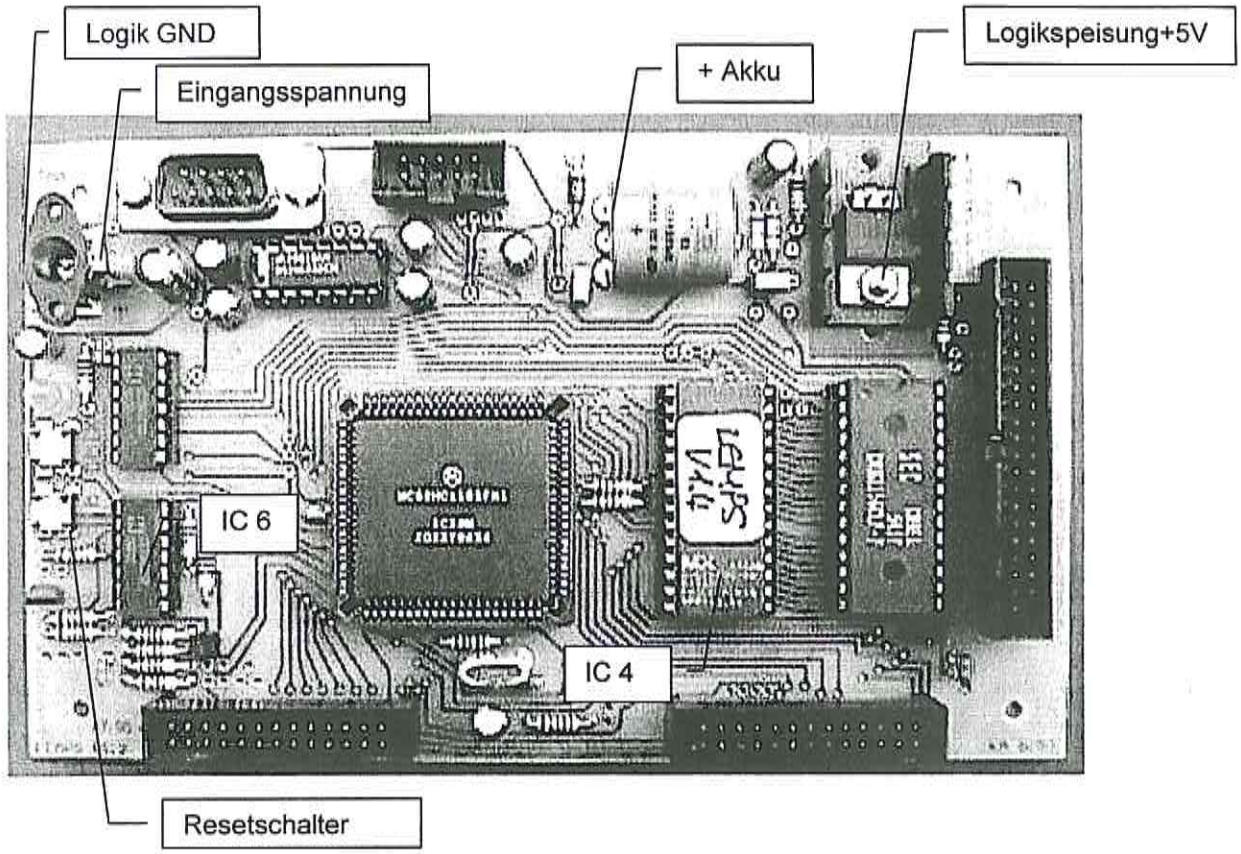
Hilfsmittel: LEMPS, PEN-Tester

Messen der Schaltpegel:

	Spannung U in V	
	von	bis
L-Pegel		
Undefiniert (Störabstand)		
H-Pegel		

Bemerkungen:

Messpunkte auf der Mikrocontroller-Leiterplatte

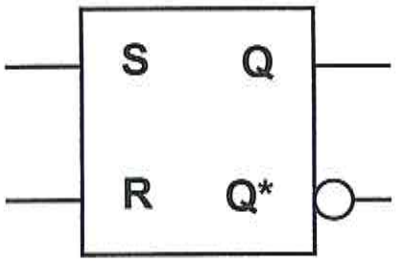


Schema und Bestückungsplan siehe Anhang LEMPS

Binärspeicher

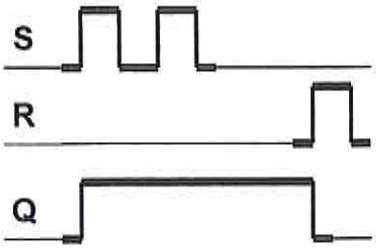
Alle Zeitdiagramme basieren auf den Abläufen nach RESET.
S = Set R = Reset D = Data Input
C = Clock Input Q / Q* = Output / Output quer

RS-Flipflop

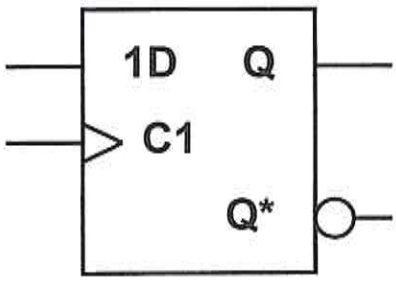


.....

.....

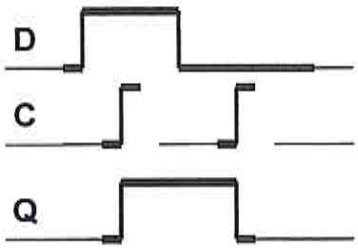


D-Flipflop

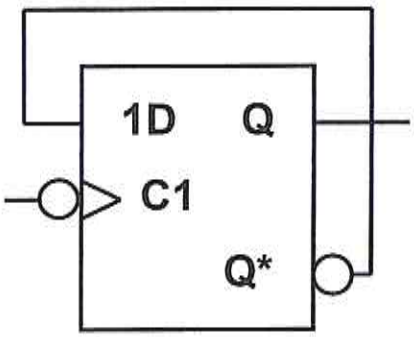


.....

.....

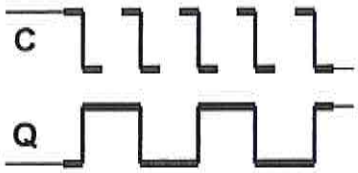


T-Flipflop

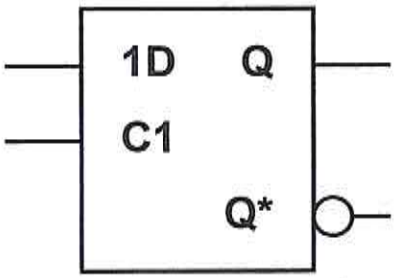


.....

.....

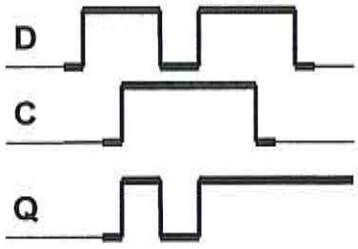


I/O-Latch



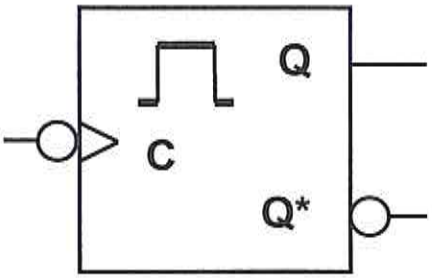
.....

.....



Diverse Grundschaltungen

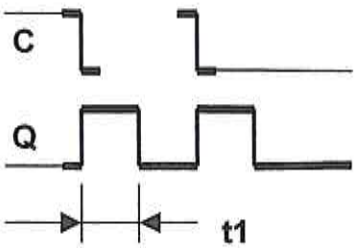
Alle Zeitdiagramme basieren auf den Abläufen nach RESET.
C = Clock Input Q / Q* = Output / Output-quer
E = Eingang A = Ausgang

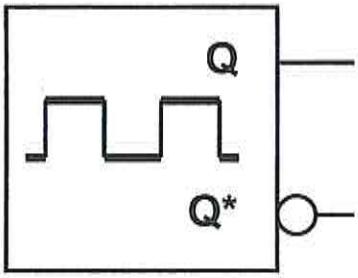


Monoflop

.....

.....

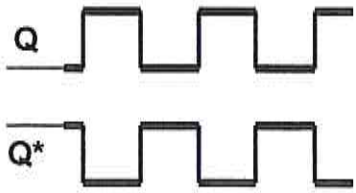


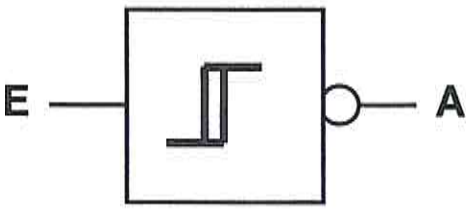


Clock-Oszillator

.....

.....

