Frank Majewski

Der EMUF86

Nach dem großen Erfolg unserer bisherigen EMUFs, ist es an der Zeit, ein Familienmitglied vorzustellen, das den PC-Benutzern entgegenkommt. Hier ist es: ein EMUF mit 8086-CPU!

EMUF steht für Einplatinencomputer mit universeller Festprogrammierung. Nun wird bei Lesern, die sich schon länger mit Computern beschäftigen, der Begriff 'Einplatinencomputer' am Ende noch immer mulmige Gefühle auslösen: Zu tief sitzt vielleicht ihre Erinnerung an jene ersten Rechnerchen, bei denen jedes Byte des fast unbezahlbaren Speichers über eine kleine Tastatur und Siebensegmentanzeigen 'per Handschlag' persönlich begrüßt werden wollte. Für diese Oldtimer und die überragende Zahl der PC-Besitzer brechen erfreuliche Zeiten herein: Ein professioneller EMUF auf Basis des populären 16-Bit-Prozessors 8086 mit voller Unterstützung durch Interrupt-Controller, Zähler, bis zu 256 KByte EPROM und 128 KByte RAM auf einer ausbaufähigen Doppeleuropakarte!

Um noch kurz bei jenen 'alten Zeiten' zu bleiben: Als die freudig erregte Gemeinde der Fachleute 1981 die Gehäuse der ersten IBM-PCs öffnete, wird es doch so manches verdutzte Gesicht gegeben haben. Denn was war dort statt der vom Großrechner-Giganten erwarteten High-Tech-Granate verschraubt: ein EMUF! Und dann noch einer von der armseligen Sorte: ohne serielle oder parallele Schnittstellen, statt dessen lieber mit Kassettenrecorderanschluß, großzügigen 16 KByte festinstalliertem RAM und 8-Bit-Datenbus.

Im Gegensatz dazu braucht der neue EMUF86 nicht über ein außen angebrachtes Namensschild zu überzeugen: Er ist von Beginn an komfortabel ausgestattet und, sollte es für einen speziellen Fall dennoch einmal nicht ausreichen, leicht über einen flexiblen 16-Bit-Bus erweiterbar (*Bild 1*). Trotz des leistungsfähigen Konzepts bleibt die Hardware aufgrund der Verwendung von Standardbauteilen selbst für Großserien extrem preisgünstig! Wer schon einen Blick auf die Platine des IBM-PC bzw. seiner zahlreichen Zwillinge oder in das dazugehörige technische Handbuch riskiert hat, wird bei der Ausstattung des EMUF86 viele 'alte Bekannte' antreffen. Umgekehrt ist das Experimentieren mit dem EMUF86 auch für bislang unerfahrene PC-Besitzer sehr lohnend, da sich die gewonnenen Erfahrungen sofort auf den PC übertragen lassen. So wäre das sorglose Experimentieren mit Bausteinen im PC nicht ohne Gefahr, da diese immer zumindestens teilweise für z. B. die Steuerung von Floppy und Harddisk oder den Refresh der dynamischen RAMs programmiert sind. Beim EMUF86, der größtenteils die gleichen Bausteine verwendet, ist das vollkommen unkritisch: Der schlimmste aller Fälle wäre ein Programmabsturz, der sich durch Drücken der Reset-Taste problemlos aus der Welt schaffen läßt. Ein darüber hinausgehender Datenverlust ist ausgeschlossen!

Der Verweis auf den PC kommt aber noch aus einem anderen Grund nicht von ungefähr: Er ist mit seiner Flut an Assemblern, Debuggern, Compilern und anderen modernen Software-Tools ein ausgezeichnetes Entwicklungssystem. Der Hardware-Rahmen ist ja, wenngleich auch sehr flexibel, durch die Platine abgesteckt, aber die für jeden Einsatz zu entwickelnde Software spielt die entscheidende Rolle. Das ist schließlich auch die eigentliche Idee hinter dem EMUF-Konzept: Festverschaltete und damit unflexible TTL-'Wüsten' durch leicht anpaßbare Software zu ersetzen.

Software entwickeln mit Komfort

Beim Thema Software werden auch schon wieder einige Bedenken anmelden: „Leicht anpaßbare Programme? Assembler wird er damit wohl doch nicht meinen!" Daß dieser Einwand berechtigt ist, kann jeder ermessen, der sich bereits mit Entwicklung und Pflege eines umfangreicheren Assemblerprogramms beschäftigt hat. Doch gerade auch hier bricht eine neue Ära heran! Die Leistungsfähigkeit moderner 16-Bit-Prozessoren ist so enorm, daß man ohne weiteres etwas davon 'verschenken' darf, um dafür die gestellte Aufgabe soweit wie möglich in einer Hochsprache lösen zu können. Das sind Zeiten!

Das Konzept: Entwickeln des Steuerprogramms auf dem PC mit Screen-Editor und Hochsprachen-Compiler, Übersetzen und dann Überspielen des erzeugten Maschinencodes in das üppige RAM des EMUF, Austesten und im Fehlerfall wieder zurück in den Editor. Der EPROM-Programmierer kommt erst ganz zum Schluß ins Spiel, das Löschgerät bleibt außen vor!

Bei der Auswahl der richtigen Programmiersprache gibt es mehrere wichtige Punkte: Der erzeugte Maschinencode soll kurz und schnell, das Programm leicht auf ein anderes Zielsystem (EMUF, Rechner) übertragbar und ein passender Compiler greifbar sein. Letztendlich will man die gewählte Sprache auch beherrschen (oder schnell erlernen

Tabelle 1: Belegung der VG-Leiste

Pin-Nr.	a	c
1	+ 5 V	+ 5 V
2	D 15	D 14
3	D 13	D 12
4	D 11	D 10
5	D 9	D 8
6	D 7	D 6
7	D 5	D 4
8	D 3	D 2
9	D 1	D 0
10	PCLK	$\overline{\text{RESET}}$
11	$\overline{\text{WRL}}$	Out 1
12	A 19	A 18
13	A 17	A 16
14	A 15	A 14
15	A 13	A 12
16	M/$\overline{\text{IO}}$	Out 2
17	A 11	A 10
18	A 9	A 8
19	A 7	A 6
20	A 5	A 4
21	A 3	A 2
22	A 1	A 0
23	Clk 1	NMI
24	$\overline{\text{IR 6}}$	$\overline{\text{IR 7}}$
25	Out 0	$\overline{\text{IR 5}}$
26	$\overline{\text{RD}}$	$\overline{\text{WRH}}$
27	$\overline{\text{INTA}}$	$\overline{\text{WR}}$
28	Clk 0	Clk 2
29	Clk	$\overline{\text{BHE}}$
30	Gate 0	INTR
31	Gate 2	Gate 1
32	GND	GND

