Der NDR-Computer

Kurzbeschreibung

zum

Aufbausystem

Inhalt

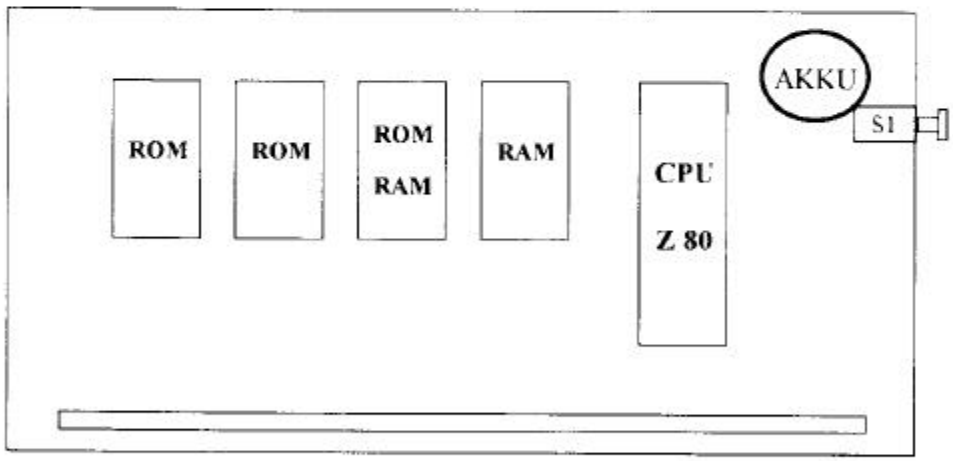
2. Aufbau eines Mikrocomputers

Ein Mikrocomputer besteht aus verschiedenen Einzelbaugruppen. Der Kern eines jeden Mikrocomputers ist der Mikroprozessor, in unserem Fall der Z80 Prozessor. Der Mikroprozessor ist über Leitungen mit den anderen Baugruppen des Mikrocomputers verbunden. Diese Leitungsverbindungen bezeichnet man als Bussystem. Die Baugruppen, mit denen unsere Prozessorkarte verbunden ist, sind:

- Die Ein- und Ausgabeeinheiten (Tastatur, Monitor, Drucker)
- Die Grafik-Karte
- Die Key-Controler-Karte
- Die I/O-Karte

2.1 Die SBC3

Zu Ihrem Aufbaugerät gehört die Baugruppe SBC3, wobei SBC für Single-Bord-Computer (Einplatinenrechner) steht, d.h. die wichtigsten Funktionseinheiten des Mirkrocomputers befinden sich auf dieser Platine. In unserem Fall sind CPU (Central Processing Unit), Speicher (RAM, ROM) und Akku auf der SBC3-Platine. Das Bussystem wird wie beim Einsteigerpaket auf einer Stiftleiste herausgeführt. Alle anderen Baugruppen befinden sich auf separaten Platinen. Um Ihnen einen Überblick zu geben, wo sich die einzelnen Bausteine, z.B. der Resetschalter S1, auf der Platine befinden, betrachten Sie folgende Skizze.



Hinweis: Die SBC3-Baugruppe ist hardwaremäßig identisch zu der SBC3-Baugruppe des Grundgerätes, jedoch ist der Speicherinhalt des EPROM's (Monitorprogramm) unterschiedlich.

