

# Anschluß an den Rest der Welt

Serielle Schnittstelle für Atari XL/XE in der Assemblerecke

**D**er kleine Atari ist nicht mit sehr vielen (normgerechten) Schnittstellen ausgestattet. So kommt es, daß er ein einsames Dasein ohne jede Verbindung zu anderen Computern fristen muß. Das gebräuchlichste Interface mit dessen Hilfe man Kontakt zu fast allen Computern, Akkustikkopplern und fremden Datenverarbeitungsanlagen aufnehmen kann, ist die RS-232-Schnittstelle. Diese muß schnell große Datenmengen übertragen können, so daß sie über den Expansions-Port anzuschließen ist. In dieser Assemblerecke liefern wir die Hard-

Am besten verwendet man eine Federleiste mit gewinkelten Anschlüssen und lötet sie direkt auf der Platine an. Da am Expansions-Port der XL-Computer leider keine Betriebsspannung von +5 V zur Verfügung steht, muß man sich mit einem Extrastecker an einem der beiden Joystickports behelfen.

Adresse	Name	Funktion beim Lesen	Funktion beim Schreiben
\$D100	DATA-Register	enthält empfangenes Daten-Byte	übergibt zu sendendes Daten-Byte an ACIA
\$D101	STATUS-Register	enthält Status-Byte	lös Programm-Reset aus
\$D102	COMMAND-Register	enthält Kommando-Byte	übergibt Kommando-Byte an ACIA
\$D103	CONTROL-Register	enthält Kontroll-Byte	übergibt Kontroll-Byte an ACIA

Für Besitzer eines Atari XE haben die Entwickler an die Betriebsspannung gedacht. Der Preis dafür ist aber ein völlig geänderter Expansions-Port. Es gibt nun das ECI, das erst in Verbindung mit dem Modul-Port einen vollständigen Expansions-Port bildet. Außerdem wird jetzt keine Federleiste mehr gebraucht, sondern zwei Platine-randstecker. Auf dem abgedruckten Schaltplan sind die Pin-Belegungen für die Ports beider Computer verzeichnet. Standardmäßig wird für die RS-232-Schnittstelle eine 25polige SUB-D-Buchse verwendet. Da das Raster der Anschlußbeinchen nicht mit dem einer Lochrasterplatine übereinstimmt, müssen Sie die Beinchen beim Einlöten entsprechend verbiegen oder die Buchse mit einem Kabel anschließen.

## Die Bauteile der RS-232-Schnittstelle

Das Herzstück der Schaltung ist der ACIA (Asynchronous Communications Interface Adapter) 6551. Seine Steuerleitungen sind kompatibel zur 6502-CPU des Atari. Der ACIA übernimmt die gesamte serielle Datenübertragung und kümmert sich um alle Formalitäten wie Baud-Rate, Parität und Start-/Stopbits. Er liefert an seinen Ausgängen aber nur standardmäßige TTL-Pegel (0 bzw. 5 V), während der RS-232-Standard höhere Spannungsdifferenzen (ca. 12 V bzw. ca. -12 V) verlangt. Deshalb sind zwei Treiber-ICs ICL232 (baugleich mit MAX232) nötig. Jedes von ihnen enthält zwei Eingangstreiber zur Umwandlung des RS-232-Pegels in den TTL-Pegel und zwei Aus-

gangstreiber für den umgekehrten Vorgang.

Außerdem benötigt man noch zwei TTL-ICs zur Speicherdekodierung. In unserer Schaltung werden dem ACIA die Speicherzellen \$D100 bis \$D103 zugewiesen. Die Speicherdekodierung ist in unserer Schaltung, um Bauteile zu sparen, nicht vollständig, d.h., die vier ACIA-Register wiederholen sich 256 Bytes lang periodisch (bis \$D1FFF).

Über das COMMAND- und das CONTROL-Register kann man dem ACIA alle für die Datenübertragung wichtigen Parameter mitteilen. Auch die Interruptsteuerung des ACIA läßt sich über diese Register beeinflussen. Das STATUS-Register kann nur gelesen werden und enthält alle wichtigen Informa-

## 8 Bit

ware-Schaltung und erklären die notwendige Software zur Einbindung in das Atari-Betriebssystem. Außerdem zeigen wir anhand eines kleinen Basic-Programms, wie man grundsätzlich ein Terminal-Programm zur Datenübertragung erstellt.