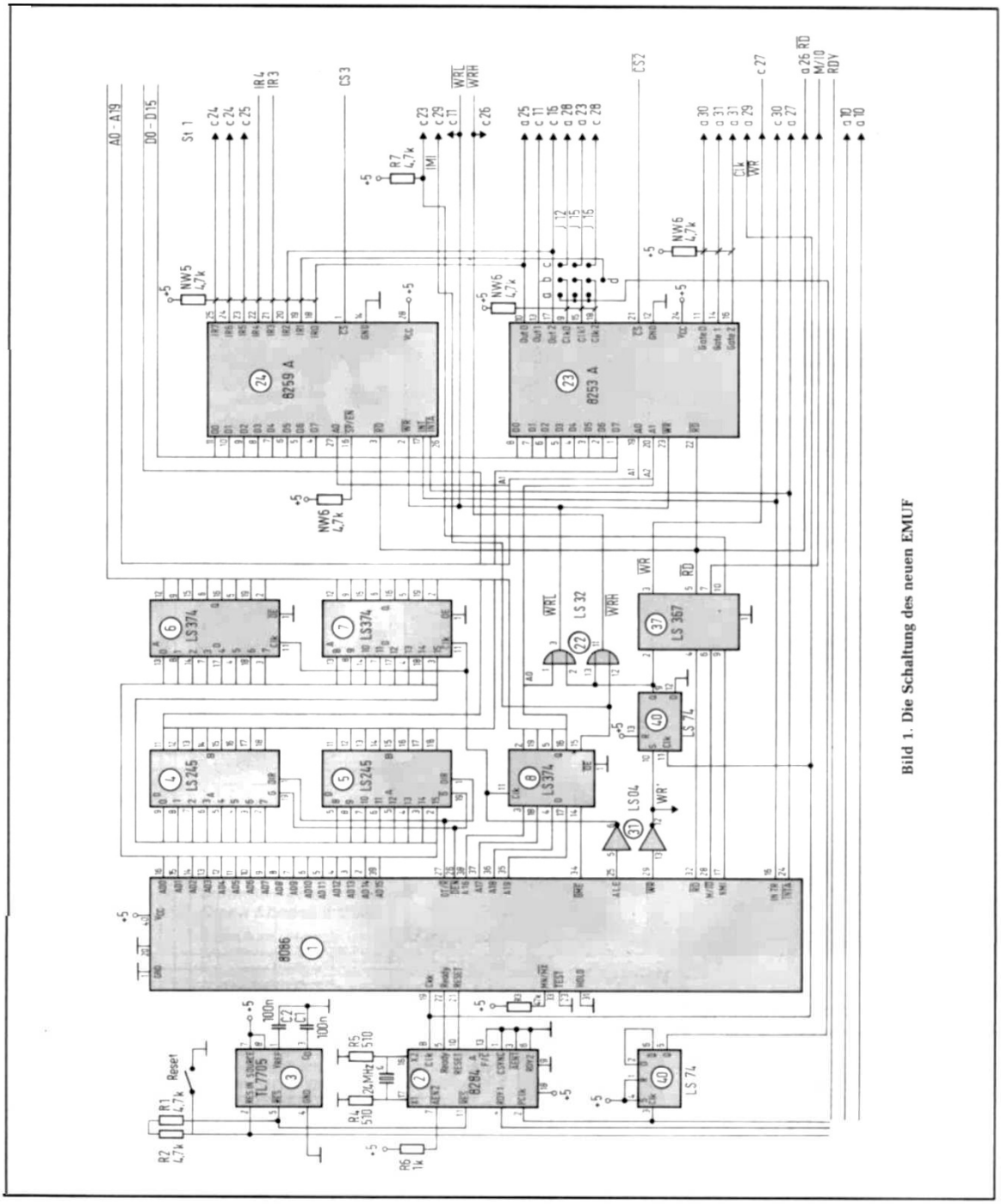


Bild 1. Die Schaltung des neuen EMUF

können). Zur Diskussion stehen u. a. Forth, Pearl, Pascal und „C", Lisp scheidet wohl aus.
In der Regel werden Forth und Pearl aufgrund mangelnder Verfügbarkeit (auf Diskette und im Kopf) keine Verwendung finden. Für Pascal spricht die große Verbreitung von Turbo-Pascal, das in seiner Grundform jedoch leider weder (EP)ROM-fähig noch voll reentrant (wichtig bei Verwendung von Interrupts) ist. 'Kochrezepte' (Patches), die diese Einschränkungen zumindestens teilweise wieder ausbügeln, wurden verschiedentlich veröffentlicht [1]. Übrig

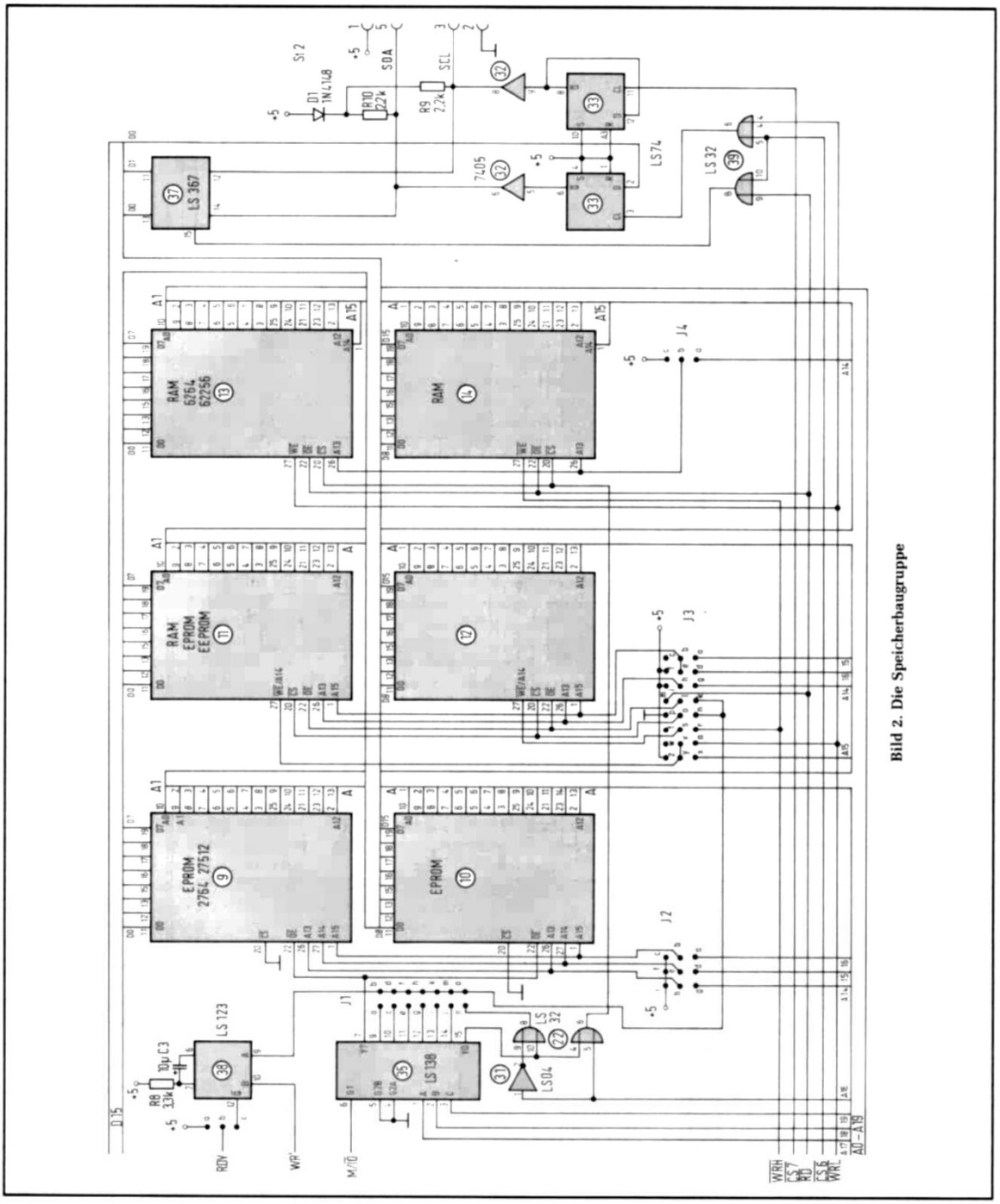


Bild 2. Die Speicherbaugruppe

bleibt, der Leser ahnt es schon, das haßgeliebte „C"! Und in der Tat ist spätestens die Programmierung eines EMUFs der rechte Anlaß, auf diese Hochsprache zu zugehen. Sie verdient es, und der Anwender hat anschließend alle Trümpfe in der Hand.

Ein wichtiger Aspekt ist die große Zahl sehr guter C-Compiler – nicht zuletzt das professionelle Microsoft-C und das vor kurzer Zeit veröffentlichte Turbo-C von Borland.