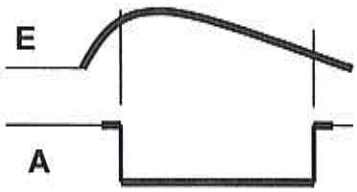


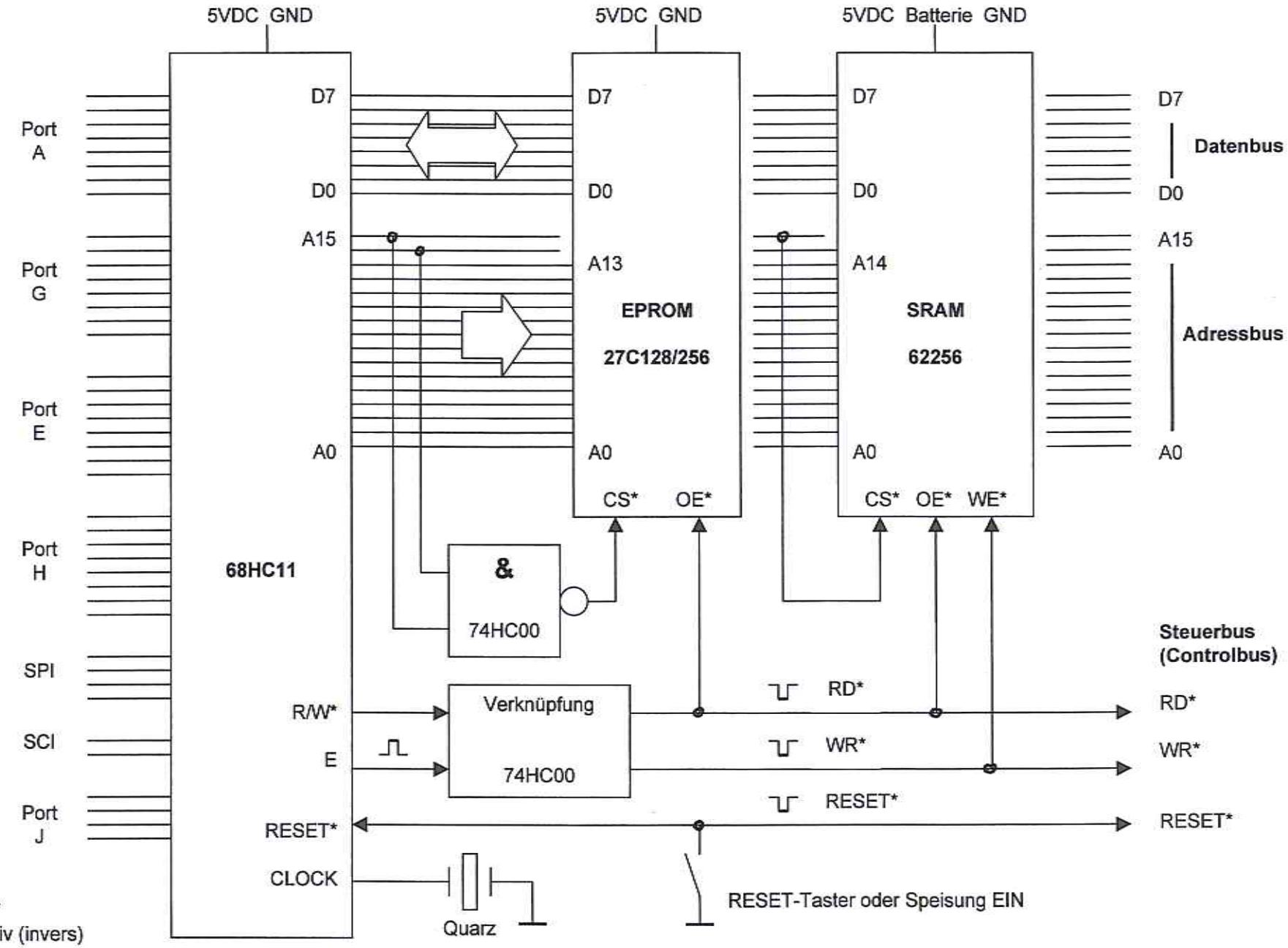


Schmitt-Trigger

.....

.....




5VDC GND
5VDC GND
5VDC Batterie GND
Port A
Port G
Port E
Port H
SPI
SCI
Port J
D7
D0
A15
A0
68HC11
A13
A14
EPROM
27C128/256
SRAM
62256
CS*
OE*
WE*
&
74HC00
Verknüpfung
74HC00
R/W*
E
RESET*
CLOCK
Quarz
RD*
WR*
RESET*
Datenbus
Adressbus
Steuerbus
(Controlbus)
RESET-Taster oder Speisung EIN
SIGNAL*
= low aktiv (invers)

Zu vorhergehender Seite

Prinzipschema LEMPS

Das Prinzipschema zeigt folgende Funktionsblöcke:

- **Controller** HC11

- **EPROM**-Programmspeicher (hier: Monitorprogramm für LEMPS)
  EPROM = Erasable Programmable Read Only Memory = Lösch- und neu programmierbarer Nur-Lese-Speicher

- **SRAM**-Datenspeicher (auch Programmspeicher)
  SRAM = Static Random Access Memory = Statischer, beliebig zugreifbarer Schreib- und Lesespeicher

- **EPROM-Adressdecodierung** mit NAND-Gatter 74HC00
  Die benutzten 16KB des EPROM liegen im obersten Viertel (Adressen $C000 bis $FFFF) des Speichers und werden demzufolge von den NAND-verknüpften Adressbits A15 und A14 angesprochen, NAND darum, weil das Freigabesignal CS* für den EPROM-Chip invers sein muss.

- Zur **SRAM-Adressdecodierung** wird direkt das Adressbit A15 verwendet, da das RAM die Adressen $0000 bis $7FFF belegt (untere Speicherhälfte).

- Der **16-Bit-Adressbus (16 Busleitungen) ist unidirektional**, d.h. der Controller HC11 ist der einzige Baustein, der auf den Adressbus schreiben kann (Adressenausgabe).

- Der **8-Bit-Datenbus (8 Busleitungen) ist bidirektional**, d.h. alle Bausteine teilen sich die Busleitungen und können darauf Datenbytes senden und empfangen. Damit keine Kurzschlüsse entstehen (Ausgang gegen Ausgang!), müssen die Bausteine **TRISTATE-Ausgänge** aufweisen, damit sie "bei Nichtgebrauch" von den Busleitungen abgetrennt werden können (= hochohmiger Aus-Zustand, High-Z).
  Die Steuerung der TRISTATE-Funktion eines Bausteins geschieht über die Adressdecodierung und die Anschlüsse CS*, OE* (**C**hip **S**elect und **O**utput **E**nable, beide low-aktiv) des Bausteins und das Signal RD* (CPU-Read) des Steuerbusses.

- Der **Steuerbus** besteht aus den Steuerleitungen CPU-Read-Signal RD*, CPU-Write-Signal WR*, RESET-Signal, CLOCK-Signal, Signale, die von der CPU ausgegeben werden. Die Anzahl der Leitungen und Signalfunktionen des Steuerbusses können – abhängig vom System – unterschiedlich sein.
  READ: Byte wird von der CPU gelesen, resp. empfangen.
  WRITE: Byte wird von der CPU zur Peripherie (Speicher, I/O) geschrieben, resp. gespeichert.

- Einige Logikbausteine sind für die RESET- und Speisungs-Ueberwachung, sowie die Steuerung der RAM-Datenhalte-Speisung (Backup-Batterie) eingesetzt.

Projektarbeit

Einleitung

Wenn ein Vorhaben die folgenden genannten Bedingungen erfüllt und im Hinblick darauf besondere organisatorische Massnahmen geplant und eingeleitet werden, bezeichnen wir es als Projekt:

- Es hat eine gewisse Einmaligkeit, wurde also in derselben oder in einer direkt vergleichbaren Form noch nicht durchgeführt.
- Es weist einen Umfang auf, in welchem eine Unterteilung in mehrere verschiedenartige, untereinander verbundene und wechselseitig voneinander abhängige Teilaufgaben erforderlich sind.
- An der Durchführung sind verschiedene Stellen innerhalb und auch ausserhalb einer Organisation beteiligt.
- Die durchzuführenden Teilaufgaben können sowohl untereinander, als auch mit anderen Aufgaben hinsichtlich der Zuteilung verschiedener Ressourcen (personeller, finanzieller, materieller u.a. Art) konkurrieren.

Dokumentation

Zur Dokumentation, die in einem Entwicklungsprojekt erstellt wird, gehören folgende wesentliche Dokumente:

- Pflichtenhefte, Anforderungspezifikationen (beschreiben WAS gemacht wird)
- Spezifikationen, Detailspezifikationen (beschreiben WIE es gemacht wird)
- Dokumentationen (beschreiben WIE es gemacht wurde)

Projekt-Management

Die Erfahrung lehrt, dass umfangreiche und komplexe Vorhaben (=Projekte) nur dann erfolgreich durchgeführt werden können, wenn dafür entsprechende organisatorische Vorkehrungen getroffen und Abläufe im Projekt eingehalten werden.

Im Rahmen des Projektmanagements werden die vielfältigen Aufgaben in einem Entwicklungsbereich ganzheitlich in einem Projekt eingebettet und unter Berücksichtigung entsprechender Kosten-, Termin- und Qualitätsmerkmale zielorientiert geplant und durchgeführt. Wegen der steigenden Anforderungen an die Entwicklungsergebnisse und der wachsenden technischen und organisatorischen Komplexität von Entwicklungsprojekten gewinnt der Projektgedanke in der Entwicklung daher zunehmend an Bedeutung.

Die folgenden Methoden und Checklisten sollen beim Einstieg in ein Projekt behilflich sein und die Durchführung erleichtern. Sie erreichen gesteckte Ziele weniger hektisch und effizienter, indem Sie Aufgaben strukturieren und Projekte systematisch behandeln.