### Aufbau der RS-232-Schaltung

Die Schaltung kann problemlos auf einer Lochrasterplatte aufgebaut werden. Etwas Sorgfalt verlangt aber der Anschluß der 50poligen Federleiste für die Verbindung mit dem Atari XL.

**CONTROL-Register**

**Bit 7 Anzahl der Stopbits**  
 0 1 Stopbit  
 1 2 Stopbits  
 1 1.5 Stopbits, wenn Wortlänge = 5  
 und keine Parität  
 1 1 Stopbit, wenn Wortlänge = 8  
 und Parität

**Bit 6 5 Wortlänge**

0 0 8 Bits  
 0 1 7 Bits  
 1 0 6 Bits  
 1 1 5 Bits

**Bit 4 Taktgeber für Empfänger**

0 externer Takt  
 1 eingestellter interner Baudrate

**Bit 3 2 1 0 Baud-Rate**

0 0 0 0 10mal externer Takt  
 0 0 0 1 50  
 0 0 1 0 75  
 0 0 1 1 109.92  
 0 1 0 0 134.58  
 0 1 0 1 150  
 0 1 1 0 300  
 0 1 1 1 600  
 1 0 0 0 1200  
 1 0 0 1 1800  
 1 0 1 0 2400  
 1 0 1 1 3600  
 1 1 0 0 4800  
 1 1 0 1 7200  
 1 1 1 0 9600  
 1 1 1 1 19200

**COMMAND-Register****Bit 7 6 Paritätsmodus**

0 0 ungerade Parität beim Senden  
 und Empfangen  
 0 1 gerade Parität beim Senden und  
 Empfangen  
 1 0 High-Parität nur beim Senden,  
 Paritätsüberprüfung aus  
 1 1 Low-Parität nur beim Senden,  
 Paritätsüberprüfung aus

**Bit 5 Parität**

0 keine Parität  
 1 Paritätsmodus einschalten

**Bit 4 Empfänger Echomodus**

0 normaler Modus  
 1 Echomodus, Bit 2 und 3 müssen  
 0 sein, RTS geht auf low.

**Bit 3 2 Sender-Interrupt-Kontrolle**

0 0 RTS-high, Sender-Interrupt aus  
 0 1 RTS-low, Sender-Interrupt ein  
 1 0 RTS-low, Sender-Interrupt aus  
 1 1 RTS-low, Sender-Interrupt aus  
 Break senden über TxD

**Bit 1 Interrupt-Kontrolle**

0 IRQ ermöglicht  
 1 IRQ aus

**Bit 0 Daten-Terminal (ACIA) bereit**

0 nicht bereit (DTR high)  
 1 bereit (DTR low)

**STATUS-Register** (mit \* gekennzeichnete Ereignisse sind Gründe für das Auslösen eines IRQ)

**Bit 7 Interrupt-Steuerung (IRQ-Ausgang) wird nach Lesen des STATUS-Registers gelöscht**

0 kein IRQ  
 1 IRQ aufgetreten\*

**Bit 6 Datengegenstelle (DSR-Eingang)**

0 bereit - DSR low\*  
 1 nicht bereit - DSR high\*

**Bit 5 Trägersignal (DCD-Eingang)**

0 liegt vor - DCD low\*  
 1 liegt nicht vor - DCD high\*

**Bit 4 STATUS-Senderregister**

0 nicht leer  
 1 leer\*

**Bit 3 STATUS-Empfangsregister**

0 nicht voll  
 1 voll\*

**Bit 2 Überlauffehler**

0 kein Fehler  
 1 Fehler

**Bit 1 Fehler bei Start- oder Stopbit**

0 kein Fehler  
 1 Fehler

**Bit 0 Paritätsfehler**

0 kein Fehler  
 1 Fehler

tionen über den Zustand des ACIA. Die Bit-Belegung aller Register entnehmen sie bitte dem Kasten.

Ansonsten sind auf der Platine nur noch acht Elkos, ein Quarz und eine LED mit Widerstand zu finden. Die Daten-, Steuer-, Adreß- und Datensteuerleitungen des ACIA sind kompatibel zu der CPU und können daher direkt mit dem Expansions-Port verbunden werden. Hervorzuheben ist noch der IRQ-Ausgang, über dem der ACIA mit Low meldet, daß er etwas vom Computer will. Doch dazu später mehr.