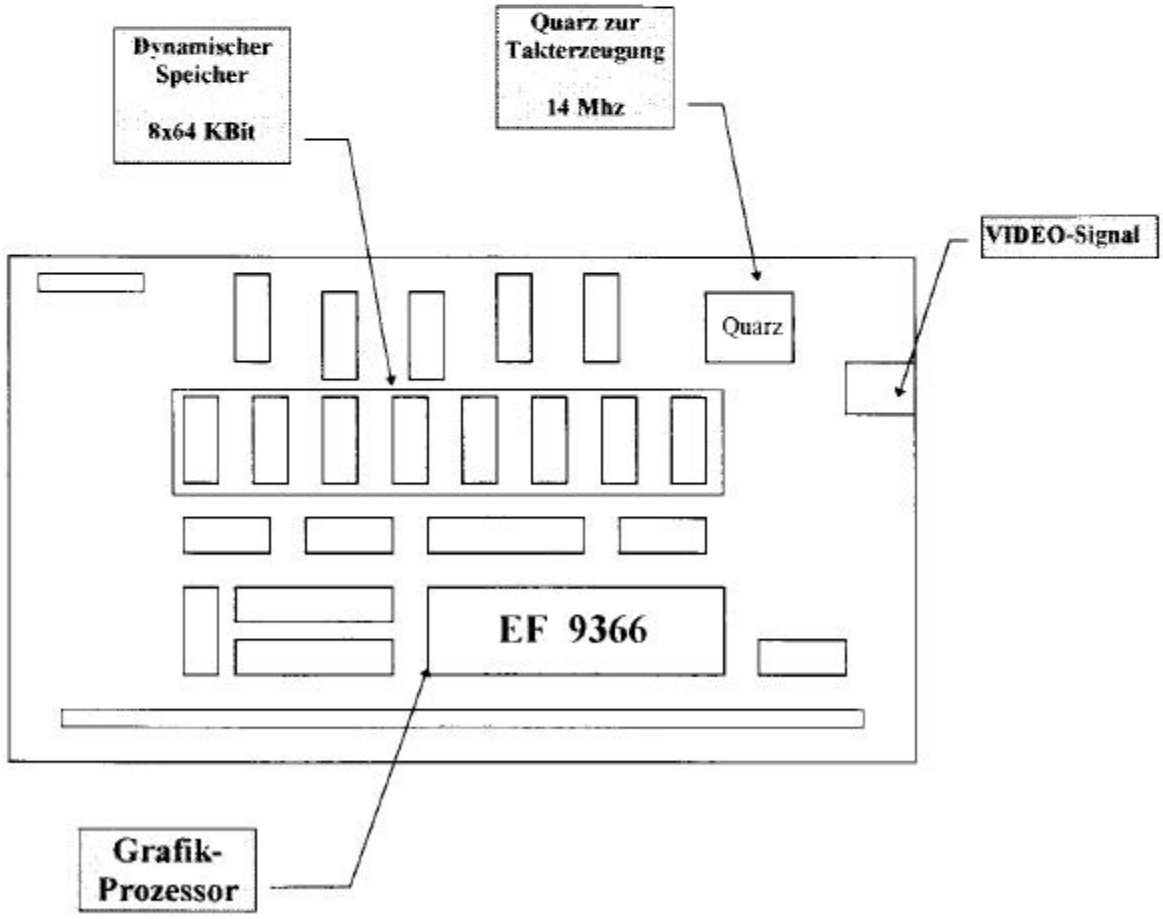
2.2 Die GDP64K

Die GDP64k ist das Bindeglied zwischen dem Mikrocomputer (SBC3, CPU Z80) und einem Monitor. Sie ermöglicht es einen Monitor mit BAS-Anschluß (HA, VS, VIDEO-Signal) oder einen TTL-Monitor anzuschließen. Nun ist es möglich Arbeitsschritte, die der Computer durchführt, auf dem Monitor anzuzeigen, Graphiken darzustellen oder Einblick in das Innenleben des Computers zu bekommen (Speicherbelegung, Kontrolle der Eingaben).

Jeder Bildpunkt auf dem Monitor muß ansprechbar sein. Bei einer Bildebene von 256 x 512 Bildpunkten wird dazu ein eigener Speicher von 16 Kbyte benötigt, wenn jeder Bildpunkt ein Bit beansprucht. Da aber vier unabhängige Bildebenen aufgebaut werden können, braucht man demnach einen Speicherplatz von 64 Kbyte. Dieser ist in 8 x 64 KBit Speichern organisiert.

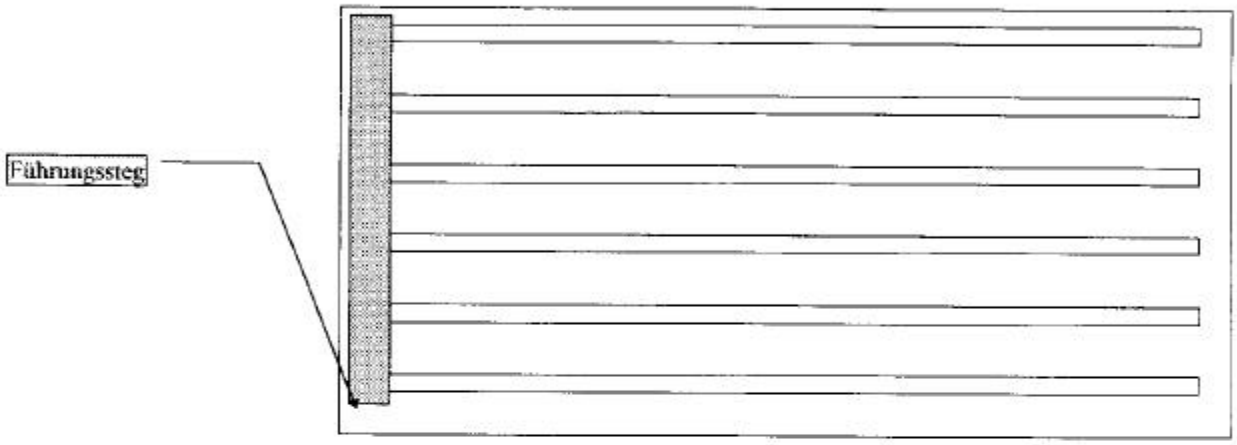
In diesem Speicher wird jeweils das gesamte Bild abgespeichert und seriell alle 20ms abgerufen (50 mal in der Sekunde); dadurch ensteht ein stehendes Bild. Die Verwaltung des Speicherbereiches übenimmt der auf der GDP 64K befindliche Graphik-Prozessor EF 9366.

Der EF 9366 ermöglicht es, Zeichen zu schreiben, Größe und Position der Zeichen zu bestimmen, schnelle Graphiken darzustellen (Blockgraphik und Vektoren), auf einen komplett intergierten ASCII-Zeichensatz zuzugreifen und viele Funtionen mehr auszuführen. Er wird vom Z80-Prozessor wie ein Ein- Ausgabebaustein angesprochen.