1. Information

In der Regel formulieren die Kunden resp. Auftraggeber die Projektaufträge sehr vage. Die gestellte Aufgabe muss zuerst erfasst und verstanden werden, um sich ein Bild vom Ziel zu machen. Der Projektleiter muss daher die Problemsituation analysieren, die Ziele des Auftraggebers evtl. interpretieren und das Vorgehen skizzieren.

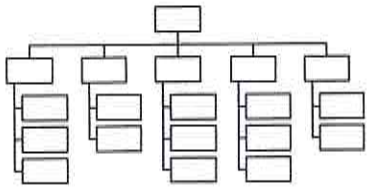
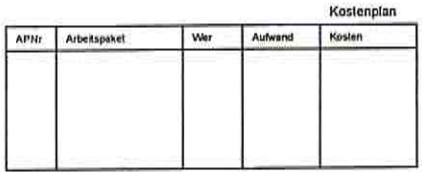
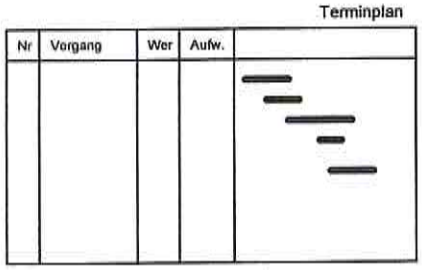
Für kleinere Projekte verwenden wir den **Projektauftrag** mit folgendem Inhalt:

- Wie lautet die Aufgabe oder der Auftrag genau?
- Welche Ziele und Ergebnisse werden erwartet?
- Wo sind die Abgrenzungen?
- Welche Randbedingungen müssen erfüllt werden?
- Welche Unterlagen müssen erstellt werden?
- Welche Mittel stehen zur Verfügung?
- Termine
- Welche ähnlichen Aufgaben wurden schon gelöst und wie?
- Sind alle Schnittstellen zur Umgebung spezifiziert?
- Wurde der Anwender der Software mit in die Analysephase einbezogen und all seine Anliegen aufgenommen?

2. Projektplanung

Beim Planen geht es um die Wege und Kosten zum Ziel. Dazu müssen die Abläufe geordnet und die Hilfsmittel und Werkzeuge festgelegt werden. Der Zeitbedarf und die Kosten sind für die einzelnen Arbeiten zu schätzen.

Projektpläne:

Darstellung	Inhalt	Vorgehen
a) 	**Objektstruktur, Produktstruktur** Hierarchische Struktur des Produkts mit den Teilsystemen und Teil-Teil-Systemen	1. Strukturierung nach bereits vorhandener Produktstruktur oder 2. neue Gliederung auf Grund einer Objekt- resp. Produktanalyse
b) 	**Projektstruktur- und Kostenplan** Projektstruktur bestehend aus Arbeitspaketen Später auch Ressourcen, Aufwand in Manpower und Kosten	1. Projektstruktur aus Produktstruktur ableiten (welche Arbeiten sind zu erledigen, um das Produkt resp. die Produktteile zu erhalten?) 2. Arbeitspakete grob Personen oder Stellen zuordnen 3. Aufwand und Kosten grob schätzen
c) 	**Terminplan** Verfeinerung des Meilenstein-Terminplans mit Tätigkeiten, Ressourcen, Aufwand, Dauer, Terminen und Abhängigkeiten	1. Projektinfo ausfüllen und Projektkalender erstellen 2. Tätigkeiten aus Projektstrukturplan ableiten 3. Abhängigkeiten eintragen 4. Ressourcen für Idealfall ohne Rücksicht auf Kapazität zuordnen 5. Aufwand schätzen und eintragen 6. Optimierung bez. Terminen, Belastung, Kosten u.s.w.

Objekt- oder Produktstrukturplan

Der Objekt- oder Produktstrukturplan dient als Grundlage für den Projektstrukturplan, für den Dokumentenplan und evtl. für den Eckterminplan. Die Gliederung entspricht in der Regel einer funktionsbezogenen Top-Down-Sicht des Objekts/Produkts.

Beispiel eines Strukturplans:

Nr.	Teil
1	Tasten einlesen
11	Entprellung
12	Wert speichern
2	Ausgabe
21	Anzeigen mit LEDs
22	Wert Ausgabe auf LCD
3	Berechnungen
31	Wandlung
	usw.

Projektstruktur- und Kostenplan

Der Projektstrukturplan zeigt die aufgabenmässige Gliederung des Projekts. Die kleinsten Elemente der Projektstruktur sind Arbeitspakete, die klar zugeordnet werden können. Für jedes Element ist der Aufwand und sind die Kosten zu schätzen.

Terminplan

Im Terminplan (Meilenstein-Plan) werden die Etappenziele (Meilensteine) entlang des Projekts terminiert und definiert und die Haupttätigkeiten entsprechend eingeplant. Der Terminplan ist die Basis für die detailliertere Planung der Tätigkeiten.

3. Entscheiden

Sind die Beschaffung der Informationen und Planung optimal erfolgt, fällt die Entscheidung für eine der möglichen Lösungen (bei mehreren Varianten) nicht mehr schwer. Werden Entscheide hinausgezögert, kostet das oft mehr Zeit und schlimmstenfalls gibt es nichts mehr zu entscheiden, weil Sachzwänge vorliegen.

- Welche Lösungsvarianten stehen zur Wahl?
- Wie wurden die Lösungsvarianten bewertet (intuitiv, Nutzwertanalyse)?
- Welche Kriterien sind für den Entscheid ausschlaggebend?

4. Realisierung oder Ausführung

Die Realisierung oder Ausführung kann, wie bei Fertigungsaufgaben üblich, den zeitlich grössten Hauptteil einer Aufgabe umfassen oder, z.B. beim Durchführen einer Veranstaltung, nur einen kleinen Teil beanspruchen. Der erstellte Terminplan ist in der Regel einzuhalten und sollte nicht leichtfertig geändert werden.

- Besteht eine klare, saubere Aufgabenstellung?
- Welche Qualitätsanforderungen sind einzuhalten?
- Erst strukturieren und konzipieren, später codieren!
- Wurde Wert auf Einfachheit gelegt (KISS = Keep It Stupid Simpel)?
- Welche Konsequenzen ergeben sich aus allfälligen Abweichungen vom Terminplan?
- Ist die Software so gestaltet, dass sie möglichst allgemein verwendet werden kann?
- Ist eine Entwicklungsdokumentation vorhanden und wird sie laufend nachgeführt?

5. Kontrolle oder Prüfung

Jede ausgeführte Arbeit ist zu kontrollieren, bevor sie aus den Händen gegeben wird. Kontrollieren heisst z.B. nochmals durchlesen, nachrechnen, mit Vorgaben vergleichen, Software ausgiebig testen. Gute Resultate bringt eine Kontrolle, wenn sie nicht auf dieselbe Art oder durch die gleiche Person erfolgt, wie die Ausführung.

- Erweisen sich die in der Planung festgelegten Prüfkriterien als richtig und vollständig?
- Deckt der Test die im Betrieb auftretenden Fälle ab (aussagekräftige Tests)?
- Erfolgt die Kontrolle auf eine andere Art oder durch eine andere Person als die Ausführung?
- Wurden grundlegende Mängel protokolliert?
- Ist die Dokumentation aktuell und sind allfällige Änderungen nachgetragen?
- Sind die einzelnen Module einzeln getestet?

6. Auswerten

Lassen Sie die einzelnen Schritte des Auftrages oder Projektes nochmals revue passieren.

- Was war gut?
- Was war unbefriedigend?
- Welche Verbesserungen sind für künftige Aufgaben möglich?

Software-Qualität

Wenn eine Brücke die Belastung mit 7 schweren Lokomotiven aushält, so dürfen wir daraus schliessen, dass sie mit nur einer Lokomotive belastet, sicher nicht einstürzt. Bei einer Software dürfen wir das nicht! Wenn der Programmierer den Fall mit nur einer Lokomotive nicht abgefangen hat, so stürzt das Programm ab. Programme zeigen im Gegensatz zu physikalischen Dingen kein stetiges Verhalten. Die periodische Wartung einer Lokomotive beugt möglichen Defekten vor. Software hingegen kann man nicht ölen, es gibt keine Verschleissteile, vorbeugende Wartung ist nicht möglich. Software-Qualität ist im Grunde genommen nicht prüfbar.

Nur die Entwicklungsabteilung (oder der Entwickler) kann die Qualität der Software sichern.
Software-Qualität entsteht durch sauberes, systematisches Arbeiten.
Die einschlägige Literatur beschreibt weitere Möglichkeiten. Aber schon das Einhalten der obigen Massnahmen stellt oft einen enormen Fortschritt dar.

Projektarbeit: Tastatur

Einleitung, Anwendung

Das Mikrocontrollersystem LEMPS wird um die Hardwarekomponente "numerische Tastatur" erweitert. Dabei handelt es sich um ein komplettes numerisches 10er-Tastaturmodul, das ohne zusätzliche Hardware (ausser einem Widerstandsnetzwerk) direkt an den PORT H des LEMPS angeschlossen werden kann.
Eine dazu passende Treibersoftware (Include-Datei), die das Tastaturmodul auf allfällig gedrückte Tasten untersucht und Tastenwerte an andere Programme übergeben kann, ist vorhanden, aber nicht komplett.