Pfui Assembler? Aber wer wird denn gleich! Schließlich wird, wer auch das Allerletzte an Leistung aus seinem Pro-

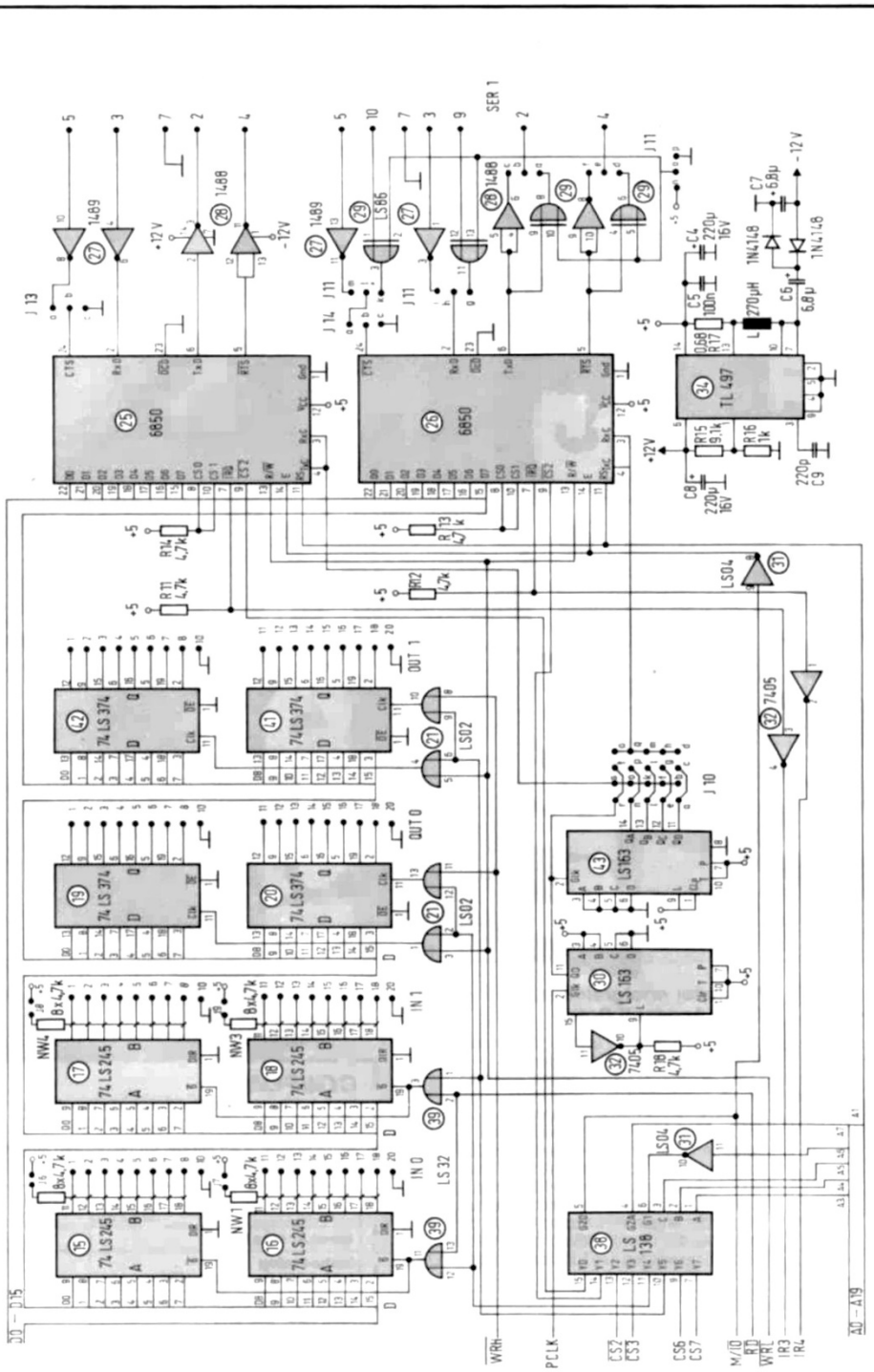


Bild 3. Die Ports

zessor kitzeln will (oder besser muß), intensiven Gebrauch von Interrupts machen und bei der Programmierung z. B. der Interrupt-Service-Routinen auf Assembler zurückgreifen. Zumal die Assemblerprogrammierung des 8086 auf einem System mit nicht mehr als 64 KByte Speicher nur angenehmer als auf einem Z80, 8085 oder gar 6502 ist (und das sage ich als langjähriger Apple-Fan!). Dies liegt daran, daß in einem solchen Fall alle Daten und der Stack noch in einem gemeinsamen Segment (64-KByte-Block) liegen. Etwas unangenehmer wird es, wenn alle vier Segmentregister verschiedene 64-KByte-Blöcke adressieren. Unzumutbar wäre jedoch selbst dieser Fall nicht, der bei einem EMUF aber sowieso die absolute Ausnahme bleiben wird und dem Einsteiger durch einige gute Bücher versüßt wird [2, 3].

Ein „tragfähiger" Kompromiß

Die Wahrheit liegt wie oft in der Mitte, und ein Optimum an effizienter Software kann entstehen, wenn Asssembler und C geschickt kombiniert werden, wobei man die Aushilfskrücke 'Inline-Code', d. h. byteweise in den Quelltext eingestreuten Maschinencode, vergessen darf: Kombinieren meint dann wirklich Linken! Für interessierte Leser sei an dieser Stelle unbedingt auf die Anregungen in [4] verwiesen.

Kompromißlose Hardware

Die Hardware des EMUF86 kommt hingegen vollkommen ohne Kompromisse aus:

- 8086(V30)-CPU mit 8 MHz Taktfrequenz
- Interrupt-Steuerbaustein (8259) mit 8 Eingängen
- Drei 16-Bit-Zähler (8253)
- Zwei serielle Schnittstellen (6850) bis 19200 Bd
- 32 TTL-Eingänge
- 32 TTL-Ausgänge
- I^2C-Bus-Adapter
- 6 Steckplätze für EPROM, EEPROM und RAM
- Spannungsüberwachung mit Reset
- 64polige Busleiste
- Nur 5-V-Versorgung erforderlich

Die Auswahl der Komponenten macht deutlich, daß der EMUF86 konsequent für abgeschlossene Automatisierungsaufgaben entwickelt wurde. Dementsprechend ist der Aufbau in moderner CMOS-Technik empfehlenswert. Für CMOS sprechen die niedrige Leistungsaufnahme und damit geringe Aufheizung der gesamten Baugruppe sowie eine gegenüber NMOS deutlich größere Störsicherheit bei dennoch erhöhter Taktfrequenz. Unabhängig davon, ob man den gesamten EMUF in CMOS realisiert, ist es in jedem Fall besser, sich bei der CPU für den V30 von NEC zu entscheiden! Dieser moderne Prozessor ist pin- und befehlskompatibel zum 8086, führt aber Programme bei gleicher Taktfreqenz um ca. 15...20 % schneller aus. Bei Verwendung des erheblich erweiterten Befehlssatzes läßt sich dieser Wert um ein Vielfaches steigern.
Die Arbeit mit diesen Befehlen wird durch Verwendung eines Makro-Assemblers wesentlich erleichtert und ist auf einem EMUF im Gegensatz zum PC sehr sinnvoll, da solche Progamme nicht auf allen 'Kompatiblen' laufen müssen, sondern für eine bestimmte Zielmaschine geschrieben werden. Trotz seiner größeren Leistung ist der V30 sogar noch sparsamer als sein Vorbild: So 'verkocht' der 8086 noch rund 1,7 W elektrische Leistung, wo der V30 mit nur 0,5 W auskommt. Dieser Wert läßt sich noch einmal auf 0,05 W absenken, wenn der V30 über den Halt-Befehl in den Standby-Modus geschickt wird. Aus diesem 'Tiefschlaf' wird er durch einen Interrupt (RESET, NMI oder INT) wieder geweckt. Diese in einem EMUF sehr sinnvolle Einrichtung ist aber nur eine von vielen Verbesserungen gegenüber dem 8086. Eine genauere Beschreibung dieses Prozessors sollte man einem guten Datenbuch entnehmen [5].