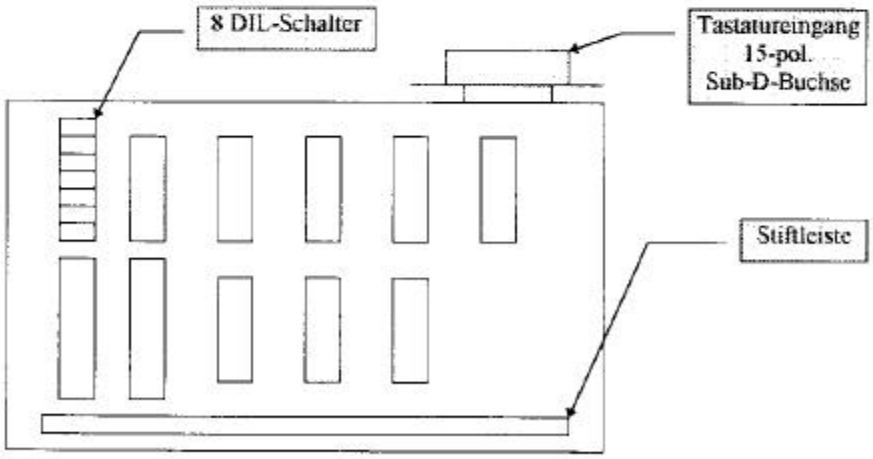
2.3 Die BUS2

Diese Platine stellt das eigentliche Bussystem dar. Auf der Platine befinden sich lediglich die Buchsenleisten, in die dann die einzelnen Baugruppen (Platinen) gesteckt werden. Das Bussystem umfaßt 54 parallel laufende Leitungen, darunter befindet sich die Spannungsversorgung, Datenbus, Adressbus und Steuerbus. Alle Platinen werden linksbündig eingesteckt, der an der linken Seite angebrachte Führungssteg verhindert ein falsches Einsetzen der Platinen.



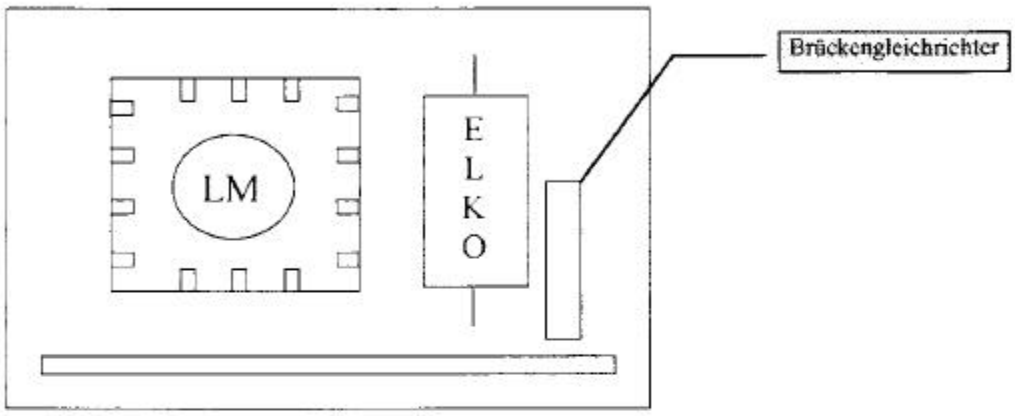
2.4 Die KEY

Die Baugruppe KEY ist die Tastaturschnittstelle des NDR-Computers. An ihr kann eine parallele sieben Bit ASCII-Tastatur angeschlossen werden. Das Strobe- und die Datensignale die die Tastatur liefert, können auf der Baugruppe invertiert werden, so daß auch Tastaturen mit neg. Strobe-Signal und invertierten Datensignalen angeschloßen werden können. Dies kann mit dem 8-poligen Dil-Schalter erfolgen.



2.5 Die POW 5V

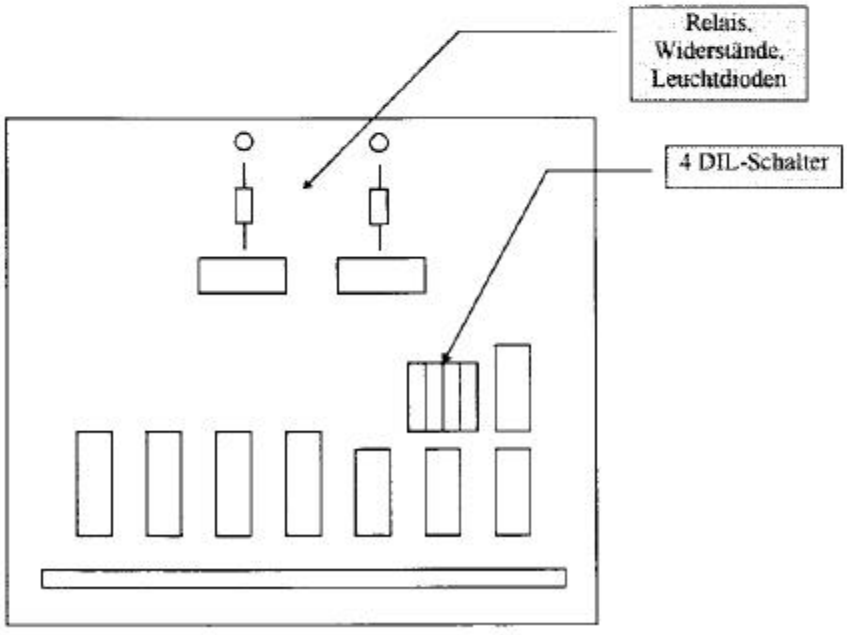
Die POW 5V stellt die Strom- und Spannungsversorgung für den Z80 und alle anderen Baugruppen, die auf der BUS2 Platine stecken, dar.



2.6 Die IOE

Die IOE-Baugruppe dient als Schnittstelle zur Außenwelt. Über die IOE-Baugruppe läßt sich z.B. eine Modelleisenbahn oder eine Heizungsanlage steuern und regeln. Die Karte verfügt über je zwei 8-Bit breite Ein- und Ausgabeports.