Aufgabe, Ziel

Sie ahnen es bereits: Der fehlende Teil der Tastatur-Treibersoftware ist als Assembler-Quelltext zu schreiben. Genauer: Die Include-Datei mit dem Namen **KEYB_INC.ASC** (dies ist eine Sammlung von Subroutinen, die im Haupt-Quelltext mit der Assembler-Direktive INCL eingefügt werden kann und die der Abfrage einer Tastatur dient) ist um eine Subroutine mit dem Namen **KEYSCAN** zu erweitern.

Damit Sie vorab eine Idee des ungefähren Aufwandes haben:
Die Subroutine KEYSCAN kann mit ca. 20 Assembler-Zeilen realisiert werden. Je nach gewähltem Lösungsansatz werden evtl. eine Hilfsroutine zu 5 Zeilen und eine oder zwei 1-Byte-Hilfsvariablen dazukommen.

Information, Hilfsmittel

Die zu erweiternde Include-Datei KEYB_INC.ASC finden Sie im gewohnten Verzeichnis:
C:\PROGRAMME\LEMPS\EM-PROG\.

Achtung:
Die Datei KEYB_INC.ASC bitte im obigen Pfad mit Hilfe des Explorers umbenennen, damit Sie Ihren neuen Quelltext beim Ausführen des Batch "LEMPS_neu_installieren" ("Sanitätsköfferli") nicht aus Versehen überschreiben.

Zum Testen Ihrer fertigen Version der Include-Datei (ehem. KEYB_INC.ASC) gehen Sie wie folgt vor:

1)
Ihre Datei in folgendes Verzeichnis kopieren: **C:\PROGRAMME\LEMPS\SYSTEM*.*** und sie wieder umbenennen nach **KEYB_INC.ASC**. Bisherige Datei unter ...\SYSTEM\... darf überschrieben werden.

2)
Zum Testen können Sie die Programme **KEY_DEMO.ASC** oder **TIMER.ASC** laden und starten wie gewohnt. Diese beiden Programme enthalten bereits die Befehle zum Einbinden Ihrer Datei KEYB_INC.ASC.

Vorgehen

- Das Projekt kann als Team-Arbeit (z.B. im 2er-Team) realisiert werden. Bilden Sie, falls gewünscht, zuerst ein Projekt-Team.
- Beschaffen Sie sich die notwendigen Informationen und erstellen Sie einen kurzen Projektauftrag.
- Erstellen Sie die Projektplanung (mindestens Terminplan mit Aktivitäten).
- Bei mehreren Lösungsmöglichkeiten begründen Sie kurz Ihre Entscheidung für die gewählte Lösungsvariante.
- Realisieren Sie das Programm und erstellen sie parallel dazu (evtl. als Kommentar im Programm) die Dokumentation.
- Prüfen Sie zuerst das Programm selber und lassen Sie es anschliessend von einem anderen Teilnehmer testen.
- Lassen Sie die einzelnen Schritte des Projektes nochmals revue passieren und präsentieren Sie kurz Ihre Lösung in der Klasse.

Projektinfos, Aufgabenbeschreibung

```
*//////////////////////////////////////////////////////////////////////////
**************************************************************************
*
*      INCLUDE :    KEYB_INC.ASC
*      FUNKTION:    Subroutinen fuer den Betrieb einer 10er-Tastatur
*      HARDWARE:    LEMPS mit 3x4-Matrix-Keyboard an PORTH
*      VERSION :
*
**************************************************************************
*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
```

Die Include-Datei **KEYB_INC.ASC** soll folgende 2 Subroutinen zur Abfrage und zur Auswertung einer Tastatur enthalten (eine davon, nämlich KEYCONV, steht fertig zur Verfügung und darf nicht verändert werden):

```
**************************************************************************
* 1 Subroutine KEYSCAN, Quelltext ist zu entwickeln, Projektarbeit
**************************************************************************
* Funktion:
*   Alle Tasten einer 10er-Tastatur, die als 3 x 4 Schalter-Matrix
*   aufgebaut ist (0...9, *, #, siehe Distrelec AG), werden abgefragt
*   (Scanner-Prinzip) und deren Scan-Code im Akku A zurueckgegeben.
*   Falls keine gedrueckte Taste gefunden: $ff im Akku A.
* Eingang :
*   keine Eing.-Daten
* Ausgang :
*   Scan-Code im Akku A,
*   falls keine Taste gedrueckt: Akku A =$ff
*
*
**************************************************************************
* 2 Subroutine KEYCONV, Quelltext ist vorhanden
**************************************************************************
* Funktion:
*   Diese Wandler-Routine konvertiert den nicht-stetigen (nicht
*   zusammenhaengenden) Scan-Code (Akku A) in einen stetigen Binaercode
*   mit den Werten 0...11d (Tastencode, entspr. den Tasten 0...9, *, #)
*   und gibt ihn im Akku A zurueck.
* Eingang :
*   Scan-Code im Akku A
* Ausgang :
*   Binaerer Tastencode im Akku A, 0...11d
*   Fehlercode: $ff
*   Keine Taste gedrueckt: $ff
```

Funktionsweise der 10er-Tastatur

```
**************************************************************************
* TASTATUR-AUFBAU
* 3 x 4 -Schalter-Matrix-Tastatur (s. Bild unten):
*   12 Taster sind so angeordnet, dass beim Druecken einer Taste
*   eine leitende Verbindung zwischen der entspr. Kolonne und der
*   zugehoerigen Zeilen-Leitung entsteht.
*   Beispiel: Der Schalter der Taste 6 verbindet das Kolonnen-Bit 7 mit
*   dem Zeilen-Bit 2, falls die Taste gedrueckt wird.
```

```
* 3 x 4 -TASTATUR,
* AUFBAU DER MATRIX:                ZEILEN     Y-Wert lesen (Byte lesen)
*
*                                                1,2,3   Tasten
*                                                     4,5,6
*                                    PORTH                   7,8,9
*                                     Bit                          *,0,#
*                    |     |     |
*                ____1_____2_____3___ 1          1     0     0     0
*                    |     |     |
*                    |     |     |
*                ____4_____5_____6___ 2          0     1     0     0
*                    |     |     |
*                    |     |     |
*                ____7_____8_____9___ 3          0     0     1     0
*                    |     |     |
*                    |     |     |
*                ____*_____0_____#___ 4          0     0     0     1
*                    |     |     |
*                    |     |     |
*
* PORTH, Bit         5     6     7
* KOLONNEN
*
* Tasten             X-Wert-Ausgabe (Byte schreiben)
*
* 3, 6, 9, #         0     0     1
*
* 2, 5, 8, 0         0     1     0
*
* 1, 4, 7, *         1     0     0
*
*
* (Bit 0 von PORTH wird nicht genutzt: Reserve-Bit)
```

```
* Die vollstaendigen Scan-Codes:
*
*                Bit:  7  6  5    4  3  2  1    0
*                      ------------------------
*     Taste "1"    %   0  0  1    0  0  0  1    0    =  $22
*     Taste "2"    %   0  1  0    0  0  0  1    0    =  $42
*     Taste "3"    %   1  0  0    0  0  0  1    0    =  $82
*
*     Taste "4"    %   0  0  1    0  0  1  0    0    =  $24
*     Taste "5"    %   0  1  0    0  0  1  0    0    =  $44
*     Taste "6"    %   1  0  0    0  0  1  0    0    =  $84
*
*     Taste "7"    %   0  0  1    0  1  0  0    0    =  $28
*     Taste "8"    %   0  1  0    0  1  0  0    0    =  $48
*     Taste "9"    %   1  0  0    0  1  0  0    0    =  $88
*
*     Taste "*"    %   0  0  1    1  0  0  0    0    =  $30
*     Taste "0"    %   0  1  0    1  0  0  0    0    =  $50
*     Taste "#"    %   1  0  0    1  0  0  0    0    =  $90
```



```
*************************************************************************
* SCAN-CODE
* So entsteht der Tasten-Scan-Code in der Subroutine KEYSCAN:
*   Ein Kolonnenzeiger-Byte wird an PORTH ausgegeben (X-Wert)
*   und speist zuerst das Kolonnen-Bit 7 mit High-Pegel.
*   PORTH wird gelesen und die Bits 1...4 (Y-Wert) als Zeilen-Antwort
*   ausgewertet.
*   Nach Kolonnen-Bit 7, wird Bit 6 und anschliessend Bit 5 mit
*   High-Pegel gespiesen und jeweils die Bits 1...4 ausgewertet.
*
*   Wird gerade eine Taste gedrueckt, verbindet der entsprechende
*   Tastenkontakt die vertikale Kolonnenleitung, die gerade High-Pegel
*   fuehrt, mit der horizontalen Zeilenleitung und schaltet den
*   High-Pegel auf die Zeile durch.
*   Die vier Zeilen-Bits 1...4 werden durch Einlesen von PORTH und
*   Kombination mit dem Kolonnen-Zeiger-Byte in einer Byte-Variablen
*   erfasst, als Scan-Code abgespeichert und dem aufrufenden
*   Hauptprogrammen im Akku A zurueckgegeben.
```