**Die Programmierung des ACIA 6551**

Der ACIA stellt vier Register zur Verfügung, mit deren Hilfe man alle Funktionen im Griff hat (s. Kasten).

Wie man erkennen kann, ist das CONTROL-Register nur zur Steuerung des Übertragungsprotokolls da und muß deshalb nur einmal vor der Datenübertragung richtig gesetzt, d.h. auf das andere Daten-Terminal eingestellt werden. Das COMMAND-Register hat wichtige Funktionen während der Datenübertragung. Die Bits 5 bis 7 müssen nur einmal richtig gesetzt werden, um eine eventuelle Paritätsüberprüfung bzw. -generierung auszuwählen. Über Bit 4 bestimmt man, ob der ACIA die ankommenden Daten sofort (um ein halbes Bit zeitvergrößert) wieder zum Sender zurückschicken soll. In diesem Fall muß der normale Sender des ACIA ausgeschaltet sein (Bit 2 und 3) gleich 0.

Die Bits 2 und 3 haben zwei Aufgaben. Einmal steuern sie das Hardware-Handshaking über den RTS auf high, bedeutet dies, daß der ACIA nicht bereit ist, weitere Daten zu empfangen. Außerdem kann man mit ihnen den Send-Interrupt des ACIA ein- bzw. ausschalten. Ebenfalls über diese beiden Bits kann der ACIA veranlaßt werden, ein

Break-Zeichen zu schicken, das den Sender stoppt. Über Bit 1 kann man dem ACIA mitteilen, ob er Interrupts erzeugen darf oder nicht. Bit 0 meldet die Bereitschaft der Schnittstelle. Auf die genaue Benutzung dieser beiden Register gehen wir gleich noch näher ein.

Doch nun zum prinzipiellen Ablauf einer Datenübertragung. Zuerst wollen wir uns mit dem Senden von Daten beschäftigen. Sobald der ACIA dazu bereit ist, meldet er dies mit einem Low-Signal am IRQ-Ausgang. Voraussetzung dafür ist aber, daß das Erzeugen von Interrupts generell (COMMAND-Register Bit 1 = 0) und auch speziell des Sender-Interrupts (Bit 3 und 2) erlaubt ist. Die CPU reagiert auf das IRQ Signal und sucht nach dessen Verursacher. Dabei liest sie auch das STATUS-Register des ACIA. Dort kann man feststellen, ob der IRQ vom ACIA ausgelöst wurde und, wenn ja, aus welchem Grund (s. Bit-Belegung STATUS-Register). In diesem Fall müssen Bit 4 und 7 gesetzt sein. Nun schreibt man das nächste zu sendende Byte in das DATA-Register, und den Rest übernimmt der ACIA.

Das Empfangen von Daten läuft im Prinzip genauso ab. Wieder meldet der ACIA per IRQ, daß er ein Daten-Byte empfangen hat. Auch nun gibt die Abfrage des STATUS-Registers Aufschluß über den Grund des IRQ. Ist der Status in Ordnung, liebt man das Daten-Byte im DATA-Register. Dies muß natürlich vor dem nächsten Empfangs-Interrupt geschehen, da sonst ein Überlauffehler auftritt.

### Installation des RS-232-Handlers

Zur Kommunikation des Atari mit dem ACIA benötigen wir ein Programm, das einige Forderungen erfüllen muß:

- Reset-Festigkeit
- automatisches Laden des Programms beim Booten des DOS

- einfache Handhabung der Schnittstelle über herkömmliche Befehle (GET, PUT usw.)
- einfaches Einstellen der Parameter (Parität, Baud-Rate usw.)

Dazu wollen wir einen Handler programmieren, der über den Namen R: anzusprechen ist. Da dieses Programm automatisch geladen werden soll, wird es als AUTORUN.SYS auf der DOS-Diskette abgespeichert. Der so gebootete Handler ist nun zu schützen, d.h., dem Betriebssystem muß mitgeteilt werden, daß der freie Speicher (z.B. des Basic) erst nach dem Handler beginnt. Zunächst wird DOS 2.5 im Bereich von \$700 bis \$1C6C geladen, direkt danach der R:-Treiber von \$1EE0 bis \$21CD. Die Adresse des untersten freien RAM-Bytes (\$2400) kommt in MEMLO.