Den Aufbau Ihrer IOE-Karte entnehmen sie bitte dem Handbuch zur IOE



3. Das RDK-Grundprogramm

Nach dem Einschalten des Monitors und der Z80-Baugruppe, erscheint das RDK-Grundprogramm (Rolf-Dieter Klein). Das RDK Grundprogramm arbeitet prinzipiell genauso wie das Grundprogramm vom Einsteigerpaket; es ist jedoch wesentlich komfortabler und leistungsfähiger. Das RDK-G. bietet dem Anwender folgende Menus an:

- „**aendern**", über diese Prozedur lassen sich alle Speicherbereiche ansehen, ändern oder neue Programme eingeben.
- „**starten**", duch Eingabe der Startadresse lassen sich Anwenderprogramme starten, z.B. die mitgelieferten Programme Basic, Gosi und der Debugger.
- „**ansehen**", ermöglicht das Ansehen von Speicherbereichen.
- „**Symbole**", gibt alle bereits definierten Symbole aus. Symbole können Sprungmarken (Labels) oder Namen sein.
- „**weiter**", ruft das nächste Menü auf.

Durch Aufrufen des Punktes w gelangen Sie in das Menü A:

- „**Laden von CAS**", lädt Programme oder Dateien von Kassette.
- „**Speichern CAS**", speichert Programme oder Dateien auf Kassette.
- „**Pruefen CAS**", vergleicht die aus Kassette gespeicherten Daten mit den Daten im Speicher.
- „**weiter**", ruft das nächste Menü auf.

Das Speichern und Laden von Programmen auf bzw. von einer Kassette ist in diesem Gerät nicht möglich, da die CAS-Baugruppe fehlt. Doch der Akku gewährleistet den Erhalt der Daten für etwa ein ¾ Jahr.

Durch Aufrufen des Punktes w gelangen Sie in das Menü B.

- „**IO-lesen**", ließt von Portbausteinen Daten, diese werden auf dem Monitor als Hex-Zahlen dargestellt
- „**IO-setzen**", gibt Daten in Form von Hex-Zahlen an Portbausteine aus.
- „**Einzelschritt**", Programme werden Befehlsweise abgearbeitet, es werden aber keine Unterprogramme durchlaufen.
- „**weiter**", ruft das nächste Menü auf.

Durch Aufrufen des Punktes w gelangen Sie in das Menü C.

- **Eprom prog** und **Eprom lesen**, hier können Eproms gelesen oder beschrieben werden allerdings ist hierfür eine PROMMER-Baugruppe nötig.

Durch Eingabe von „3" oder „w" gelangen Sie in das Grundmenü zurück.

3.1 Programmieren mit dem RDKG

Das RDKG bietet Ihnen eineVielzahl von Programmierhilfen. Beim Grundgerät mußten Sie die Hex Codes eingeben, durch das RDKG wird die Eingabe von Programmen wesentlich vereinfacht. Anhand des nachfolgenden Programmbeispiels soll dies deutlich weden.

- Rufen Sie im Grundmenü den Punkt 1-**aendern** auf.
- Geben sie die Adresse 9000h (h steht für Hex und braucht nicht eingegeben werden)
- Geben die nachfolgendes Programm ein.

Eingabe:	Speicherinhalt:
kreis:=$	9000: 21 68 01
21#360 w	9003: CD 0F 00 : CD SCHLEIFE
cd schleife	9006: 21 02 00
21#2 w	9009: CD 03 00 : CD SCHREITE
cd schreite	900C: 21 01 00
21#1 w	900F: CD 06 00 : CD DREHE
cd drehe	9012: CD 12 00 : CD ENDSCHLEIFE
cd endschleife	9015: C9
c9	

Das RDKG hat unsere Eingabe verstanden und in Hexzahlen umgewandelt. Rechts neben den Hexzahlen stehen die Unterprogramm-Routinen des RDKG, die wir in unserem Programm verwendet haben.

Kehren Sie zum Grundmenü zurück und rufen Sie Punkt 4-Symbole auf.

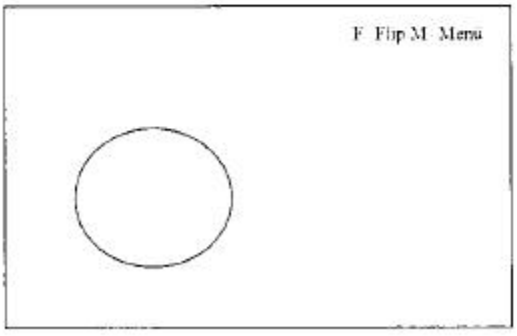
Es sollte das Symbol:

9000 Kreis

erscheinen.

Starten Sie das Programm, durch Grundmenüpunkt „2-starten" und Eingabe der Adresse oder des Labels Kreis.

Ihr Bildschirm sollte nun so aussehen:



3.2 Der Debugger V2.1

Als Debugger bezeichnet man Programme, die eine Fehlersuche in Programmen ermöglichen. Der Debugger V2.1 des NDR-Computers dient aber hauptsächlich der Programmierung in Assemblersprache. Er verfügt über einen Assembler, mit dem man Programme in mnemotechnischer Assenblerschreibweise in den Rechner eingeben kann. Der Assembler übersetzt die Mnemonik in das entsprechende Bitmuster für den Z80. Außer dem Assembler verfügt der Debugger V2.1 über einen Disassembler. Der Disassembler ermöglicht eine Übersetzung der Bitmuster von den einzelnen Befehlen in die mnemotechnische Schreibweise. Er arbeitet also genau entgegengesetzt dem Assembler.

Gestartet wird er duch Grundmenüpunkt „2-starten" mit der Adresse 6000h (h steht für Hex und baucht nicht eingegeben werden).