Hinweise zur Funktion und Programmierung der Subroutine KEYSCAN

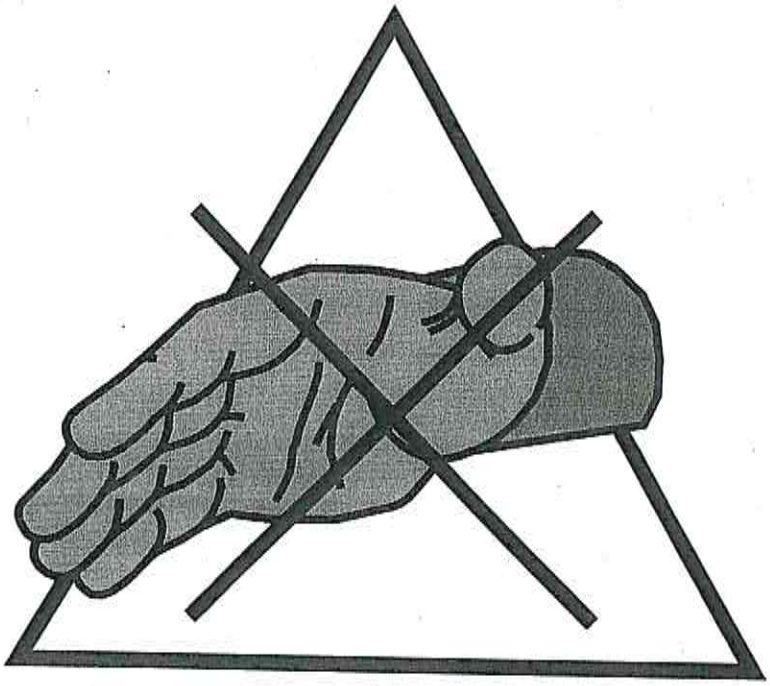
```
*************************************************************************
* Subroutine KEYSCAN
*************************************************************************
* Ein Voll-Scan (Abfrage) fuer das ganze 3 x 4 -Matrix-Keyboard wird
* ausgefuehrt.
*
* Tasten 0...9,*,#, 3 Kolonnen (Spalten) x 4 Zeilen.
* Bei der ersten gefundenen aktivierten Taste, wird die Scan-Routine
* abgebrochen und der zugehoerige Scan-Code im Akku A zurueckgegeben.
* Ist keine Taste gedrueckt, wird nach einem Scan-Durchlauf (alle 3
* Kolonnen abfragen) der Wert $ff im Akku A zurueckgegeben.
* Eine Tasten-Entprellung wird nicht ausgefuehrt.
*
* Die Kolonnen werden bitweise adressiert (gespiesen mit High) vom
* Kolonnenzeiger-Byte (X-Wert).
* Zugehoerige Zeilen-Antwort Bits 1...4 (Y-Wert) ist High bei
* gedrueckter Taste.
*
* ACHTUNG:
* Nur EIN Kolonnen-Bit darf jeweils als Ausgang konfiguriert sein und
* High-Pegel fuehren, die zwei anderen Kolonnen-Bits muessen waehrend
* dieser Zeit als Eingaenge konfiguriert sein.
* Grund: Kurzschlussgefahr beim gleichzeitigen Druecken von 2 Tasten!.
```

Weitere Hinweise:

Falls nötig, können Sie auch eigene Hilfs-Subroutinen und Variablen definieren.
Zusätzliche Hilfs-Subroutinen fügen Sie einfach direkt nach der Routine KEYSCAN ein.

Falls Sie Hilfsvariablen benötigen, deklarieren Sie diese am Schluss des Programmtextes unter dem Titel "Variablen im RAM" nach folgendem Beispiel (1-Byte-Hilfsvariable im RAM):

```
varname         rmb     1       Ihr Kommentar: Funktion der Variablen
```

Elektronische Komponenten sind ESD-gefährdet:
elektrostatische Entladungen können die Bauteile zerstören.

ESD-Schutzregeln

- **Die Leiterplatten-Module müssen immer - auch während der Arbeiten im Praktikum - im Koffer bleiben.**
- **Alle beteiligten Personen, Komponenten, Messgeräte und Werkzeuge sollten beim Arbeiten mit dem System immer gleiches elektrisches Potential aufweisen (0 Volt/GND/Erde).**
- **Berühren Sie keine Komponenten und deren Anschlüsse auf der Leiterplatte.**
- **Berühren Sie immer kurz die Alu-Halterung der Leiterplatte im Koffer, bevor Sie den RESET-Taster betätigen, Steckverbindungen auf der Leiterplatte vornehmen oder Messpunkte kontaktieren.**
- **Führen Sie die Messübungen erst nach einer Instruktion durch den Fachlehrer durch.**

LEMPS Monitor Tokens

Tokens sind Befehle welche dem LEMPS vom PC-Terminal aus gegeben werden können. Es ist auch möglich diese Befehle (z.B. aus einem Turbo Pascal Programm) via RS 232 an den LEMPS-Prozessor zu senden.

Argumente in [..] sind optional und müssen nicht in jedem Fall angegeben werden.

Token	*Argument*	*Beschreibung*
help		Gibt eine Uebersicht über die implementierten Tokens.
helptp		Gibt eine Uebersicht über die implementierten Tokens, die für den Aufruf aus Turbo Pascal Programmen vorgesehen sind.
regi	[reg dat]	Gibt die momentanen Inhalte der Prozessorregister aus. Wenn die Argumente vorhanden sind, so wird Register "reg" auf "dat" gesetzt.
xirq	adr	Nach betätigen der Taste XIRQ wird das Programm an der Adresse "adr" ausgeführt.
goto	adr	Das Programm an der Adresse "adr" wird ausgeführt.
auto	[adr]	Nach dem Aufruf dieses Tokens wird nach einem Reset das Programm automatisch an der Adresse "adr" ausgeführt. Ein mit auto gestartetes Programm wird durch die XIRQ-Taste unterbrochen und in den Terminalmode gebracht. Der Aufruf von "auto" ohne Adresse "adr" sperrt den Autostart.
inst	[adr]	Wie Auto, gedacht für kleine Installationsprogramme, die LEMPS erweitern.
read	adr1 [adr2]	Der Speicherinhalt im Bereich zwischen "adr1" und "adr2" wird an das Terminal ausgegeben.
write	adr dat1 [dat2 [datN]]	Die Datenbytes "dat1..datN" werden im Speicher ab Adresse "adr" abgelegt.
baud	dat	Einstellen der Baud-Rate am Prozessor dat = 0 300Bd 1 600Bd 2 1200Bd 3 2400Bd 4 4800Bd 5 9600Bd (8MHz Quarz) 6 19200Bd (Spez.-Quarz)
message	on / off	Das Prompt "LEMPS>" kann ein- oder ausgeschaltet werden.
dis	[adr]	Ab der Adresse "adr" werden 10 Zeilen disassembliert und am Terminal ausgegeben. Beim Aufruf von "dis" ohne "adr" wird von der nächsten Adresse an weiter disassembliert.
exit		Nach einem XIRQ (z.B. im auto-Mode) wird mit "exit" der Terminalmodus verlassen und das Programm weiter geführt.
iop*	[0..255]	Lesen (ohne Argument) oder schreiben (mit Argument) eines Ports, * = (a,e,g,h,j) für das entsprechende Port.

dirp*	[0..255]	Lesen (ohne Argument) oder schreiben (mit Argument) des entsprechenden Datenrichtungsregisters DDR*, * = (a,g,h,j) für das entsprechende Port.
inad	[Kanal]	AD-Wandler-Register lesen und am Terminal ausgeben. Kanal = (1..8). Ohne Argument werden alle 8 Kanäle gelesen und ausgegeben.
wand	Zahl	Umwandlung einer Dez-, Hex- oder Binärzahl in die anderen Zahlensysteme. Zahl = Dezimalzahl $Zahl = Hexadezimalzahl %Zahl = Binaerzahl "Zahl" kann in einem der drei Zahlensysteme eingegeben werden. Ausgegeben wird "Zahl" in den anderen beiden Zahlensystemen.

sample E/10 adr Bytes [AKanal [BKanal]][u|t Pegel[Kanal [Timeout]] [/x]

Mit diesem Token kann eine wählbare Anzahl analoger Messwerte über einen oder zwei Kanäle digitalisiert (8 Bit) und im RAM gespeichert werden. Nach der Erfassung aller Messwerte, sendet sie LEMPS an den PC. Den gesendeten Daten wird als Erkennung der String 'BODY' vorausgeschickt.
Die minimale Sampling-Zeit ist 40 E-Clockperioden.
Unterstützt werden auch Triggerfunktionen.