Natürlich soll der Treiber auch noch nach jedem Reset aktiv sein. Dazu verbiegen wir den Vektor DOSINI, über den normalerweise das DOS nach jedem Reset initialisiert wird, auf eine eigene Routine, die den Handler, die Handler-Tabelle, aber auch den ACIA initialisiert. Den ursprünglichen Wert von DOSINI müssen wir uns natürlich merken, um den normalen Ablauf weiterzuführen.

Nun ist unser Programm sozusagen nicht mehr kleinzukriegen. Aber schon taucht das nächste Problem auf. Der ACIA macht sich ja immer mit einem IRQ bemerkbar, wenn er der CPU etwas mitteilen will. Natürlich ist im Betriebssystem der Atari ein IRQ, der vom ACIA kommt, zu nächst nicht vorgesehen. Zum Glück haben die Entwickler bei Atari weitergedacht, so daß beim Auftreten eines IRQ durch den Vektor VIMIRQ gesprungen wird. Nun ist alles ganz einfach. Wir merken uns die Adresse in VIMIRO, schreiben die unserer eigenen IRQ-Routine ein und springen (nachdem unsere Routine jetzt vom Betriebssystem

aufgerufen wurde) danach zur gemerkten Adresse.

### Die Interrupt-Routine des Handlers

Diese Routine wird immer dann aufgerufen, wenn ein IRQ auftritt. Zunächst müssen wir also mit Hilfe des STATUS-Registers (Bit 7) feststellen, ob der IRQ vom ACIA ausgelöst wurde. Falls dies der Fall war, wird dieses Register ausgewertet, um herauszufinden, was darauf zu reagieren ist. Dabei gibt es drei Möglichkeiten.

1. Der ACIA möchte das nächste zu sendende Byte entgegennehmen.

In diesem Fall wird das nächste Byte aus dem Sendebuffer in das DATA-Register des ACIA geschrieben. Dieser Buffer (OBUF, im Handler mit 256 Byte Länge angelegt) hat den Vorteil, daß sich kürzere Datensätze blitzschnell vom Basic aus übertragen lassen. Sie landen dabei zunächst im Buffer, von wo sie dann nach und nach ausgegeben werden. Zur Verwaltung dieses Buffers gibt es zwei Pointer, PPOI und TPOI. PPOI gibt an, an welcher Stelle im Buffer das nächste Byte bei einem PUT- oder PRINT-Befehl geschrieben werden muß. TPOI besagt, welches Byte vom Buffer als nächstes an den ACIA zu übergeben ist. Da der Buffer nur 256 Bytes lang ist, kann es natürlich sein, daß beim schnellen Drucken PPOI so schnell erhöht wird, daß er TPOI überholen würde. In diesem Fall muß der Handler warten, bis wieder Platz im Buffer ist. Dann kommt es im (Basic-Programm) zu Wartezeiten. Natürlich ist noch zu berücksichtigen, daß kein Byte ausgegeben wird, wenn der Buffer leer ist (PPOI = TPOI). Es muß aber auch darauf geachtet werden, ob die Gegenstelle vielleicht ein Xoff-Zei-

chen gesendet hat. In diesem Fall dürfen wir keine weiteren Daten senden, bis die Gegenstelle wieder ein Xon meldet. Diese Art des Übertragungsprotokolls (mit Xon und Xoff) nennt man Software Handshaking. Das Abfragen des CTS-Eingangs für das Hardware-Handshaking übernimmt der ACIA selbst.

## 2. Der ACIA hat ein Daten-Byte empfangen.

Auch hier existiert ein Buffer (IBUF, 256 Bytes). Es wird also das Byte im DATA-Register des ACIA in den Empfangs-Buffer an die Stelle des Pointers RPOI geschrieben. Falls der Buffer nun voll ist, muß der Handler dies der Gegenstelle mitteilen, damit diese zu senden aufhört. Das kann über die Leitung RTS geschehen oder durch Senden des Wertes in STOP (Xoff). Wenn wieder Platz im Buffer ist, ist entweder RTS auf low zu setzen oder der Wert in CONT (Xon) zu senden. Mit GET oder INPUT werden dann Daten aus dem Buffer an der Stelle IPOI ausgelesen. Diesmal ist zu beachten, daß der Buffer leer sein kann (IPOI = RPOI), so daß kein Byte gelesen werden kann. Dann wartet der GET-Befehl im Normalfall, bis ein Byte empfangen wird.