Das erscheinende Menü bietet folgende Möglichkeiten:

- 1-Assembler, ermöglicht das zeilenweise Eingeben von Programmen in der Mnemotechnischen Schreibweise.
- 2-Disassembler, übersetzt Maschinenprogramme in mnemotechische Schreibweise des Z80.
- 3-Starten, entspricht dem RDKG.
- 4-Einzelschritt, entspricht dem RDKG.
- 5-Protokoll an/aus, ermöglicht das Ausdrucken aller Aktivitäten des Disassemblers und des Assemblers.
- 6-Grundprogramm, kehrt ins RDKG zurück.

3.3 Die Programmiersprache GOSI

Gosi ist die Abkürzung von **G**rafisch **O**rientierter **S**prache **I**, wird aber wie „Gosi" gesprochen. Sie besitzt Sprachelemente für einen Schreibzeiger. Der Schreibzeiger kommt auch in dem RDKG vor, heißt dort aber Turtle (Schildkröte). Gosi ist leicht erlernbar. Das Prinzip der Sprache beruht darauf, neue Sprachelemente zu entwickeln. Diese neuen Sprachelemente gehen quasi in den Befehlssatz von GOSI über.
Gestartet wird GOSI über das RDKG-Hauptmenü „starten" und Speicheradresse: 2000H. Um einen Eindruck zu vermitteln wie leicht die Programmierung ist, wollen wir ein kleines Beispiel progammieren. Wir wollen ein Quadrat mit der Seitenlänge 100 zeichnen, die Lage des Quadrats auf dem Bildschirm bleibt dabei ohne Bedeutung.

- Nach dem Bildschirmaufbau, sollten sie die Programmierung in Großbuchstaben vornehmen, drücken sie hierzu die ALPHA LOCK-TASTE.

Der Befehlssatz für das Beispiel lautet:
Die eckigen Klammern erhalten sie durch ALPHA-LOCK und Ä oder Ü.

WIEDERHOLE 4[Vorwaerts 100 RECHTS 90]

GOSI verfügt aber auch über eine Kurzschreibweise, diese lautet:

WH 4[VW 100 RE 90]

Der Zeiger hat ein Quadrat gezeichnet und steht wieder auf der Ausgangsposition. Wie Sie sehen, ist das Zeichnen mir GOSI nicht schwer und leicht erlernbar. Mehr dazu in Ihrem Handbuch.

3.4 Die Programmiersprache BASIC

BASIC steht für „**B**eginners **A**ll Purpose **S**ymbolic **I**nstruction **C**ode" ist auch sehr leicht erlernbar und stellt schon eine höhere Programmiersprache dar. BASIC unterliegt großen Vorurteilen, so sagt man die Sprache sei nur für den Hobbyberich brauchbar. Dies ist aber ein Irrtum, denn durch strukturiertes Programmieren reicht es weit über die Bedürfnisse von Anfängern hinaus.

Die im NDR-Computer gespeicherte Basic-Version entspricht weitestgehend dem Microsoft-Standard. Es wird ebenfalls duch das RDKG „gestartet" unter Startadresse: 4000h

Nach dem Starten erscheint ein Fragezeichen am linken oberen Bildschirmrand, der Basic-Interpreter will von Ihnen wissen ob sie einen Warm- oder Kaltstart durchführen möchten. Beim Warmstart bleiben zuvor abgespeichrte Daten erhalten, daher sollte nach dem Einschalten des Rechners der Kaltstart nur einmal durchgeführt werden.Geben Sie also ein:

C = Kaltstart
W = Warmstart

Nach drücken der Resettaste starten Sie also immer mit dem Warmstart um Ihre Programme nicht zu verlieren!

Hier nun ein kleines Beispielprogramm:

```
1 rem Gitternetz 25*25 Felder
5 clrs
10 page 0,0                    :rem Schreibseite 0
20 for I = 0 to 500 Step 20    :rem Senkrechte Linien
30 moveto I,0
40 drawto I,250
50 next
60 for I = 0 to 500 step 10    :rem Waagrechte Linien
70 moveto I,0
80 drawto I,250
90 next
100 goto 100
```

Das Programm kann mit der Tastenkombination CTRL-S angehalten und anschließend mit ESC abgebrochen werden, geben sie CLRS ein um den Bildschirm zu löschen.

Sicherlich gibt es heute wesentlich leistungsstärkere BASIC-Interpreter, aber prinzipiell lassen sich auch mit diesem alle Problemstellungen bewältigen. Hervorzuheben sind allerdings die Befehle IN und OUT, sie ermöglichen einen direkten Datenaustausch mit den Portbausteinen. Somit können Sie Steuer- und Regelungsaufgaben in der Programmiersprache Basic ausführen.

4. Ein kleines Testprogramm für den I/O-Bausatz

Nachdem Sie die Platine vollständig aufgebaut haben und Sie überprüft haben, ob alle Verbindungen korrekt angeschlossen sind, starten Sie den Debugger V2.1 und wechseln zum Assembler. Nun geben Sie folgendes Programm ein:

Adresse	Hexcode	Label	Assembler
xxxx			org 9000h
9000		Blink:	
9000	3E 00		LD A,00H
9002	D3 F0		OUT (0F0H),A
9004		Wieder:	
9004	3E 03		LD A,03H
9006	D3 F0		OUT (0F0H),A
9008	21 D0 07		LD HL,2000
900B	CD 0F 00		CALL SCHLEIFE
900E	00		NOP
900F	CD 12 00		CALL ENDSCHLEIFE
9012	3E 00		LD A,00H
9014	D3 F0		OUT (0F0H),A
9016	21 D0 07		LD HL,2000
9019	CD 0F 00		CALL SCHLEIFE
901C	00		NOP
901D	CD 12 00		CALL ENDSCHLEIFE
9020	18 E2		JR WIEDER

Verlassen Sie nun den Debugger V2.1 durch die Reset-Taste und starten Sie das Programm duch das RDKG mit der Startadresse 9000h.
Beide LED's müßten nun im Rythmus von etwa ½ Sekunde blinken.