Interrupts

Zwei weitere, für die Arbeit mit einem Rechner für Steuerungsaufgaben sehr wichtige Bausteine sind die Chips 8259 und 8253. Der Interrupt-Controller 8259 liegt zwischen dem Prozessor und dessen Peripherie. Erwartet z. B. der 6850 die 'Zuwendung' des Prozessors, weil er ein Zeichen über die serielle Schnittstelle empfangen hat, so legt er seine INT-Leitung auf H-Pegel. Der 8259 ermittelt nun, ob der 6850 eine ausreichend hohe Priorität besitzt, die CPU zu diesem Zeitpunkt zu unterbrechen. Die Priorität eines Bausteins ist dadurch festgelegt, an welche der acht möglichen Interrupt-Eingänge (IR0...IR7) er angeschlossen wurde. Die IR0-Leitung hat die höchste, IR7 die geringste 'Wichtigkeit'.
Läßt der 8259 eine Interrupt-Anforderung zu, so zieht er seinerseits die INT-Leitung der CPU auf H-Pegel. Ist diese zur Interrupt-Behandlung bereit, abhängig davon, ob der Programmierer das I-Flag im Statusregister gesetzt (Unterbrechungen zugelassen) oder gelöscht hat, so bestätigt sie die Anforderung des 8259. Daraufhin legt der Interrupt-Controller ein Byte (Vektor) auf den Datenbus, aus dem der Prozessor die Adresse des Programmteils berechnet, das im genannten Beispiel das empfangene Zeichen aus dem 6850 liest und auf dem Bildschirm darstellt oder in einem Puffer ablegt.

IR0	Zähler 0	(8253A, IC 23)
IR1	Zähler 1	(8253A, IC 23)
IR2	Zähler 2	(8253 A, IC 23)
IR3	SER0	(6850, IC 25)
IR4	SER1	(6850, IC 26)
IR5	frei	(VG-Leiste, Pin 25c)
IR6	frei	(VG-Leiste, Pin 24a)
IR7	frei	(VG-Leiste, Pin 24c)

Bild 4. Interrupt-Quellen

Das Zusammenspiel von CPU und 8259 sieht auf den ersten Blick etwas kompliziert aus, erleichtert aber dem Programmierer die Arbeit ungemein. Er kann seine INT-Service-Routine wie ein normales Unterprogramm schreiben, da sie die Quelle der aktuellen Unterbrechung nicht erst zu ermitteln braucht. Beim 8259 im EMUF86 können die drei Zähler und die beiden 6850 Interrupts erzeugen. Die Leitungen IR5...IR7 stehen an der Erweiterungsleiste zur Verfügung (*Tabelle 1, Bild 4*).

Zähler

Der Zähler/Zeitgeber 8253 enthält drei voneinander unabhängige 16-Bit-Rückwärtszähler, von denen jeder einen Eingang, ein sogenanntes Tor (Gate) und einen Ausgang besitzt. Grundsätzlich generiert also jeder Kanal aus dem Signal an seinem Eingang, geteilt durch eine beliebige 16-Bit-Zahl, ein Ausgangssignal programmierbarer Frequenz.
Über den Gate-Anschluß läßt sich der Zähler triggern, d. h. zu einem festgelegten Zeitpunkt (oder Ereignis) starten. Jeder Kanal kann in sechs möglichen Betriebsarten arbeiten:

M0 Erzeugung periodischer Interrupts nach Eingang einer bestimmten Anzahl von Impulsen.
M1 Programmierbares Monoflop (retriggerbar)
M2 Taktgenerator
M3 Rechteckgenerator

J1: Adreßbereich für IC 11 + IC 12

n–o	10000h–1FFFFh
l–m	20000h–3FFFFh
i–k	40000h–5FFFFh
g–h	60000h–7FFFFh
e–f	80000h–9FFFFh
c–d	A0000h–BFFFFh
a–b	C0000h–DFFFFh

Adreßbereich von IC 13 + IC 14
0–0FFFFh

Adreßbereich von IC 9 + IC 10
E0000h–FFFFFh

J2: EPROM-Typ
2764: b–c, e–f, h–i
27128: b–c, e–f, g–h
27256: b–c, d–e, g–h
27512: a–b, d–e, g–h

J3: Speichertyp für IC 11 + IC 12
6264: e–f, h–i, l–m, n–o, r–s, u–v
65256: (32-KByte-RAM): a–b, g–h, l–m, n–o, r–s, u–v
2764: e–f, k–l, o–p, s–v, y–z
27128: e–f, g–h, k–l, o–p, s–v, x–z
27256: d–e, g–h, k–l, o–p, s–v, x–y
27512: d–e, g–h, k–l, o–p, s–v, x–y

J4: RAM-Typen für IC 13 + IC 14
6264: b–c
65256: a–b

J5:
a–b: wenn keine EEPROMs für IC 11 + IC 12 verwendet werden
b–c: für EEPROMs HN 58064 (J3 muß für 6264 gesteckt sein)

J6, J7, J8, J9
wenn gesteckt, Pull-up-Widerstände an INx aktiv

J10: Baudrate SER0 +SER1

	19200	9600	4800	2400	1200
SER0	r–s	n–o	i–k	e–f	a–b
SER1	t–u	p–q	l–m	g–h	c–d

bei Initialisierung des 6850 mit Clock/16

J11: V.24/TTL-Pegel an SER1

	T×D	$\overline{RTS}$	R×D	$\overline{CTS}$
TTL	a–b	d–e	g–h	k–l
V.24	b–c	e–f	h–i	l–m

n–o: TTL-Signale an SER1 invertiert
o–p: TTL-Signale an SER1 nicht invertiert

J12:
a–b: Clk0 = 2 MHz
b–c: Clk0 von ext. über a28 der VG-Leiste

J14:
a–b: $\overline{CTS}$ von SER1
b–c: $\overline{CTS}$ mit Masse verbunden

J15:
a–b: Clk1 = 2 MHz
b–c: Clk1 von ext. über a23 der VG-Leiste

J16:
a–b: Clk2 = 2 MHz
b–c: Clk2 von ext. über c28 der VG-Leiste
b–d: Clk2 verbunden mit OUT1 (Zähler 1 und 2 sind hintereinandergeschaltet)

J18:
a–b: $\overline{CTS}$ von SER0
b–c: $\overline{CTS}$ mit Masse verbunden

Bild 5. Jumper auf der Platine und ihre Bedeutung

M4 Softwaremäßig triggerbarer Ausgang (bei Nulldurchgang des Zählers)
M5 hardwaremäßig triggerbarer Ausgang (bei Flanke am Gate)

Die Eingänge und Tore sind entweder über den Bus zugänglich, oder es wird als Zähltakt ein auf der Platine vom CPU-Takt abgeleitetes 2-MHz-Signal verwendet. Zusätzlich kann der Ausgang von Zähler 1 auf den zweiten Zähler geschaltet werden, um so einen 32-Bit-Zähler zu erhalten.

Eine Entscheidung für jeden der drei Kanäle trifft man mit den Steckbrücken (Jumper) J12, J15 und J16 (*Bild 5*). Eine detaillierte Beschreibung der sechs Modi dieses sehr leistungsfähigen Bausteins sollte auf jeden Fall den Datenblättern der Hersteller (u. a. NEC, Intel und Siemens) entnommen werden.

Serielle Verbindung mit anderen Computern

Die Verbindung des EMUF86 mit anderen Computern erfolgt seriell per V.24-Schnittstellen, die mit dem 6850 realisiert sind. Hier werden zum erstenmal nicht PC-typische Bausteine eingesetzt, da der standardmäßig verwendete ACE 8250 rund den 5fachen Preis kostet, ohne wesentlich besser zu sein.

Die Pfostenstecker beider Schnittstellen (SER0, SER1) sind normentsprechend belegt und flachbandkompatibel zu je einer DB25-Buchse (*Bild 6*). Dadurch erfolgt bei Einbau des EMUFs in ein eigenes Gehäuse die Verkabelung wie mit einem Modem. Die für die Pegelwandlung von TTL nach V.24 benötigten ±12 V werden auf der Platine durch einen TL497 erzeugt, wodurch man mit nur einer Betriebsspannung für die gesamte Schaltung auskommt. Bei SER0 liegen die Übertragungspegel fest auf V.24-Norm, während bei SER1 die Möglichkeit besteht, auf die Pegelumwandlung zu verzichten und mit TTL-Signalen zu arbeiten. Dies spart zwei ICs und ermöglicht andererseits eine Kommunikation über die parallele Druckerschnittstelle des Entwicklungssystems, wenn dieses über kein serielles Interface verfügt. Diese Form des Datenaustauschs im TTL-Pegel ist problemlos möglich, solange die Verbindungskabel nur eine Länge von wenigen Metern haben.