## 3. Der Status von DSR oder DCD hat sich geändert.

Je nach Status des Akustikkopplers oder Computers können diese Meldungen entfallen oder verschieden sein. Deshalb läßt sich im Handler über CHECK bestimmen, ob und welches der beiden Signale beachtet wird. Näheres folgt später.

## Die einzelnen Handler-Routinen

Zu jedem richtigen Handler gehören die verschiedenen Treiber-routinen OPEN, CLOSE,

PROT	\$6F0	0 = kein Handshake, 1 = Software-Handshake, 2 = Hardware-Handshake Hardware-Handshake bedeutet: - Der ACIA sendet über RTS-Ausgang seine Bereitschaft (RTS low). - Der ACIA empfängt über CTS-Eingang Bereitschaftssignale der Gegenstelle. Software-Handshake bedeutet: - Der ACIA sendet von Xon/Xoff-Zeichen als Bereit-/Nichtbereitmeldung. Der ACIA empfängt Xon/Xoff von der Gegenstelle.
STOP	\$6F1	Zeichen für Xoff bei Softhandshake (norm. 19)
CONT	\$6F2	Zeichen für Xon bei Softhandshake (norm. 17)
FULLFL	\$6F3	0 = im Input-Buffer ist Platz, 1 = Input-Buffer voll
DATABRK	\$6F4	0 = Normalzustand, 1 = Handler wartet auf Gegenstelle
ERR	\$6F5	0 = alles o.k. sonst Fehler aufgetreten
STAT	\$6F6	ACIA-Status (ACSTAT)
CHECK	\$6F7	0 = weder DSR noch DCD abfragen, 32 = nur DCD abfragen, 64 = nur DSR abfragen, 96 = beide abfragen
OFF	\$6F8	0 = Normalzustand, 1 = nur noch Output-Buffer senden, dann Übertragung stoppen.
RWFL	\$6F9	12 = bei GET warten, wenn Input-Buffer leer, 13 = nicht warten
NOCHR	\$6FA	0 = Zeichen empfangen, 1 = kein Zeichen empfangen (nach GET-Befehl)

Die Variablen des Treiberprogramms.

PUT, GET und bei uns auch SPECIAL. Doch fallen diese für unsere Schnittstelle überraschenderweise recht kurz aus, denn die Interrupt-Routine nimmt uns schon viel Arbeit ab. Doch nun zu den Programmteilen.

### OPEN

Hier wird eigentlich nur der Gegenstelle mitgeteilt, daß unser Computer zur Datenübertragung bereit ist. Außerdem wird das Flag für den Wait-Modus (s. später) entsprechend dem ersten Parameter des OPEN-Befehls gesetzt. Wie bei jeder Handler-Routine erfolgt vor dem Rücksprung im Y-Register die Übergabe der Fehlernummer (1 = Status OK).

### CLOSE

Es wird nun das Stop-Flag (OFF) gesetzt. Dieses teilt der IRQ-Routine mit, daß nur noch der Buffer gesendet und dann die Gegenstelle ausgeschaltet wird.

### GET

Falls kein Fehler aufgetreten ist und die BREAK-Taste nicht gedrückt wurde, wird das aktuelle

Zeichen aus dem Buffer gelesen. Ist der Input-Buffer leer, hängt es davon ab, ob der Wait-Modus aktiviert ist oder nicht. Im ersten Fall wartet die GET-Routine so lange, bis ein Zeichen empfangen wurde, im zweiten wird die GET-Routine mit dem Rückgabewert 0 verlassen und das Flag NOCHR (\$6FA) gesetzt. War der Input-Buffer voll, wird auch FULLFL zurückgesetzt, um der IRQ-Routine mitzuteilen, daß nun wieder Platz für Zeichen ist. Dies geschieht aber nur, wenn mindestens sechs Zeichen im Input-Buffer frei sind.