Wie bereits besprochen, handelt es sich hier nur um eine Kurzbeschreibung, sie ist nicht geeignet, um das Programmieren zu lernen. Beim Aufbaugerät wird ein sehr guter Aufbaukurs mitgeliefert, mit ihm sollte dies kein Problem mehr sein.

Wir wünschen Ihnen aber trotzdem viel Spaß mit dem Z80-Mikroprozessor und hoffen, daß Sie nun einem kleinen Überblick der vielseitigen Anwendungsmöglichkeiten dieses Gerätes erhalten haben.

8-bit-Ladebefehle

	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(nn)	n
LD A,..	7F	78	79	7A	7B	7C	7D	7E	0A	1A	3AXXXX	3EXX
LD B,..	47	40	41	42	43	44	45	46				06XX
LD C,..	4F	48	49	4A	4B	4C	4D	4E				0EXX
LD D,..	57	50	51	52	53	54	55	56				16XX
LD E,..	5F	58	59	5A	5B	5C	5D	5E				1EXX
LD H,..	67	60	61	62	63	64	65	66				26XX
LD L,..	6F	68	69	6A	6B	6C	6D	6E				2EXX
LD (HL),..	77	70	71	72	73	74	75					36XX
LD (BC),..	02											
LD (DE),..	12											
LD (nn),..	32XXXX											

	A	B	C	D	E	H	L
LD ..(IX+d)	DD7EXX	DD46XX	DD4EXX	DD56XX	DD5EXX	DD66XX	DD6EXX
LD ..(IY+d)	FD7EXX	FD46XX	FD4EXX	FD56XX	FD5EXX	FD66XX	FD6EXX
LD (IX+d),.	DD77XX	DD70XX	DD71XX	DD72XX	DD73XX	DD74XX	DD75XX
LD (IY+d),.	FD77XX	FD70XX	FD71XX	FD72XX	FD73XX	FD74XX	FD75XX

LD (IX+d),n DD36XXXX LD (IY+d),n FD36XXXX

		S	Z	H	P/V	N	C
LD A,I	ED57	*	*	0	*	0	-
LD A,R	ED5F	*	*	0	*	0	-
LD I,A	ED47	-	-	-	-	-	-
LD R,A	ED4F	-	-	-	-	-	-

16-bit-Ladebefehle

	BC	DE	HL	SP	IX	IY
LD ..,nn	01XXXX	11XXXX	21XXXX	31XXXX	DD21XXXX	FD21XXXX
LD ..,(nn)	ED4BXXXX	ED5BXXXX	2AXXXX	ED7BXXXX	DD2AXXXX	FD2AXXXX
LD (nn),..	ED43XXXX	ED53XXXX	22XXXX	ED73XXXX	DD22XXXX	FD22XXXX
LD SP			F9		DDF9	FDF9

	BC	DE	HL	AF	IX	IY
PUSH ..	C5	D5	E5	F5	DDE5	FDE5
POP ..	C1	D1	E1	F1	DDE1	FDE1

EX (SP),HL	E3	EX DE,HL	EB
EX (SP),IX	DDE3	EX AF,AF'	08
EX (SP),IY	FDE3	EXX	D9 BC-BC' DE-DE' HL-HL'

Blocktransfer- und Suchbefehle

		S	Z	H	P/V	N	C	
LDI	EDA0	-	-	0	*	0	-	LD(DE),(HL);INC HL;INC DE;DEC BC
LDIR	EDB0	-	-	0	0	0	-	wie LDI,wiederholen bis BC=0
LDD	EDA8	-	-	0	*	0	-	LD(DE),(HL);DEC HL;DEC DE;DEC BC
LDDR	EDB8	-	-	0	0	0	-	wie LDD,wiederholen bis BC=0
CPI	EDA1	*	*	*	*	1	-	LD A,(HL);INC HL;DEC BC
CPIR	EDB1	*	*	*	*	1	-	wie CPI,wiederholen bis BC=0 oder Zeichen gefunden
CPD	EDA9	*	*	*	*	1	-	LD A,(HL);DEC HL;DEC BC
CPDR	EDB9	*	*	*	*	1	-	wie CPD,wiederholen bis BC=0 oder Zeichen gefunden

Sprungbefehle

	Z	NZ	C	NC	PE	PO	M	P
JP	CAXXXX	C2XXXX	DAXXXX	D2XXXX	EAXXXX	E2XXXX	FAXXXX	F2XXXX
CALL	CCXXXX	C4XXXX	DCXXXX	D4XXXX	ECXXXX	E4XXXX	FCXXXX	F4XXXX
RET	C8	C0	D8	D0	E8	E0	F8	F0
JR	28XX	20XX	38XX	30XX				

	unbedingt	(HL)	(IX)	(IY)
JP	C3XXXX	E9	DDE9	FDE9
CALL	CDXXXX			
RET	C9			
JR	18XX			

RST 00 08 10 18 20 28 30 38
C7 CF D7 DF E7 EF F7 FF

DJNZ 10XX DEC B;JR NZ,d
RETI ED4D zurueck vom Interrupt
RETN ED45 zurueck vom nicht maskierbaren Interrupt