Abweichend von der Grundeinstellung nach Bild 5 sollte der Aufbau dann gemäß *Bild 7* und *Bild 8* erfolgen.

SER 0	
Pin	Signal
2	TxD
3	RxD
4	RTS
5	CTS

SER 1	
Pin	Signal
2	TxD (abh. v. J11b)
3	RxD (V.24)
4	RTS (abh. v. J11e)
5	CTS (V.24)
7	Gnd
9	RxD (TTL)
10	CTS (TTL)

Bild 6. Belegung der beiden seriellen Schnittstellen

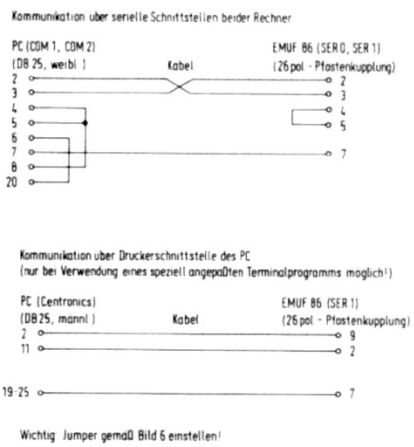


Bild 7. Der PC kann über die serielle Schnittstelle oder über die Druckerschnittstelle an den EMUF angeschlossen werden

Jumper	Funktion	Stellung
J11	TTL-Pegel an T×D	a–b
J11	TTL-Pegel an $\overline{RTS}$	d–e
J11	TTL-Pegel an R×D	g–h
J11	TTL-Pegel an $\overline{CTS}$	k–l
J11	TTL-Pegel nicht invertiert	o–p
J10	Baudrate 1200 Bd	c–d
J14	Kein Hardware-Handshake	b–c

Bild 8. Jumper-Einstellung bei Kommunikation über die Druckerschnittstelle des PC

Da die Kommunikation in der Regel das Software-orientierte XON-/XOFF-Protokoll verwendet, besteht die Möglichkeit, über J13 (SER0) und J14 (SER1) das $\overline{CTS}$-Handshake-Signal dauerhaft zu aktivieren (L-Pegel). Es ist zu beachten, daß die TTL-Signale bei SER1 unabhängig von den mit Jumper J11 gewählten Pegeln mit auf dem Pfostenstecker (Belegung siehe Bild 4) liegen. Dadurch besteht auch die Möglichkeit, externe Umsetzer auf z. B. RS485, LWL oder 20-mA-Stromschleife anzuschließen. Dazu kann es eventuell nötig werden, die TTL-Signale vorher mittels J11, Pin o zu invertieren.
Ähnlich flexibel läßt sich die Baudrate einstellen. Sie erfolgt für jeden der beiden Kanäle getrennt durch J10 (Bild 5). Zusammen mit dem internen Teiler des 6850 ergeben sich so sieben verschiedene Übertragungsgeschwindigkeiten zwischen 300 und 19 200 Bd. In Bild 7 wird detailliert gezeigt, wie man PC und EMUF verbinden kann.

„Altmodische" Ein-/Ausgabe

Die Ein-/Ausgabe-Ports sind mit einfachen TTL-Bausteinen realisiert, was auf den ersten Blick etwas altmodisch aussieht, jedoch flexibel und preiswert die Möglichkeit bietet, bei Bedarf 16 Bit breite Daten in einem Zyklus zu lesen (IN0, IN1) oder ausgeben (OUT0, OUT1). Dazu werden jeweils zwei 8-Bit-Port-Bausteine zusammengefaßt und über einen 20poligen Pfostenstecker (*Bild 9*) herausgeführt, an die mit Flachbandkabel unter anderem passende E/A-Module für die 24-V-Steuerungstechnik angeschlossen werden können. Mehrere solcher Erweiterungsmodule sind bereits realisiert.

Bild 9. Belegung der E/A-Stiftleisten

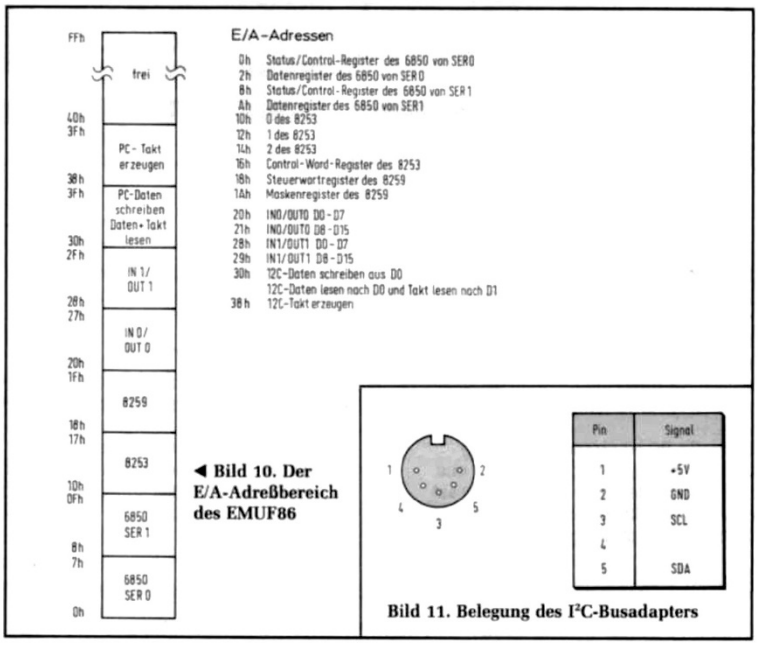
Die Adressen, unter denen die Ports angesprochen werden können, zeigt *Bild 10*. Die Eingänge sind wahlweise mit Pull-up-Widerständen bestückbar, können aber jederzeit in 8-Bit-Gruppen wieder davon freigeschaltet werden (J6...J9). Die Stiftleisten IN0 und OUT0 sowie IN1 und OUT1 sind so angeordnet, daß durch Bestücken des Flachbandkabels mit zwei Pfostenkupplungen die Datenausgänge zurückgelesen werden können.

Mit I²C-Bus!

I²C! Nein, diesmal nicht das Neueste aus der Elektrotechnik, sondern der Name eines bidirektionalen 2-Bit-Bussystems. Dieser 'Bus' ist auf eine 5polige DIN-buchse (*Bild 11*) herausgeführt und ermöglicht den Anschluß weiterer Baugruppen. Nach I²C-Konvention werden die angeschlossenen Systeme untereinander durch eine Daten- und Taktleitung verbunden. Die Kommunikation erfolgt dann bitseriell, ist aber nicht zeitsynchronisiert (wie z.B. V.24), sondern der Sender synchronisiert seine Daten selbst, indem er fortlaufend nach dem Herausschreiben eines Bits die Taktleitung invertiert.
Zur Realisierung des I²C-Anschlusses ist ein 2-Bit-Port vorgesehen. Wird dieser Port auf Adresse 30h gelesen, so befindet sich in D0 der momentane Wert von Pin 5 (SDA) der DIN-Buchse und in D1 der Wert an Pin 3 (SCL). Das Bit in D0 ist ein gültiges Datenbit, wenn die Taktleitung L-Pegel führt und sich ihr Zustand seit dem letzten Zugriff geändert hat. Umgekehrt schreibt der Sender das auszugebende Bit D0 in die Adresse 30h und invertiert anschließend durch bloßen Zugriff (lesend oder schreibend) auf die Adresse 38h die Taktleitung. Damit signalisiert er dem Empfänger die Gültigkeit des geschriebenen Bits. Ein Datenaustausch ist jedoch nicht ausschließlich mit Baugruppen möglich, die speziell für den I²C-Bus ausgelegt sind, sondern auch mit C-Bus-Geräten oder anderen 2-Draht-Bussen. Meist ist dann bei mehreren Stationen ein Hinzufügen von Freigabeleitungen nötig, die über OUT0 und OUT1 leicht erzeugt werden können.