### PUT

Zuerst wird der ACIA-Sende-Interrupt eingeschaltet und die BREAK-Taste abgefragt. Sofern noch Platz im Output-Buffer ist, wird ein Zeichen in diesen geschrieben, andernfalls gewartet, bis wieder Platz frei ist.

### SPECIAL

Diese Routine ist für alle XIO-Befehle zuständig. Je nach XIO-Nummer wird zu verschiedenen kleinen Unterroutinen gesprungen. Eine Beschreibung der ein-

zelen XIOs folgt später.

Sollten Ihnen noch verschiedene Teile des Handlers unverständlich sein, schauen Sie sich doch das abgedruckte Listing an, denn es ist ausführlich kommentiert. Eine Liste aller vom Handler verwendeten (eigenen) Variablen finden Sie im Kasten.

### Bedienung des Handlers

- Folgende Basic-Befehle werden vom R:-Handler unterstützt:
- OPEN #Kanal, Wait-Modus (12 oder 13), 0, "R:" GET #Kanal, Byte
- PUT #Kanal, Byte
- PRINT #Kanal, Text
- INPUT #Kanal, Text
- XIO 40, #Kanal, Wert für COMMAND-Register des ACIA, Wert für CONTROL-Reg., "R":
- XIO 41, #Kanal für PROT, 0, "R:"
- XIO 42, #, Wert für STOP, Wert für CONT, "R:"

Nach RESET wird der Handler bzw. der ACIA auf 9600

Baud, keine Parität, 1 Stopbit und Software-Handshake initialisiert. STOP erhält den Standardwert 19 (CTRL-S), CONT den Wert 17 (CTRL-Q) und CHECK den Wert 32.

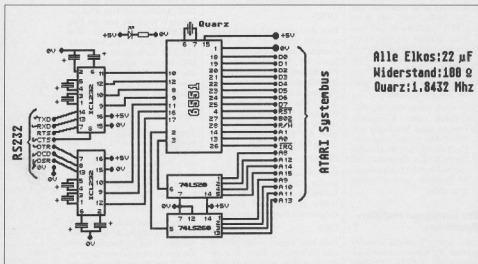
### Die Programme

Listing 1 zeigt den R:-Handler als Atmas-2-Quellcode und ist für alle Assembler-Kundigen gedacht. Da Hardware-Erweiterungen auch für Basic-Programmierer interessant sind, haben wir in Listing 2 den gleichen Handler als "AMD"-File abgedruckt. Listing 3 ist ein kleines Basic-Programm, mit dem schon erste Kontaktversuche unternommen kann. Wir haben mit diesem Miniterminal-Programm mit dem VTS2-Emulator des Atari ST kommuniziert, und das klappte einwandfrei.

Wenn Sie den Handler in Basic benutzen wollen, müssen Sie das Handlerfile als AUTORUN.SYS auf ihrer DOS-2.5-Diskette legen. Dazu tippen Sie das "AMD"-Listing ab (Name AUTORUN.SYS), oder Sie geben den Quellcode mit Atmas ein. In

diesem Fall müssen Sie Ihr fertiges (assembliertes) Programm mit dem Monitor abspeichern (\$1EE0 bis \$21CD) und die Startadresse \$1EE0 an dieses File anhängen (s. Assemblerecke 6/88).

Sie können den Handler auch in Ihre eigenen Assembler-Programme einbinden oder nachladen. In diesem Fall müssen Sie ihn aber erst initialisieren. Das geschieht mit einem Sprung zu (JSR) \$1EE0. Dadurch wird der Handler eingetragen und resetfest gemacht. Damit der Handler nur eingetragen wird, ist ein Sprung zu \$1EF2 erforderlich. Sie können das Handlerfile auch im ATMAS-(Monitor) oder BI-BO-Assembler (BLOAD) laden. In beiden Fällen ist der Handler über \$1EF2 zu initialisieren. Eventuell müssen Sie ihn dann an eine andere Stelle assemblieren. Wenn er nicht mehr direkt nach dem DOS liegt, entfällt die Änderung von MEMLO im Unterprogramm VECSET! Auch können Sie die Adressen für die Handler-Variablen (PROT, FULLFL usw.) beliebig abwandeln.