CPU-Steuerbefehle

		S	Z	H	P/V	N	C	
NOP	00	-	-	-	-	-	-	Leerbefehl
HALT	76	-	-	-	-	-	-	
CCF	3F	-	-	*	-	0	*	Komplementiere Carry-Flag
SCF	37	-	-	0	-	0	1	Setze Carry-Flag
EI	FB	-	-	-	-	-	-	Interrupts freigeben
DI	F3	-	-	-	-	-	-	Interrupts sperren
IM 0	ED46	-	-	-	-	-	-	Interrupt-Modus 0
IM 1	ED56	-	-	-	-	-	-	Interrupt-Modus 1
IM 2	ED5E	-	-	-	-	-	-	Interrupt-Modus 2

Ein-/Ausgabebefehle

	A	B	C	D	E	H	L	S	Z	H	P/V	N	C
IN ..,(C)	ED78	ED40	ED48	ED50	ED58	ED60	ED68	*	*	0	*	0	-
OUT (C),.	ED79	ED41	ED49	ED51	ED59	ED61	ED69	-	-	-	-	-	-

		S	Z	H	P/V	N	C	
IN A,n	DBXX	-	-	-	-	-	-	
OUT n,A	D3XX	-	-	-	-	-	-	
INI	EDA2	?	*	?	?	1	-	IN (HL),(C);INC HL;DEC B
INIR	EDB2	?	1	?	?	1	-	wie INI,wiederholen solange B()0
IND	EDAA	?	*	?	?	1	-	IN (HL),(C);DEC HL;DEC B
INDR	EDBA	?	1	?	?	1	-	wie IND,wiederholen solange B()0
OUTI	EDA3	?	*	?	?	1	-	OUT (C),(HL);INC HL;DEC B
OTIR	EDB3	?	1	?	?	1	-	wie OUTI,wiederholen solange B()0
OUTD	EDAB	?	*	?	?	1	-	OUT (C),(HL);DEC HL;DEC B
OTDR	EDBB	?	1	?	?	1	-	wie OUTD,wiederholen solange B()0

8-bit Arithmetische und Logische Befehle

	B	C	D	E	H	L	(HL)	A	n	(IX+d)	(IY+d)	S	Z	H	P/V	N	C
ADD A,	80	81	82	83	84	85	86	87	C6XX	DD86XX	FD86XX	*	*	*	*	0	*
ADC A,	88	89	8A	8B	8C	8D	8E	8F	CEXX	DD8EXX	FD8EXX	*	*	*	*	0	*
SUB	90	91	92	93	94	95	96	97	D6XX	DD96XX	FD96XX	*	*	*	*	1	*
SBC A,	98	99	9A	9B	9C	9D	9E	9F	DEXX	DD9EXX	FD9EXX	*	*	*	*	1	*
AND	A0	A1	A2	A3	A4	A5	A6	A7	E6XX	DDA6XX	FDA6XX	*	*	1	*	0	0
XOR	A8	A9	AA	AB	AC	AD	AE	AF	EEXX	DDAEXX	FDAEXX	*	*	1	*	0	0
OR	B0	B1	B2	B3	B4	B5	B6	B7	F6XX	DDB6XX	FDB6XX	*	*	1	*	0	0
CP	B8	B9	BA	BB	BC	BD	BE	BF	FEXX	DDBEXX	FDBEXX	*	*	*	*	1	*
INC	04	0C	14	1C	24	2C	34	3C		DD34XX	FD34XX	*	*	*	*	0	-
DEC	05	0D	15	1D	25	2D	35	3D		DD35XX	FD35XX	*	*	*	*	1	-

		S	Z	H	P/V	N	C	
DAA	27	*	*	*	*	-	*	BCD-Korrektur im Akku
CPL	2F	-	-	1	-	1	-	Komplementiere Akku (1er-Komplement)
NEG	ED44	*	*	*	*	1	*	Komplementiere Akku (2er-Komplement)

16-bit Arithmetische und Logische Befehle

	BC	DE	HL	SP	IX	IY	S	Z	H	P/V	N	C
INC	03	13	23	33	DD23	FD23	-	-	-	-	-	-
DEC	0B	1B	2B	3B	DD2B	FD2B	-	-	-	-	-	-
ADD HL,.	09	19	29	39			-	-	*	-	0	*
ADC HL,.	ED4A	ED5A	ED6A	ED7A			*	*	*	*	0	*
SBC HL,.	ED42	ED52	ED62	ED72			*	*	*	*	1	*
ADD IX,.	DD09	DD19		DD39	DD29		-	-	*	-	0	*
ADD IY,.	FD09	FD19		FD39		FD29	-	-	*	-	0	*

Rotations- und Schiebebefehle

	B	C	D	E	H	L	(HL)	A	(IX+d)	(IY+d)
RR	CB18	CB19	CB1A	CB1B	CB1C	CB1D	CB1E	CB1F	DDCBXX1E	FDCBXX1E
RL	CB10	CB11	CB12	CB13	CB14	CB15	CB16	CB17	DDCBXX16	FDCBXX16
RRC	CB08	CB09	CB0A	CB0B	CB0C	CB0D	CB0E	CB0F	DDCBXX0E	FDCBXX0E
RLC	CB00	CB01	CB02	CB03	CB04	CB05	CB06	CB07	DDCBXX06	FDCBXX06
SRA	CB28	CB29	CB2A	CB2B	CB2C	CB2D	CB2E	CB2F	DDCBXX2E	FDCBXX2E
SLA	CB20	CB21	CB22	CB23	CB24	CB25	CB26	CB27	DDCBXX26	FDCBXX26
SRL	CB38	CB39	CB3A	CB3B	CB3C	CB3D	CB3E	CB3F	DDCBXX3E	FDCBXX3E