Argumente / Parameter

E/10	Anzahl E-Clock-Perioden * 10 => Messintervall (>=4)
Startadr	Anfangsadresse ab der die Messwerte abgelegt werden
Anz_Bytes	Anzahl zu erfassende Messwerte (Bytes)
AKanal	Nummer (1..8) des ersten Kanals (A-Kanal)
BKanal	Nummer (1..8) des zweiten Kanales (B-Kanal)
u¦t	'u' Triggerung auf ansteigende Flanke 't' Triggerung auf fallende Flanke
Pegel	Trigger-Pegel 0..FF
Kanal	Kanal auf den getriggert wird (default: A_Kan)
Timeout	Autotrigger auf Timeout*1s setzen, max 255s, Default = 10s Timeout=0: keine automatische Triggerung, LEMPS wartet auf einen Triggerimpuls
/x	Zeichen werden im ASCII-Format gesendet

Beispiele

1 sample a 0 100 /x

Samplingintervall $A*10 E-Clockperioden
Messwerte ab Adresse $0000 abspeichern
100 Messwerte aufnehmen
Resultat in ASCII-Format an PC

2 sample 4 2000 !256 3 2 u !80 4 10 /x

Speicheraufteilung LEMPS (Assembler-Anwendungen)

$FFFF	$FFC0...$FFFF	Interruptvektoren	EPROM
	$FC00...$FFBF	Initialisierung	
$F000			
$E000			
$D000	$D000...$FC00	**LEMPS-Monitor**	
	$CFFC...$CFFF	Vektoren (COP, RAM, RESTART)	
$C000			
$B000			**Reserve** nicht bestückt
$A000			
$9000			
$8000	$8000...$81FF	**RAM des HC11 (Anwender)**	**HC11**
$7000			**RAM**
$6000			
$5000			
$4000			
$3000			
$2000	$2000...$7FFF	Anwender-RAM (Assembler)	
		Reserviert für Stack	
		Reserviert für Monitor	
$1000	$1000...$107F	**Registerblock des HC11**	**HC11**
$0000	$0000...$0FFF	Anwender-RAM (Assembler)	**RAM**
	$0000...$00FF	**Direkte Adressierung**	

Kurs EM, Zwischentest Nr. 1

Anzahl Aufgaben	**20**	Bewertung	**1 Punkt pro Aufgabe**
Richtzeit	**40 Minuten**	Total	**20 Punkte**

Hinweise

Bitte nur **beiliegendes Antwortblatt** ausfüllen
Pro Aufgabe ist nur eine Antwort richtig
Führen Sie bitte Berechnungen **ohne Taschenrechner** durch

$A1 = Hexadezimalwert A1
%1010 = Binärwert 1010
23d = Dezimalwert 23

1. Welche der folgenden **Aussagen** ist **nicht** zutreffend?

   (A) Die Digitaltechnik löst zunehmend die Analogtechnik ab
   (B) Analog-Digital-Wandler verbinden die analoge mit der digitalen Welt
   (C) Bausteine der Digitaltechnik sind in zunehmendem Masse programmierbar
   (D) Mikroprozessoren arbeiten den Programmcode im Speicher sequenziell ab
   (E) Der Einsatz von Mikrocontrollern lohnt sich nur zur Lösung von sehr komplexen Steuerungsaufgaben

---

2. Aus welchen Hauptblöcken setzt sich ein einfacher **Mikrocontrollerchip** zusammen?

   (A) CPU, Arithmetikprozessor, Display
   (B) Prozessor, Speicher, Timer, A/D-Wandler
   (C) CPU, ROM, RAM, EEPROM
   (D) Prozessor, Programmspeicher, serielle Schnittstelle
   (E) CPU, Programmspeicher, Datenspeicher, I/O-Block

---

3. Wie heissen die zwei **8-Bit-Arbeitsregister** (Operations- resp. Resultatregister) der CPU im Mikrocontroller 68HC11?

   (A) Akku X und Y
   (B) Programmzähler PC
   (C) Indexregister
   (D) Akkumulatoren A und B
   (E) CCR

---

4. Welche 3 Programme (Werkzeuge) in der **"integrierten Entwicklungsumgebung"** (IDE) des LEMPSwin benötigen Sie mindestens, um eine Mikrocontroller-Anwendung programmieren zu können?

   (A) Compiler, Assembler, Relaxer
   (B) Editor, Assembler, Terminalprogramm
   (C) Editor, Monitorprogramm, Debugger
   (D) Assembler, Terminaleditor, Hexadezimal-Rechner
   (E) Assembler, Debugger, Monitorprogramm

---

5. Mit einem **4-Bit-Register** kann von 0 bis 15 gezählt werden (16 Werte). Wie viele Bits werden benötigt, um von 0 bis 31 (32 Werte, also die doppelte Anzahl) zählen zu können?

   (A) 8 Bit
   (B) Da es keine gerade Anzahl Bits ergibt, ist die Aufgabenstellung nicht erfüllbar
   (C) ganzes Byte
   (D) 6 Bit
   (E) 5 Bit

---

6. Welche Funktionen erfüllt das **Monitorprogramm** im EPROM des LEMPS?

   (A) Mitteilungen auf dem LC-Display des LEMPS anzeigen
   (B) Kommunikation des PC mit dem LEMPS ermöglichen, Terminalkommandos interpretieren
   (C) Es zeigt dem Programmierer allfällige Assemblerfehler
   (D) Port-LEDs ansteuern
   (E) Es überwacht die Controllerfunktionen

---

7. Was sollten Sie bei der praktischen **Handhabung**, Verarbeitung, Montage etc. von Elektronikkomponenten und -Baugruppen besonders beachten?

   (A) Handschuhe tragen
   (B) EMV-Problematik
   (C) ESD-Problematik
   (D) Datenverluste vermeiden
   (E) Elektrische Felder abschirmen

---

8. Welche Vorteile bietet der **Assembler-Quelltext** gegenüber der direkten Programmcode-Eingabe in Hex-Werten ins RAM des Zielsystems?

   (A) Ich brauche mich nicht mit der Assemblersprache zu befassen
   (B) Der Quelltext ist lesbar, speicherbar und änderbar
   (C) Der Assembler-Quelltext ist prozessor-unabhängig
   (D) Der Quellext kann auch in Schwiizerdütsch aufgesetzt werden
   (E) Fehler im Quelltext werden automatisch korrigiert

---

9. Wie viele unterschiedliche **Bit-Muster** können mit den 8 LEDs am 8-Bit-Ausgangsport dargestellt werden, mit der Vorgabe, dass immer mindestens eine LED leuchten sollte (d.h. ohne das Byte mit dem Wert Null)?

   (A) 8
   (B) beliebig viele
   (C) 255
   (D) 2 hoch 8
   (E) 64k

---

10. Welche digitale **Logikverknüpfung** und welches **Maskenbyte** würden Sie anwenden, um im Akku A der CPU die vier hochwertigen Bits zu löschen (auf den logischen Wert Null setzen), ohne jedoch die vier niederwertigen Bits zu verändern?

    (A) AND-Verknüpfung, Maske: $0F
    (B) OR-Verknüpfung, Maske: %0000 1111
    (C) AND-Verknüpfung, Maske: %1000 0011
    (D) OR-Verknüpfung, Maske: $F0
    (E) AND-Verknüpfung, Maske: $FF

---

11. Folgende 2 Bytes werden über die **Logikverknüpfung OR** bitweise miteinander verknüpft:

    ```
    %0011 1010
    %1100 0010
    ```

    Wie sieht das resultierende Byte dieser Operation aus?

    (A) $0A
    (B) %0000 0010
    (C) $18
    (D) $FA
    (E) %0000 1111

---

12. Wozu dient ein **I/O-Latch-Baustein** in einem Mikroprozessorsystem grundsätzlich?

    (A) Serielle Kommunikation mit der Peripherie
    (B) Als Stromtreiber von LEDs und anderen Anzeigen
    (C) Temporäre Speicherung von digitalen Ein- oder Ausgangssignalen
    (D) Speicherung von Systemeinstellungen
    (E) Schutz vor zu hohen Eingangsspannungen an den Ports

---

13. Welche Aufgabe erfüllt das **16-Bit-Register PC** (ProgramCounter) der CPU im Mikrocontroller 68HC11?

    (A) Es steuert die Kommunikation mit dem externen PC
    (B) Es zeigt ausschliesslich auf Datenbytes, die von der CPU benötigt werden
    (C) Es enthält die Adresse von Systemdaten und steuert den Adressbus
    (D) Es enthält die nächste von der CPU benötigte Instruktion
    (E) Es enthält eine Adresse, resp. einen Zeiger auf das nächste Instruktionsbyte, das die CPU benötigt

---

14. Sie möchten mit einem konventionellen IC-Baustein der **Familie 74HCxxx** eine Hardware-Erweiterung zum LEMPS-System aufbauen. Welche Betriebsspannung verwenden Sie, um Ihren Zusatz-IC zu speisen (der dafür benötigte Strom ist vernachlässigbar)?

    (A) Die stabilisierte 5,0 VDC-Speisung vom LEMPS-Board
    (B) 15 VDC mittels Netzgerät
    (C) 3,3 VDC mittels Zusatzadapter
    (D) Die LEMPS-Hardware mit dem 68HC11 ist nicht kompatibel mit der 74HCxxx-Familie
    (E) 9 V Blockbatterie ohne Spannungsstabilisierung

---

15. Wie lautet der Hex-Wert des Byte **%1000 0011**?

(A) $13
(B) $83
(C) $8B
(D) $C3
(E) $87

---

16. Wie lautet der Dezimalwert des Byte **%0010 0101**? (Tip: zuerst Wertigkeiten der einzelnen Stellen festlegen.)

(A) 34d
(B) 35d
(C) 36d
(D) 37d
(E) 38d

---

17. Wie lautet der Binärwert der Hex-Zahl **$B07F**?

(A) 1101 0001 0111 1111
(B) 1011 0000 1110 0111
(C) 1011 0000 0111 1111
(D) 1101 1000 1011 1111
(E) 1001 0000 1001 1110

---

18. Welche **Technologie** wird nach heutigem Stand in erster Linie zur Fertigung von Mikroprozessoren und Speichern verwendet?

(A) TTL
(B) LS-TTL
(C) CMOS
(D) NMOS
(E) 3,3TTL

---

19. Welche der folgenden Informationen führt der **Adressbus** in einem Mikroprozessorsystem oder in einem Mikrocontrollersystem im erweiterten Modus (wie LEMPS), das mit externem ROM und RAM arbeitet?