Diverses

Neben den auffälligen Bestandteilen verrichten auch noch etliche andere Bausteine ihren Dienst, die von 'Software-Technikern' unbeachtet bleiben werden, da sie nicht programmierbar sind. Trotzdem läuft ohne sie gar nichts! Da ist z.B. ein TL7705, der die 5-V-Spannungsversorgung überwacht. Sollte sie einen Wert von 4,8 V unterschreiten, wird sicherheitshalber ein Reset ausgelöst, um ein Weiterrechnen mit falschen Daten zu



◀ Bild 10. Der E/A-Adreßbereich des EMUF86

Bild 11. Belegung des I²C-Busadapters

verhindern. Damit dies nicht geschieht, sollte das Netzteil für eine Belastung mit 2 A ausgelegt werden, wovon der EMUF, abhängig vom Umfang der Bestückung rund 1,5 A aufnimmt. Der 'Saft' kann sowohl über die Busleiste als auch per separatem seitlichem Anschluß zugeführt werden.
Während der Entwicklungszeit wird es in den meisten Fällen möglich sein, den EMUF86 über das PC-Netzteil zu versorgen. Dies hängt natürlich von dessen Belastbarkeit und der Zahl der Erweiterungskarten im PC ab. Spätestens nach Fertigstellung des Steuerprogramms braucht der EMUF jedoch eine eigene Stromversorgung, für die sich z. B. nach einer kleinen Modifikation die POW5V-Baugruppe des NDR-Klein-Computers gut eignet. Diese Schaltung hat sich vielfach bewährt und ist mit einem Preis von rund 30 DM auch sehr preiswert.

Großzügiger Speicher

Speichern kann der EMUF86 reichlich! Er ist mit sechs Sockeln ausgestattet, von denen jeweils zwei nur entweder EPROMs (bis27512) oder RAMs (bis 62256) aufnehmen können (*Bild 12*). Das dritte Paar, die beiden mittleren Sockel (IC11 und IC12), sind mit J1 auf sieben verschiedene Startadressen einzustellen, u. a. so, daß je nach Bedarf ein zusammenhängender ROM- oder RAM-Bereich entsteht. Außerdem lassen sich in dieses Sockelpaar auch EEPROMs (elektrisch löschbare PROMs) einsetzen. Hierbei können sowohl die RAM-ähnlichen Data-Polling-Typen eingesetzt werden, als auch die billigen Typen, die das $\overline{WR}$-Signal für 10 ms auf low benötigen. Dazu wird der Prozessor entsprechend lange angehalten.

0h...0FFFFh	RAM
10000h...DFFFFh	RAM, EPROM oder EEPROM (64-KByte-Bereich mit J1 wählen)
E0000h...FFFFFh	EPROM

Bild 12. Speicheradressen und mögliche Bestückung

Da die CPU nach einem Reset mit der Ausführung eines Programms an der Adresse FFFF0h beginnt, müssen mindestens die EPROM-Sockel IC9 und IC10 bestückt werden. Normalerweise wird an dieser Stelle das Monitorprogramm eingesetzt, das die Initialisierung der Peripheriebausteine besorgt und darüber hinaus einige nützliche Befehle zur Verfügung stellt (*Bild 13*). Der Monitor ermöglicht unter anderem das Laden eines Programms ins RAM oder EEPROM, das auf einem anderen Rechner erstellt wurde. Darüber hinaus lassen sich auch Register, Speicher und E/A-Bausteine gezielt 'bearbeiten', wobei der PC zum Terminal degradiert wird. Ein entsprechendes Programm für PC-kompatible Rechner mit folgenden Eigenschaften ist verfügbar:

- Terminal-Emulation mit Kommunikation über V.24.
- Programm-'Download' zum EMUF86.
- Daten-'Upload' vom EMUF86 mit Aufzeichnung auf Floppy oder Harddisk.
- Ausgabe der vom EMUF86 aufgenommenen Daten auf den Drucker.
- Umwandeln einer MS-DOS Maschinencode-Datei in zwei getrennte Dateien für das Programmieren von EPROMs (ODD, EVEN).

Adreßbelegung

Die Adreßdecodierung ist recht einfach gehalten: Der E/A-Bereich wird nur auf 256 Port-Adressen ausdecodiert, wovon auf der Platine nur 64 belegt sind. Die restlichen 192 bleiben frei für Erweiterungen über den Bus. Die Beschränkung auf 256 Adressen ist sinnvoll, da so unnötige Decodierlogik vermieden wird. Im verbleibenden Adreßraum lassen sich noch mehr als zwei Dutzend weiterer Port-Bausteine unterbringen. Programmiertechnisch ergibt sich ebenfalls ein Vorteil, da die ersten 256 Ports direkt adressiert werden können. Ein vorheriges Laden des DX-Registers beim Zugriff auf eine 16-Bit-Port-Adresse ist nicht nötig. Zum Beispiel:

```
OUT 10,al ; Schreibe Akku in den Port
          ; auf Adresse 10
```

gegenüber:

```
MOV DX,300 ; lade DX-Register mit Port
           ; auf adresse 300
OUT DX,AL  ; indirekt adressierter
           ; Port-Zugriff
```

Außer den TTL-Ports werden die E/A-Bausteine nur auf den geraden Adressen erreicht. Das bedeutet, daß z. B. das Statusregister des 6850 auf Adresse 0 liegt, sein Datenregister aber nicht auf 1, sondern auf 2. Eine genaue Aufstellung der Portadressen aller im EMUF86 verwendeten ICs enthält Bild 10.

Enter ASCII-Text Into Memory	: A
Display Memory	: D Start Ende
Set ES-Register	: E Wert
Fill Memory	: F
Execute a User Program	: G
ADD & SUB two Hex Numbers	: H
Read Port	: I
Load Program	: L
Move Memory	: M
Test Memory	: N
Output to a Port	: O
Register Display	: R
Enter Hex Data into Mem.	: S
Compare two Memory Blocks	: V

Bild 13. Liste der Monitorkommandos

Nun wird der EMUF zusammengebaut

Der Zusammenbau des EMUF86 sollte in jedem Fall sehr gewissenhaft erfolgen, da die Fehlersuche auf einer solch umfangreichen Platine kein 'Zuckerlecken' ist. Es wäre z. B. sehr sinnvoll, wenn Anfänger einen Freund mit mehr Erfahrung (und im Fehlerfall mit Oszilloskop) um Mithilfe bäten. Die Verwendung von qualitativ guten Sockeln ist nicht nur bei den 'großen' ICs gute Schule, sondern hat sich allgemein als sehr sinnvoll herausgestellt. An dieser Stelle ein paar Mark sparen zu wollen kann hartnäckigen Ärger einbringen.

Platinen, Bausätze und auch Fertiggeräte können vom Elektronikladen, Detmold, bezogen werden. Die Stückliste zeigt *Bild 14* (siehe Seite 144).

Hinweise für das Bestücken umfangreicher Platinen wurden in mc schon vielfach gegeben, deshalb das Wesentliche in Kürze: Zuerst die wenigen passiven Bauteile (Widerstände, Kondensatoren, Quarz u. ä.), dann Sockel, Stiftreihen und Erweiterungsleiste einlöten. Als letztes kommt die einem Programmierer ewig dubiose Spule für den ±12-V-Schaltregler an die Reihe: Sie besteht aus nichts weiter als einem kleinen Ring aus Ferrit (Durchmesser 12 mm), durch den etwa acht Windungen isolierten Kupferdrahts (bis 1 mm Durchmesser) gezogen werden. Das war's schon!