Alle Elkos: 22 µF  
Widerstand: 100 Ω  
Quarz: 1.8432 Mhz

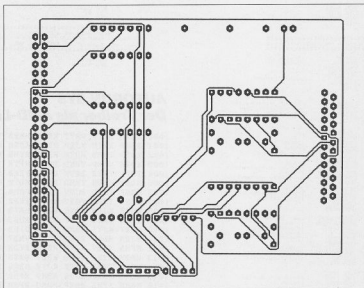




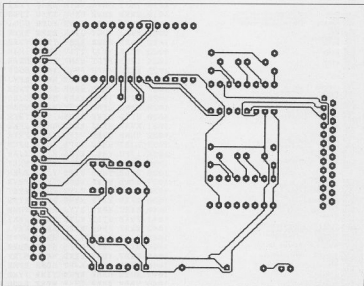




Die Oberseite  
der Platine...



...und hier die  
Unterseite



# Was ist eine Mini-SPEEDY?

Wie von Ihnen können sicher die SPEEDY 1050 und schätzen die Leistungen dieses Floppy-Speeders. Mini-SPEEDY bedeutet aber nicht, daß die Leistungen der Mini-SPEEDY gegenüber der SPEEDY 1050 kleiner geworden sind. Lediglich die Platte ist kleiner geworden. Allerdings können bei der Mini-SPEEDY seine Hardware-Erweiterungen mehr vorgenommen werden, wie es bei der SPEEDY 1050 möglich ist.

## Hier alle Leistungen der Mini-SPEEDY im Überblick:

- 3 Speicherchips
  - Single Density 88 KByte
  - Medium Density 126 KByte
  - Double Density 180 KByte
- 4 Geschwindigkeiten
  - 1050 Normal (ca. 19.200 Baud)
  - Speedy Normal
  - Speedy High-Speed (ca. 78.000 Baud)
  - Speedy Super-Speed (ca. 96.000 Baud)
 durch mittelgefertigter HSS-Sektor-Kopierer
- formatiert in 9 Sekunden
- sehr leichter Einbau
- modulare Bauweise
- BIBO-DOS im Lieferumfang enthalten
- kann geschützte Software kopieren
- wird ständig erweitert und verbessert
- ständig wachsender Softwaremarkt
- arbeitet mit dem 16-K-BIBOMON zusammen
- sehr hohe Datensicherheit
- 8-KByte-Trackbuffer
- 4-KByte-RAM durch User nutzbar
- Cache-Speicher für schnelles Arbeiten
- Leicht zu programmieren
- volle User-Unterstützung
- deutsches Handbuch
- deutsche Programme
- Update-Garantie bei neuer Software
- vieles mehr.

Wie Sie sehen, sind die Leistungen der Mini-Speedy mit denen der SPEEDY 1050 völlig identisch. Der Hauptunterschied zwischen der Mini-SPEEDY und der SPEEDY 1050 besteht in folgenden Punkten:

Die SPEEDY 1050 gibt es in acht verschiedenen Ausführungen, die Mini-SPEEDY nur noch in drei. Ingegengesetzt ist die aufwendigste Track- und Density-Kopiergeräts und der Summe. Dadurch haben wir die Platine um mehr als 50% verkleinern können. Das wiederum hat sich auf den Verkaufspreis der Mini-SPEEDY ausgewirkt!

Doch sonst ist die Mini-SPEEDY 100% kompatibel zur SPEEDY 1050! Das bedeutet, alles was für die SPEEDY 1050 lieferbar ist, gilt es auch für die Mini-SPEEDY! Alle Programme, die für die SPEEDY 1050 geschrieben wurden, laufen auch auf der Mini-SPEEDY! Und nicht zuletzt ist die gesamte User-Unterstützung, die es für die SPEEDY 1050 gibt, auch auf die Mini-SPEEDY überbart!

## Hier die Preise für die neue Mini-SPEEDY:

Mini-SPEEDY N .....	85,- DM
Homounterstützung ohne Floppy	
Mini-SPEEDY D .....	105,- DM
16 BDD-DOS im RAM	
Mini-SPEEDY S .....	105,- DM
16 HSS-Sektor-Kopierer im ROM	

## TEST-GARANTIE!

Sie können die Mini-SPEEDY völlig ohne Risiko 14 Tage lang testen!

Wie funktioniert das?