(A) Adressen von Speicher-ICs des Systems
(B) Zeiger auf Bytes im Programmspeicher, Datenspeicher, I/O-Bereich
(C) Nur Adressen von Datenbytes
(D) Nur Adressen von Programmbytes
(E) Zeiger auf CPU-Register

---

20. Der bidirektionale **Databus** in einem Mikroprozessorsystem oder in einem Mikrocontrollersystem mit externem ROM und RAM, dient dem Transfer von Daten- und Programmbytes zwischen

(A) dem Programmierer-PC und dem Mikrocontrollersystem
(B) den I/O-Ports und den Schaltern oder LEDs die daran angeschlossen sind
(C) dem Daten- und dem Programmspeicher
(D) der CPU und den Speicher- und sonstigen Peripherieblöcken
(E) der RS 232-Schnittstelle und der CPU

Antwortblatt

Kurs EM, Zwischentest Nr. 1

Name		Vorname	
Datum	Kursort	Fachlehrer	Erreichte Punkte

Richtige Lösungen sind mit gut sichtbarem **X** zu markieren.
Bei Irrtum bitte das **X** durchkreuzen (+) oder radieren.

Aufgabe	Lösungen	
1	A B C D E	
2	A B C D E	
3	A B C D E	
4	A B C D E	
5	A B C D E	
6	A B C D E	
7	A B C D E	
8	A B C D E	
9	A B C D E	
10	A B C D E	
11	A B C D E	
12	A B C D E	
13	A B C D E	
14	A B C D E	
15	A B C D E	
16	A B C D E	
17	A B C D E	
18	A B C D E	
19	A B C D E	
20	A B C D E	

Kurs EM, Zwischentest Nr. 2

Anzahl Aufgaben	**25**	Bewertung	**4 Punkte pro Aufgabe**
Richtzeit	**90 Minuten**	Total	**100 Punkte**
Name		Vorname	
Datum	Kursort	Fachlehrer	Erreichte Punkte

1. Nennen Sie 2 Beispiele von Geräten, resp. Einrichtungen, für die Sie Mikrocontrollerbausteine zur Steuerung oder Ueberwachung einsetzen würden (sofern Sie der bisherige Kurs noch nicht davon abgeschreckt hat).

   1) .....

   2) .....

2. Aus welchen Baugruppen besteht typischerweise ein mikrocontroller-unterstütztes Gerät mit Anwender- und Datenschnittstellen (bitte 4 bis 5 Baugruppen angeben)?

   .....

   .....

3. Für eine Steuerung möchten Sie ein bestehendes Mikrocontrollersystem (wie z.B. LEMPS) einsetzen. Mit welchen 5 Schritten gelangen Sie von der Idee bis zum fertigen Programmcode, den Sie im RAM des Zielsystems austesten können?

   1) .....

   2) .....

   3) .....

   4) .....

   5) .....

4. Welche Funktion erfüllt die Pufferbatterie auf der Mikrocontrollerkarte LEMPS?

   .....

5. Warum genügt für den Aufbau des Adressbus eines Mikrocontrollersystems (z.B. LEMPS) eine unidirektionale Busstruktur?

   .....

   .....

6. Ergänzen Sie bitte in der folgenden Tabelle die Spannungspegel der 5V-CMOS-Familie:

Signalpegel	*5V-TTL-Familie*	*5V-CMOS-Familie*	*Allgemein*
L (Low / Lo / 0)	0 bis 0,7 V	.....	Phys. Wert niedrig
H (High / Hi / 1)	2,4 V bis 5 V	.....	Phys. Wert hoch

7. Beschreiben Sie kurz die 2 bis 3 wichtigsten Vorgänge, die beim Assemblieren/Compilieren (ohne Download) eines einfachen Quelltextes mit dem HC11-Assembler in der LEMPSwin-Umgebung ablaufen.

.....

.....

.....

8. Folgende Zeilen zeigen einen Ausschnitt aus dem Assembler-Quelltext des Licht-Timers.
Sie würden gerne die Funktion dieses Programmteiles nachvollziehen, welche Hilfsspalte mit Inhalt vermissen Sie dabei?

.....

Zeile	Label	Mnemonic	Operand	?
	...			
1	getSwitch	tst	PORTG	
2		beq	getSwitch	
3	timer	com	PORTA	
4		ldx	#3000	
5		jsr	delayXms	
6		clr	PORTA	
7		bra	getSwitch	
	...			

9. Welche Zeile würden Sie im obenstehenden Quelltext am ehesten manipulieren, um die Timerzeit versuchsweise zu ändern?

Zeilen-Nr.: .....

10. Welchen Hex-Wert schreiben Sie ins DDRH-Register um den Port H des HC11 nach dem folgenden Muster zu konfigurieren?

```
PortH-Bit:  7  6  5  4  3  2  1  0
Konfig:     A  E  E  E  E  E  E  A    (A=Ausgang, E=Eingang)
```

Konfigurations-Byte (Hex): .....

11. Nennen Sie 2 Unterprogramme aus der Sammlung der LEMPS-Monitor-Bibliothek (EPROM), die Ihnen als Programmierer zur Verfügung stehen.

1) *2)*

12. Nach welchen der folgenden HC11-Instruktionen kann im Progammablauf sinnvollerweise eine Verzweigung auf Null mit den Instruktionen BEQ und BNE folgen (alle zutreffenden Mnemonic-Bezeichnungen angeben)?

Einige HC11-Instruktionen (Zyklen in µ-Sek für LEMPS mit 4 MHz-Quarz):

Mnemonic	*Operation*	*Beschreibung*	*Adress-Art*	*Code (Hex)*	*Zyklen (µ-Sek)*	*CCR-Flag* **Zero**	*CCR-Flag* **Carry**
CMPA	Compare A to Memory	A - M rechnen und Flags setzen	EXT	B1 hh ll	4	>	>
INCA	Increment A	Inhalt von Akku A um 1 erhöhen	INH	4C	2	>	/
JSR	Jump to Subroutine	Unterprogramm bei Adr. hh ll ausführen	EXT	BD hh ll	6	/	/
LDAA	Load Accu A	Lade A mit unmittelbarem Wert	IMM	86 ii	2	>	/
NOP	No Operation	Keine Aktivitäten	INH	01	2	/	/
STAA	Store Accu A	Speichere A nach Adr. hh ll	EXT	B7 hh ll	4	>	/

Legende: / Flag wird durch die Operation nicht beeinflusst
> Flagzustand entsprechend der Operation

13. Wie viele NOP-Zeilen (NOP: siehe HC11-Instruktionen oben) müssen nach der Zeile 4 eingefügt werden, damit der LED-Ein-Puls an Port A mindestens 10 Mikrosekunden lang wird?

Anzahl NOP-Zeilen nach Zeile 4:

Zeile	Label	Mnemonic	Operand	Kommentar
1	init	clr	PORTA	alle LEDs aus
2	puls	ldaa	#$ff	
3		staa	PORTA	alle LEDs ein
4		nop		warten
		...		
		clr	PORTA	LEDs wieder aus

14. Wo (im Speicher) holt sich die HC11-CPU den Operanden (Arbeitswert, z.B. Byte-Variable oder Konstante) eines Befehls, dessen Adressierungsart (Addressing Mode)
a) **EXT** (z.B. `ldaa VarName`) sowie b) **IMM** (z.B. `ldaa #Wert`) lautet?

a) EXTern:

b) IMMediate:

15. Versehen Sie bitte folgendes Programm mit Kommentaren (Stichworte auf punktierte Linien).
Achtung: Beschreiben Sie bitte nicht einfach die Funktion der einzelnen Befehle, sondern deren Funktion bezogen auf die Funktion des ganzen Programmes!

```
* Programm: ..........................................................
*...................................................................
****.................................*************************************
PORTA     equ    $1000   ..............................................
DDRA      equ    $1001   ..............................................
PORTG     equ    $1002   ..............................................
DDRG      equ    $1003   ..............................................
          org    $0000   ..............................................
****.................................*************************************
initial   ldaa   #$FF    ..............................................
          staa   DDRA    ..............................................
          ldaa   #$00    ..............................................
          staa   DDRG    ..............................................
****.................................*************************************
endlos    ldaa   PORTG   ..............................................
          staa   PORTA   ..............................................
          staa   svar    Byte sichern in der Variablen svar
          jmp    endlos  ..............................................
****.................................*************************************
svar      rmb    1       1-Byte-Variable reservieren
          end            ..............................................
```

16. Zu obigem Quelltext:
Die Ausdrücke der ersten Spalte werden als Label oder symbolische Namen bezeichnet.
Das Label "PORTA" beispielsweise steht stellvertretend für den Wert $1000, der ihm mit dem Befehl EQU zugeordnet wird.
Welche Art von Werten oder Grössen werden den Labels "initial", "endlos" und "svar" beim Assemblieren zugewiesen?

.....

17. Knacknuss zu obigem Quelltext:
Welche Adressierungsart (Addressing Mode) wird der HC11-Assembler voraussichtlich auf den Speicherbefehl `staa svar` anwenden? (Tip: Assemblierung geschieht nach org $0000)

18. **ORG** und **INCL** sind Assembler-Direktiven (-Anweisungen), Instruktionen also, die **nur** für den Assembler bestimmt sind. Nennen Sie 2 weitere Assembler-Direktiven und beschreiben Sie kurz deren Funktion und Wirkung.