Bevor das ganze (*Bilder 15* und *16*) zum ersten Mal unter Spannung gesetzt wird, müßen noch die der gewählten Ausbauvariante entsprechenden Jumper gemäß

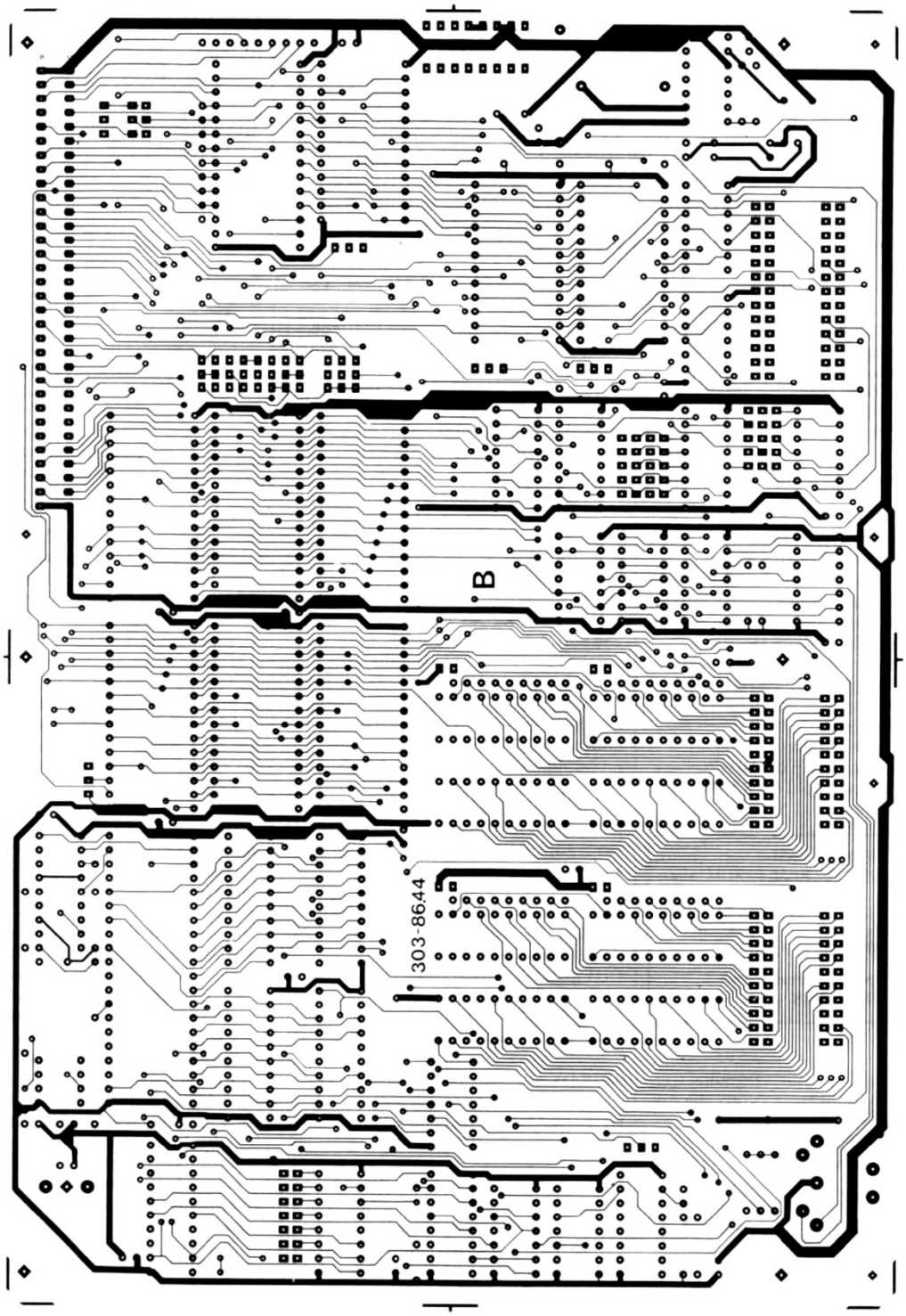


Bild 15. Layout (Lötseite)

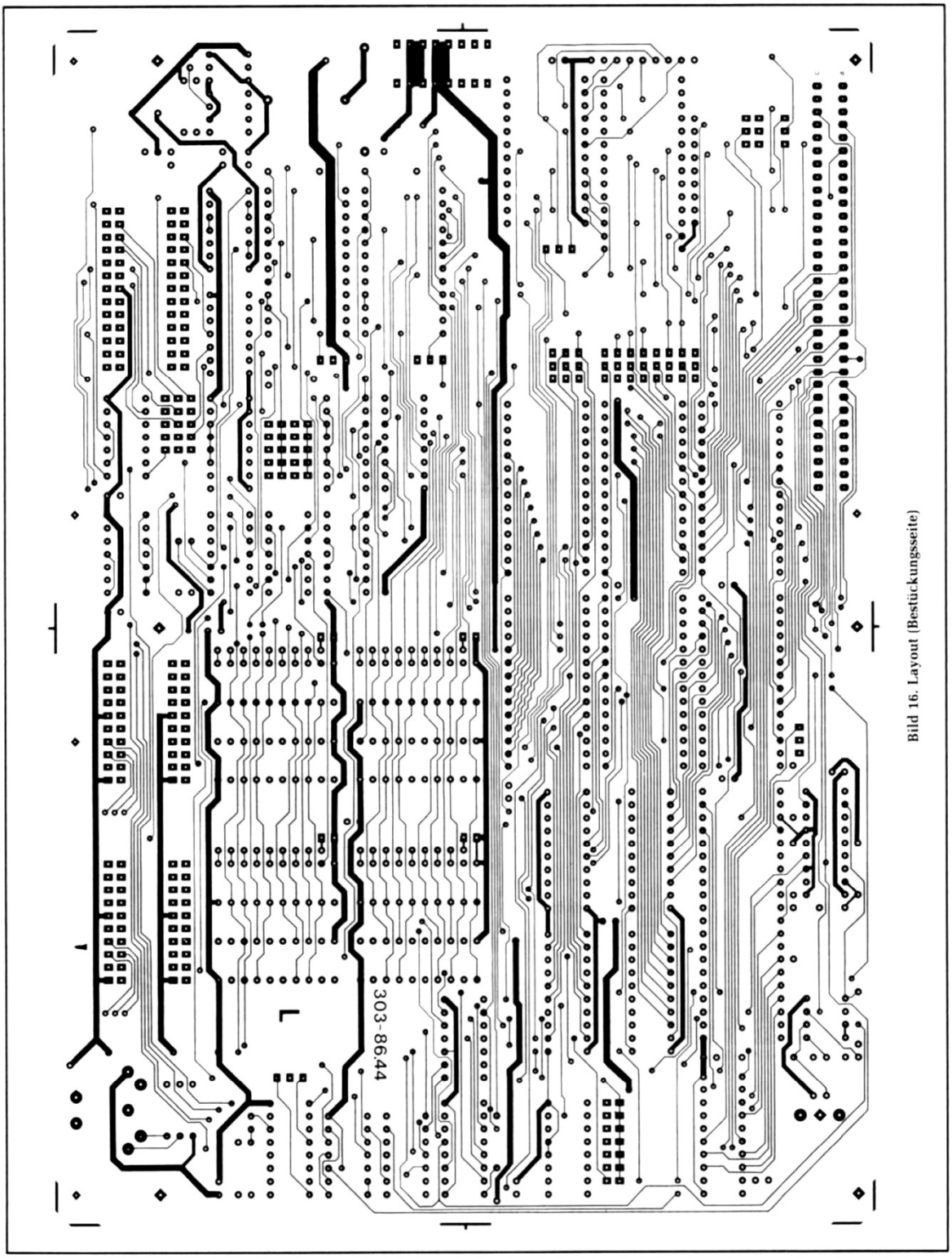


Bild 16. Layout (Bestückungsseite)