Danz enthält wir geben Ihnen die Möglichkeit, eine Version der Mini-SPEEDY zu Hause in aller Ruhe zu testen. Sie haben 14 Tage lang Zeit, sich alle Funktionen der Mini-SPEEDY genauere anzusehen. Prüfen Sie ganz genau, ob die Mini-SPEEDY das ist, was wir Ihnen versprochen!

Wenn Sie unser Angebot annehmen möchten, fordern Sie ausführliche Unterlagen und einen Bestellzettel bei uns an! Alle, die die Mini-SPEEDY gleich kaufen wollen, können den obgedruckten Bestellzettel verwenden.

## BESTELLSCHEIN

Ich möchte bei Ihnen die angekreuzte

Mini-SPEEDY-Variante bestellen.

- Mini-SPEEDY N ..... 85,- DM
- Mini-SPEEDY D ..... 105,- DM
- Mini-SPEEDY S ..... 105,- DM

Ich bitte um sofortige und kostenlose

Lieferung

Ergebenes an:

Compy Shop OHG

Grünenaustraße 29, D-4330 Mülheim, Tel. 02 06 / 49 71 09

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```

1049 RfVJ RVMR NUKJ RTHB MKRF KJRR 31307
1050 LVDF YTKB MURF MRTU KBMB RFUH 31033
1051 NBMV RFVJ RKBK EHYR CYYR KJRR 32078
1052 HBMU RFKV MBRF NNMB RfCJ RRYR 31480
1053 KRRT FRKE HRFR KBMD RfVJ RTBR 31151
1054 RKKB MDRF YJRU BRFR KRHM FRKR 31197
1055 HMFR KRHJ FRHJ KBRY BTYJ MURJ 31481
1056 RIHB RYBT KDTT MBRY BDBM RPTH 30743
1057 FJRY VBMM RfMR MKVH MHRF NNHM 31508
1058 RFFH JJRR YUKR RfFR KDYY VJYH 32282
1059 MRKX VJYJ MRTI VJYK MHTX KRJY 32130
1060 FRKD YKHB RYBT KDVC HBRU BTKR 31007
1061 RTFR RfYK HBMR RfKR RTFR KDYK 31306
1062 DHMT RPKD YCHB MYRF KRRT FRRR 31610
1063 MMM NRRY NTRY NRTN 15106 *
    
```

## RS232-Terminal

```

10 REM *****BASIC*****
20 REM *****PROGRAM*****
30 REM
40 REM *****
50 REM *****
60 REM *****
70 KEYTAB=PEEK(121)+256*PEEK(122)
80 NOCHR=1786
90 POKE 82,0
100 ? CHR$(125)
110 X10 40,#1,9,22,"R:"
120 X10 41,#1,1,0,"R:"
130 X10 42,#1,19,17,"R:"
140 OPEN #1,3,0,"R:"
150 POKE 764,255
160 GET #1,A
170 IF A=13 THEN A=155
180 IF A=10 THEN 210
190 IF PEEK(NOCHR)=1 THEN 210
200 ? CHR$(A);
210 IF PEEK(764)=255 THEN 160
220 KEY=PEEK(KEYTAB+PEEK(764))
230 IF KEY=155 THEN PUT #1,13:KEY=10
240 PUT #1,KEY
250 POKE 764,255
260 GOTO 160
    
```

**NEU** (2) Risk / 1/1 Franken / 1/1 Optionen / 1/1 Mode

**SCHALT-PLAN** Schaltungs-  
mit mind. 323 Ohm  
DR 124 - 97 104  
über 128 Bausteine  
auf dem Bauplatz  
mit dem 63107  
ausreichend viele  
Bausteine

**DM 98,-**  
incl. Versand  
zz + 3,- 5,00  
Info Nr. - 88  
Ordnung

**SCHALT-PLAN** mit Bausteine EDITOR  
Wählwerk

Autom. Bausteine Liste, 1/1 Bausteine  
über, andere von  
Sinter, Fotoresist einlesen  
Programme arbeiten angeschlossen  
mit zwei Bildschirmen gleichzeitig GEM 043

**und vieles mehr...**

**Jürgen Dürr Einsteinstr. 6 6526 Worms 26**  
Tel. 06241/34148 - 02 - 34149 - 0 30 - 332 - 34149 01 - 13