1)

2)

19. Was bewirken die Quelltext-Zeilen a) und b) am Anfang eines LEMPS-Assembler-Programms?

```
****  Programmanfang
              incl      myinc.inc          a)
              org       $2000              b)
              ...
              incl      disp_inc.asc       c)
```

a)

b)

c) Subroutinen-Bibliothek für die LCD-Ansteuerung einfügen

20. Zu obiger Aufgabe: Warum muss die Zeile c) mit dem INCL-Befehl **nach** dem ORG-Befehl stehen?

21. Mit dem HC11-Befehl **jsr SubName** wird eine Subroutine (Unterprogramm) mit dem Namen SubName aus dem Hauptprogramm heraus aufgerufen, abgearbeitet und danach wieder zurück ins Hauptprogramm gesprungen.
a) Beschreiben Sie kurz den Ablauf dieses Aufrufs (Hinsprung, Rücksprung) unter Berücksichtigung des Stack-Speichers, sowie des **P**rogram**C**ounter-Registers der CPU.
b) Wie heisst der HC11-Befehl zum Verlassen der Subroutine?

a)

b)

22. LEMPS-Speicheraufteilung: Programm-**Variablen** müssen mittels der Variablen-Deklaration immer im .....-Bereich des Speichers platziert werden.

23. Für einen Schreib- oder Lesezugriff der CPU auf die Speicherbausteine, werden diese zuerst über gewisse Adressleitungen selektiert und freigegeben.
a) Mit Hilfe von welchem Adressbit (A?) wird die Adressdecodierung und Selektion des RAM-Bausteines 62256 der LEMPS-Hardware durchgeführt?
b) Welcher Bereich von Speicheradressen (in Hex) wird durch den RAM-Baustein des LEMPS belegt?

a) b)

24. Der Controller 68HC11 des LEMPS wird durch einen Quarzoszillator der mit 4 MHz schwingt, getaktet.
a) Welche Frequenz besitzt der E-Clock der am entspr. Pin des HC11 auf dem LEMPS ansteht?
b) Welche Steuersignale werden hauptsächlich vom E-Clock abgeleitet (über eine Logikverknüpfung)?

a)
b)

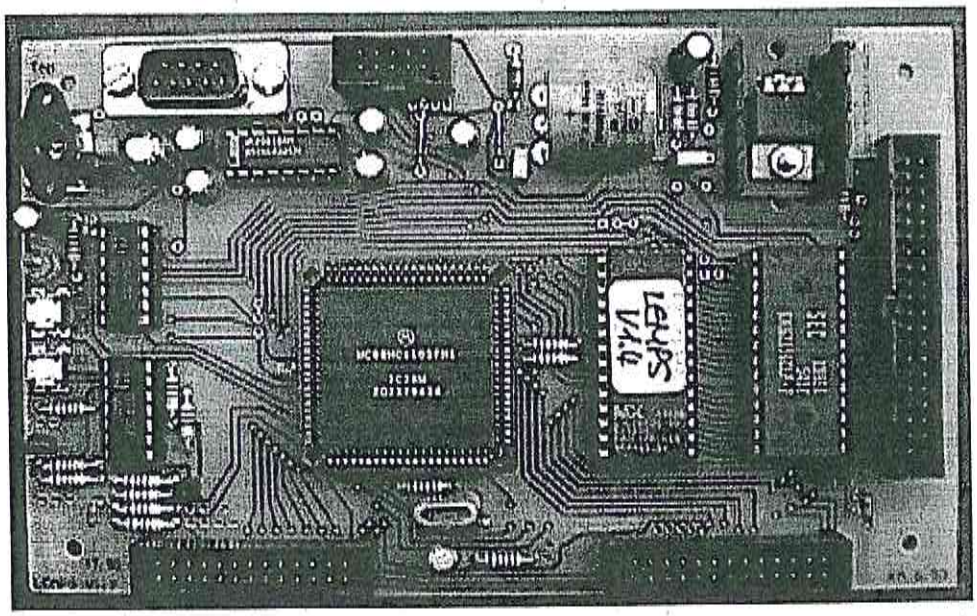
25. Sie stehen vor der dankbaren Aufgabe, eine defekte Mikrocontrollerkarte ohne die Hilfe eines Diagnose- und Testsystems wieder instand setzen zu müssen. Glücklicherweise haben Sie Zugang zu allen Hauptkomponenten auf der Platine, wie Controller-IC, EPROM-Baustein, RAM-Baustein etc. Ein Schema, Test-EPROM, Testadapter, sowie Voltmeter und Oszillograf (KO) stehen zur Verfügung.
In welcher Reihenfolge führen Sie die folgenden Schritte zur Fehlerlokalisierung am sinnvollsten durch (Nummern einsetzen, die Schritte 1...5 sind bereits vorgegeben)?

( ) Clockfrequenz- und RESET-Signale an den dafür vorgesehenen Check-Punkten mit KO untersuchen
(1) Visuelle Prüfung auf mechanische Schäden oder verbrannte Bauteile
( ) Wird der EPROM-Baustein angesprochen? Entsprechende Signale mit KO überprüfen.
(4) Speisung einschalten, Stromaufnahme kontrollieren
( ) Sind auf den Leitungen des Adress-, Daten- und Kontrollbus eindeutige Signalwechsel mit sauberen Flanken und richtigen Spannungspegeln mit dem KO sichtbar?
(2) Test-EPROM mit Diagnose-Programm einstecken
( ) RAM-Puffer-Batterie überprüfen
(5) Diagnose-Programm im EPROM starten: die Monitor-LEDs im Testadapter weisen leider auf keinen eindeutigen Fehler hin.
( ) Wird der RAM-Baustein angesprochen? Entsprechende Signale mit KO überprüfen.
(3) Mikrocontroller-Karte in Testadapter einstecken
( ) Sie sind mit Ihrem Latein am Ende: jetzt muss der Spezialist oder eine neue Kontroller-Karte her!
( ) Speisespannung an allen IC-Bausteinen messen

Einführung in die

MICROCONTROLLER-TECHNIK

mit dem Learning Microprocessor System



Version: Januar 1998

Verfasser und Copyright:

Bruno Wamister, Berufsschullehrer
Churzrüti,
3664 **Burgistein-Dorf**

E-Mail: b_wamister@bluewin.ch
Homepage: http://www.datacomm.ch/wamister

Zusammenfassung:

'Einführung in die Microcontrollertechnik' ist ein Unterrichtshilfsmittel, welches zusammen mit dem Microcontrollersystem LEMPS auf der Stufe Berufs- und Technikerschule eingesetzt werden kann. Grundlagenkenntnisse in Mathematik (Zahlensysteme) und Digitaltechnik (Grundfunktionen) werden vorausgesetzt. Schrittweise wird der Schüler mit den Möglichkeiten moderner Controller vertraut gemacht. Zu jedem Abschnitt gibt es verschiedene Beispiele, Übungen und Programmieraufgaben.

In einem einführenden Teil werden die Hardwaremöglichkeiten moderner Microcontroller kurz beschrieben und die Funktionsweise eines Bus-Systems für den Betrieb von externen Speicherbausteinen aufgezeigt. Anhand von Beispielen wird dargestellt, wie ein Programm mit einer Codiertabelle codiert wird und wie ein Programm mit einem Assembler auf einem PC compiliert werden kann. Befehle aus dem Befehlssatz des 68HC11 werden beschrieben und in Beispielen angewendet. Am Beispiel des Analog-Digitalwandlers und des Pulsweitenmodulators werden die Funktionsweise und die Programmierung der Controllerfunktionen veranschaulicht. Das Timersystem des Controllers 68HC11 wird diskutiert und anhand von Beispielen erläutert. Der Hardwareaufbau des LEMPS sowie die wichtigsten Eigenschaften der LEMPS-Software werden im letzten Teil besprochen. Im Anhang befinden sich neben wichtigen Hilfsmitteln zur Programmierung auch Aufgabenstellungen für kleinere Projektarbeiten mit dem LEMPS. Ergänzende Angaben und Hinweise zum LEMPS befinden sich im Menue 'Manual' der LEMPSWIN-Entwicklungsumgebung.

Dank:

Allen meinen Kollegen, die mich bei der Entwicklung und Einführung des LEMPS unterstützt und motiviert haben, danke ich für ihre konstruktive Mitarbeit. Das LEMPS wäre nie entstanden, wenn nicht laufend initiative Schüler meiner Klassen an diesem Projekt mitgearbeitet hätten.

Speziell bedanken möchte ich mich bei folgenden Schülern, die das LEMPS in ihrer Freizeit entwickelt haben:

Ruben Reusser, Karl Albrecht, Alexandra Duppenthaler, Peter Heimann, Christof Kuhn, Dänu Buchs, Martin Stettler, Daniel Wenger, Felix Aeschlimann, Mario Däpp.

Burgistein, Januar 1998
Bruno Wamister

sfb Bildungszentrum
Bernstrasse 394
8953 Dietikon

Tel. 0848 80 00 84
Fax 01 744 45 00
Internet www.sfb.ch
E-Mail info@sfb.ch

esg Centre de formation
Chemin des Croisettes 26
1066 Epalinges

Tél. 021 654 01 54
Fax 021 654 01 59
Internet www.esg.ch
E-Mail info@esg.ch

soa Centro di formazione
Corso Elvezia 16
6901 Lugano

Tel. 091 923 50 54
Fax 091 923 46 36
Internet www.soa.ch
E-Mail info@soa.ch