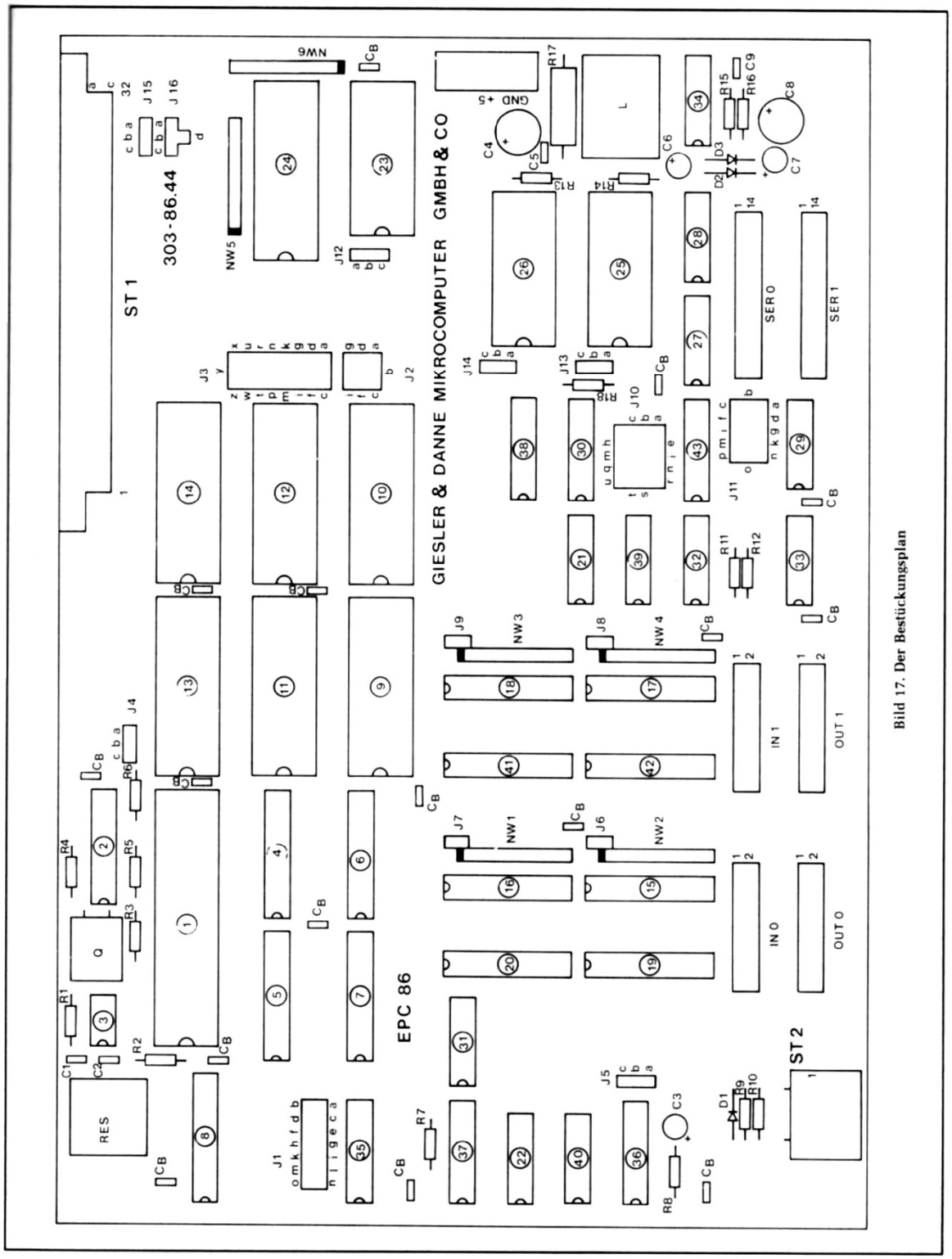


Bild 17. Der Bestückungsplan

Stück	Aufdruck	Beschreibung
1	IC 1	8086-2 o. V30-8
1	IC 2	8284A
1	IC 3	TL7705
6	IC 4, 5, 15, 16, 17, 18	74LS245
7	IC 6, 7, 8, 19, 20, 41, 42	74LS374
6	IC 9, 10, 11, 12, 13, 14	Speicher
1	IC 21	74LS02
2	IC 22, 39	74LS32
1	IC 23	8253
1	IC 24	8259A
2	IC 25, 26	6850
1	IC 27	MC1489
1	IC 28	MC1488
1	IC 29	74LS86
2	IC 30, 43	74LS163
1	IC 31	74LS04
1	IC 32	7405
2	IC 33, 40	74LS74
1	IC 34	TL497
2	IC 35, 38	74LS138
1	IC 36	74LS123
1	IC 37	74LS367
1	Q	Quarz 24 MHz HC-18/U Serienresonanz
19	C1, C2, C5, CB	Kondensator 100 nF RM 2,54 mm
1	C3	Tantal-Elko 10 µF/16 V
1	C4, C8	Elko 220 µF/16 V stehend RM 5,08 mm
2	C6, C7	Tantal-Elko 6,8 µF/16 V
1	C9	Kondensator 200 pF Keramik
9	R1, R2, R3, R7, R11, R12, R13, R14, R18	Widerstand 4,7 kΩ
2	R4, R5	Widerstand 510 Ω
1	R6, R16	Widerstand 1 kΩ
1	R8	Widerstand 3,3 kΩ
2	R9, R10	Widerstand 2,2 kΩ
1	R15	Widerstand 9,1 kΩ
1	R17	Widerstand 0,68 Ω/1 W
5	NW1, NW2, NW3; NW4, NW5	SIL-Array 8× 4,7 kΩ
1	NW6	SIL-Array 4× 4,7 kΩ
3	D1, D2, D3	Diode 1N4148
1	L	Spule 270 µH
1	zu IC 2	IC-Sockel DIL 18
3	zu IC 23, 25, 26	IC-Sockel DIL 24
7	zu IC 9, 10, 11, 12, 13, 14, 24	IC-Sockel DIL 28
1	zu IC 1	IC-Sockel DIL 40
4	IN0, IN1, OUT0, OUT1	Stiftleiste 2× 10pol.
2	SER0, SER1	Stiftleiste 2× 13pol.
1	J1	Stiftleiste 2× 7pol.
1	J2	Stiftleiste 3× 3pol.
1	J3	Stiftleiste 3× 8pol.
7	J4, J5, J12, J13, J14, IR012, Gate012	Stiftleiste 1× 3pol.
4	J6, J7, J8, J9	Stiftleiste 1× 2pol.
1	J10	Stiftleiste 4× 5pol.
1	J11	Stiftleiste 3× 5pol.
1	NMI/IR	Stiftleiste 1× 4pol.
1	–	Buchsenleiste 8pol.
20	–	Jumper
1	RES	Siemens-Tastenelement mit Tastenkappe
1	ST1	VG64-Messerleiste abgew./gerade
1	ST2	Diodenbuchse 5pol. zur Platinenmontage
1	–	Leiterplatte

Bild 14. Die Stückliste

Bild 5 gesteckt und eine sorgfältige Prüfung der Löt- und Bestückungsarbeit durchgeführt werden. Nach dem Anschluß der Versorgungsspannung ist zu kontrollieren, ob an allen Sockeln Masse und +5 V an den richtigen Pins (und sonst keinen!) anliegen. Nach dem anschließenden Trennen von der Betriebsspannung kann man beginnen, die ICs in ihre Sockel zu setzen, wobei peinlichst darauf geachtet werden muß, alle Bausteine richtig herum und auch alle Pins unverbogen zu 'versenken'. Hier beugt ein Blick auf den Bestückungsplan *Bild 17* unnötigen Kopfschmerzen wirksam vor.
Beim Arbeiten mit CMOS-ICs sollten die bekannten Vorsichtsmaßnahmen, wie Transsport nur in speziellen Vorrichtungen und vor dem Berühren ein sich 'Entladen' über geerdete Stahlteile, unbedingt eingehalten werden. Nach Fertigstellung und Test sollte der Einbau in ein robustes Gehäuse erfolgen, wie es für den industriellen Einsatz z. B. die Fa. Rittal anbietet.

Nun wird's spannend

Nach dem Verbinden von EMUF86 und PC wird das Netzteil erneut aktiviert. Befindet sich nun das Monitorprogramm in den EPROMs und läuft ein Terminalprogramm auf der Gegenseite, so erscheint (hoffentlich) folgende Meldung:

EPC 86 – Monitor

Dies signalisiert dem faszinierten Betrachter das erfolgreiche Durchlaufen der Initialisierungsroutine im EMUF86. Sollte diese Bestätigung auch nach wiederholtem Drücken der Reset-Taste ausbleiben, muß sich der Besitzer auf den 'dornigen Weg' der Fehlereingrenzung begeben. Nach einem erfolgreichen Starten des Monitors stehen dann die Befehle aus Bild 13 zur Verfügung.

Literatur

[1] c't 12/85, S.74 und 2/87, S.72
[2] *Sargent,Shoemaker,Stelzer:* Assemblersprache und Hardware des IBM PC. Addison-Wesley.
[3] *Bradley:* Programmieren in Assembler, Hanser-Verlag.
[4] *Biggerstaff:* System Software Tools. Prentice-Hall.
[5] V30 User's Manual. NEC Deutschland.