	S	Z	H	P/V	N	C	
RR/RL	*	*	0	*	0	*	Rotiere Reg rechts/links durch Carry
RRC/RLC	*	*	0	*	0	*	Rotiere Reg rechts/links
SRA/SLA	*	*	0	*	0	*	Shift Reg rechts/links arithmetisch
SRL	*	*	0	*	0	*	Shift Reg rechts/links logisch

		S	Z	H	P/V	N	C	
RRCA	0F	-	-	0	-	0	*	Rotiere Akku rechts
RLCA	07	-	-	0	-	0	*	Rotiere Akku links
RRA	1F	-	-	0	-	0	*	Rotiere Akku rechts durch Carry
RLA	17	-	-	0	-	0	*	Rotiere Akku links durch Carry
RLD(HL)	ED6F	*	*	0	*	0	-	Rotiere Ziffer links zwischen Akku und (HL)
RRD(HL)	ED67	*	*	0	*	0	-	Rotiere Ziffer rechts zwischen Akku und (HL)

Einzelbitbefehle

	B	C	D	E	H	L	(HL)	A	(IX+d)	(IY+d)
BIT 0	CB40	CB41	CB42	CB43	CB44	CB45	CB46	CB47	DDCBXX46	FDCBXX46
BIT 1	CB48	CB49	CB4A	CB4B	CB4C	CB4D	CB4E	CB4F	DDCBXX4E	FDCBXX4E
BIT 2	CB50	CB51	CB52	CB53	CB54	CB55	CB56	CB57	DDCBXX56	FDCBXX56
BIT 3	CB58	CB59	CB5A	CB5B	CB5C	CB5D	CB5E	CB5F	DDCBXX5E	FDCBXX5E
BIT 4	CB60	CB61	CB62	CB63	CB64	CB65	CB66	CB67	DDCBXX66	FDCBXX66
BIT 5	CB68	CB69	CB6A	CB6B	CB6C	CB6D	CB6E	CB6F	DDCBXX6E	FDCBXX6E
BIT 6	CB70	CB71	CB72	CB73	CB74	CB75	CB76	CB77	DDCBXX76	FDCBXX76
BIT 7	CB78	CB79	CB7A	CB7B	CB7C	CB7D	CB7E	CB7F	DDCBXX7E	FDCBXX7E
RES 0	CB80	CB81	CB82	CB83	CB84	CB85	CB86	CB87	DDCBXX86	FDCBXX86
RES 1	CB88	CB89	CB8A	CB8B	CB8C	CB8D	CB8E	CB8F	DDCBXX8E	FDCBXX8E
RES 2	CB90	CB91	CB92	CB93	CB9A	CB95	CB96	CB97	DDCBXX96	FDCBXX96
RES 3	CB98	CB99	CB9A	CB9B	CB9C	CB9D	CB9E	CB9F	DDCBXX9E	FDCBXX9E
RES 4	CBA0	CBA1	CBA2	CBA3	CBA4	CBA5	CBA6	CBA7	DDCBXXA6	FDCBXXA6
RES 5	CBA8	CBA9	CBAA	CBAB	CBAC	CBAD	CBAE	CBAF	DDCBXXAE	FDCBXXAE
RES 6	CBB0	CBB1	CBB2	CBB3	CBB4	CBB5	CBB6	CBB7	DDCBXXB6	FDCBXXB6
RES 7	CBB8	CBB9	CBBA	CBBB	CBBC	CBBD	CBBE	CBBF	DDCBXXBE	FDCBXXBE
SET 0	CBC0	CBC1	CBC2	CBC3	CBC4	CBC5	CBC6	CBC7	DDCBXXC6	FDCBXXC6
SET 1	CBC8	CBC9	CBCA	CBCB	CBCC	CBCD	CBCE	CBCF	DDCBXXCE	FDCBXXCE
SET 2	CBD0	CBD1	CBD2	CBD3	CBD4	CBD5	CBD6	CBD7	DDCBXXD6	FDCBXXD6
SET 3	CBD8	CBD9	CBDA	CBDB	CBDC	CBDD	CBDE	CBDF	DDCBXXDE	FDCBXXDE
SET 4	CBE0	CBE1	CBE2	CBE3	CBE4	CBE5	CBE6	CBE7	DDCBXXE6	FDCBXXE6
SET 5	CBE8	CBE9	CBEA	CBEB	CBEC	CBED	CBEE	CBEF	DDCBXXEE	FDCBXXEE
SET 6	CBF0	CBF1	CBF2	CBF3	CBF4	CBF5	CBF6	CBF7	DDCBXXF6	FDCBXXF6
SET 7	CBF8	CBF9	CBFA	CBFB	CBFC	CBFD	CBFE	CBFF	DDCBXXFE	FDCBXXFE

Flagbeeinfussung:

	S	Z	H	P/V	N	C
BIT	?	*	1	?	0	-
SET	-	-	-	-	-	-
RES	-	-	-	-	-	-

Flag-Register

Bit	7	6	5	4	3	2	1	0
	S	Z	X	H	X	P/V	N	C

	gesetzt	nicht gesetzt	wird gesetzt bei:
C Carry-Flag	C	NC	Uebertrag von Bit 7
N Add-/Subtract-Flag			Subtraktionen
P/V Parity-/Overflow-Flag	PE	PO	gerader Paritaet
H Half-Carry-Flag			Uebertrag von Bit 3
Z Zero-Flag	Z	NZ	Ergebnis 0
S Sign-Flag	M	P	neg. Ergebnis

X nicht verwendet

Beeinflussung:
- 1 gesetzt
- 0 zurueckgesetzt
- * abhaengig vom Ergebnis einer Operation
- - nicht beeinflusst
- ? unbestimmt